

OpenCOBOL

1.1pre-rel

Generated by Doxygen 1.7.4

Sun May 1 2011 16:55:26



# Contents

<b>1</b>	<b>OpenCOBOL 1.1pre-rel source tree</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>Directory Hierarchy</b>	<b>3</b>
2.1	Directories . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Directory Documentation</b>	<b>9</b>
5.1	bin/ Directory Reference . . . . .	9
5.2	cobc/ Directory Reference . . . . .	10
5.3	lib/ Directory Reference . . . . .	11
5.4	libcob/ Directory Reference . . . . .	12
<b>6</b>	<b>Class Documentation</b>	<b>13</b>
6.1	__cob_screen Struct Reference . . . . .	13
6.1.1	Detailed Description . . . . .	14
6.1.2	Member Data Documentation . . . . .	14
6.1.2.1	attr . . . . .	14
6.1.2.2	backg . . . . .	14
6.1.2.3	child . . . . .	14
6.1.2.4	column . . . . .	14
6.1.2.5	field . . . . .	14

6.1.2.6	foreg	14
6.1.2.7	line	15
6.1.2.8	next	15
6.1.2.9	occurs	15
6.1.2.10	type	15
6.1.2.11	value	15
6.2	call_hash Struct Reference	15
6.2.1	Detailed Description	16
6.2.2	Member Data Documentation	16
6.2.2.1	cancel	16
6.2.2.2	flag_is_active	16
6.2.2.3	func	16
6.2.2.4	name	16
6.2.2.5	next	16
6.3	cb_alphabet_name Struct Reference	16
6.3.1	Detailed Description	17
6.3.2	Member Data Documentation	17
6.3.2.1	cname	17
6.3.2.2	common	17
6.3.2.3	custom_list	18
6.3.2.4	high_val_char	18
6.3.2.5	low_val_char	18
6.3.2.6	name	18
6.3.2.7	type	18
6.4	cb_alt_key Struct Reference	18
6.4.1	Detailed Description	19
6.4.2	Member Data Documentation	19
6.4.2.1	duplicates	19
6.4.2.2	key	19
6.4.2.3	next	19
6.4.2.4	offset	19
6.5	cb_assign Struct Reference	19
6.5.1	Detailed Description	20
6.5.2	Member Data Documentation	20

---

6.5.2.1	common	20
6.5.2.2	val	20
6.5.2.3	var	20
6.6	cb_binary_op Struct Reference	21
6.6.1	Detailed Description	21
6.6.2	Member Data Documentation	21
6.6.2.1	common	21
6.6.2.2	op	22
6.6.2.3	x	22
6.6.2.4	y	22
6.7	cb_call Struct Reference	22
6.7.1	Detailed Description	23
6.7.2	Member Data Documentation	23
6.7.2.1	args	23
6.7.2.2	common	23
6.7.2.3	is_system	23
6.7.2.4	name	23
6.7.2.5	returning	23
6.7.2.6	stmt1	23
6.7.2.7	stmt2	23
6.8	cb_cast Struct Reference	24
6.8.1	Detailed Description	24
6.8.2	Member Data Documentation	24
6.8.2.1	common	24
6.8.2.2	type	24
6.8.2.3	val	25
6.9	cb_class_name Struct Reference	25
6.9.1	Detailed Description	25
6.9.2	Member Data Documentation	25
6.9.2.1	cname	26
6.9.2.2	common	26
6.9.2.3	list	26
6.9.2.4	name	26
6.10	cb_const Struct Reference	26

6.10.1	Detailed Description	27
6.10.2	Member Data Documentation	27
6.10.2.1	common	27
6.10.2.2	val	27
6.11	cb_continue Struct Reference	27
6.11.1	Detailed Description	28
6.11.2	Member Data Documentation	28
6.11.2.1	common	28
6.12	cb_decimal Struct Reference	28
6.12.1	Detailed Description	28
6.12.2	Member Data Documentation	29
6.12.2.1	common	29
6.12.2.2	id	29
6.13	cb_exception Struct Reference	29
6.13.1	Detailed Description	29
6.13.2	Member Data Documentation	29
6.13.2.1	code	29
6.13.2.2	enable	29
6.13.2.3	name	29
6.14	cb_field Struct Reference	30
6.14.1	Detailed Description	32
6.14.2	Member Data Documentation	32
6.14.2.1	children	32
6.14.2.2	common	32
6.14.2.3	count	32
6.14.2.4	ename	33
6.14.2.5	false_88	33
6.14.2.6	file	33
6.14.2.7	flag_any_length	33
6.14.2.8	flag_anylen_done	33
6.14.2.9	flag_base	33
6.14.2.10	flag_binary_swap	33
6.14.2.11	flag_blank_zero	33
6.14.2.12	flag_chained	33

---

6.14.2.13 flag_external . . . . .	33
6.14.2.14 flag_field . . . . .	34
6.14.2.15 flag_indexed_by . . . . .	34
6.14.2.16 flag_invalid . . . . .	34
6.14.2.17 flag_is_c_long . . . . .	34
6.14.2.18 flag_is_global . . . . .	34
6.14.2.19 flag_is_pdiv_parm . . . . .	34
6.14.2.20 flag_is_pointer . . . . .	34
6.14.2.21 flag_is_verified . . . . .	34
6.14.2.22 flag_item_78 . . . . .	34
6.14.2.23 flag_item_based . . . . .	34
6.14.2.24 flag_item_external . . . . .	35
6.14.2.25 flag_justified . . . . .	35
6.14.2.26 flag_local . . . . .	35
6.14.2.27 flag_local_allocated . . . . .	35
6.14.2.28 flag_no_init . . . . .	35
6.14.2.29 flag_occurs . . . . .	35
6.14.2.30 flag_real_binary . . . . .	35
6.14.2.31 flag_sign_leading . . . . .	35
6.14.2.32 flag_sign_separate . . . . .	35
6.14.2.33 flag_spare . . . . .	35
6.14.2.34 flag_synchronized . . . . .	36
6.14.2.35 id . . . . .	36
6.14.2.36 index_list . . . . .	36
6.14.2.37 index_qual . . . . .	36
6.14.2.38 indexes . . . . .	36
6.14.2.39 keys . . . . .	36
6.14.2.40 level . . . . .	36
6.14.2.41 memory_size . . . . .	36
6.14.2.42 name . . . . .	36
6.14.2.43 nkeys . . . . .	36
6.14.2.44 occursDepending . . . . .	36
6.14.2.45 occurs_max . . . . .	37
6.14.2.46 occurs_min . . . . .	37

6.14.2.47	offset	37
6.14.2.48	param_num	37
6.14.2.49	parent	37
6.14.2.50	pic	37
6.14.2.51	redefines	37
6.14.2.52	rename_thru	37
6.14.2.53	screen_backg	37
6.14.2.54	screen_column	37
6.14.2.55	screen_flag	38
6.14.2.56	screen_foreg	38
6.14.2.57	screen_from	38
6.14.2.58	screen_line	38
6.14.2.59	screen_to	38
6.14.2.60	sister	38
6.14.2.61	size	38
6.14.2.62	storage	38
6.14.2.63	storage_id	38
6.14.2.64	usage	38
6.14.2.65	values	39
6.15	cb_file Struct Reference	39
6.15.1	Detailed Description	41
6.15.2	Member Data Documentation	41
6.15.2.1	access_mode	41
6.15.2.2	alt_key_list	41
6.15.2.3	assign	42
6.15.2.4	cname	42
6.15.2.5	common	42
6.15.2.6	external	42
6.15.2.7	external_assign	42
6.15.2.8	file_status	42
6.15.2.9	fileid_assign	42
6.15.2.10	finalized	42
6.15.2.11	global	42
6.15.2.12	handler	42



---

6.15.2.13 handler_prog . . . . .	43
6.15.2.14 key . . . . .	43
6.15.2.15 latbot . . . . .	43
6.15.2.16 latfoot . . . . .	43
6.15.2.17 lattop . . . . .	43
6.15.2.18 lineage . . . . .	43
6.15.2.19 lineage_ctr . . . . .	43
6.15.2.20 lock_mode . . . . .	43
6.15.2.21 name . . . . .	43
6.15.2.22 optional . . . . .	43
6.15.2.23 organization . . . . .	44
6.15.2.24 record . . . . .	44
6.15.2.25 recordDepending . . . . .	44
6.15.2.26 record_max . . . . .	44
6.15.2.27 record_min . . . . .	44
6.15.2.28 same_clause . . . . .	44
6.15.2.29 sharing . . . . .	44
6.15.2.30 special . . . . .	44
6.16 cb_funcall Struct Reference . . . . .	44
6.16.1 Detailed Description . . . . .	45
6.16.2 Member Data Documentation . . . . .	45
6.16.2.1 argc . . . . .	45
6.16.2.2 argv . . . . .	45
6.16.2.3 common . . . . .	45
6.16.2.4 name . . . . .	46
6.16.2.5 screenptr . . . . .	46
6.16.2.6 varcnt . . . . .	46
6.17 cb_goto Struct Reference . . . . .	46
6.17.1 Detailed Description . . . . .	47
6.17.2 Member Data Documentation . . . . .	47
6.17.2.1 common . . . . .	47
6.17.2.2 depending . . . . .	47
6.17.2.3 target . . . . .	47
6.18 cb_if Struct Reference . . . . .	47

6.18.1	Detailed Description	48
6.18.2	Member Data Documentation	48
6.18.2.1	common	48
6.18.2.2	stmt1	48
6.18.2.3	stmt2	48
6.18.2.4	test	48
6.19	cb_initialize Struct Reference	48
6.19.1	Detailed Description	49
6.19.2	Member Data Documentation	49
6.19.2.1	common	49
6.19.2.2	def	49
6.19.2.3	flag_statement	50
6.19.2.4	rep	50
6.19.2.5	val	50
6.19.2.6	var	50
6.20	cb_integer Struct Reference	50
6.20.1	Detailed Description	51
6.20.2	Member Data Documentation	51
6.20.2.1	common	51
6.20.2.2	val	51
6.21	cb_intrinsic Struct Reference	51
6.21.1	Detailed Description	52
6.21.2	Member Data Documentation	52
6.21.2.1	args	52
6.21.2.2	common	52
6.21.2.3	intr_field	52
6.21.2.4	intr_tab	52
6.21.2.5	length	52
6.21.2.6	name	52
6.21.2.7	offset	53
6.22	cb_intrinsic_table Struct Reference	53
6.22.1	Detailed Description	53
6.22.2	Member Data Documentation	53
6.22.2.1	args	53

---

6.22.2.2	category	53
6.22.2.3	implemented	53
6.22.2.4	intr_enum	53
6.22.2.5	intr_routine	54
6.22.2.6	name	54
6.22.2.7	refmod	54
6.23	cb_field::cb_key Struct Reference	54
6.23.1	Detailed Description	55
6.23.2	Member Data Documentation	55
6.23.2.1	dir	55
6.23.2.2	key	55
6.23.2.3	ref	55
6.23.2.4	val	55
6.24	cb_label Struct Reference	55
6.24.1	Detailed Description	56
6.24.2	Member Data Documentation	56
6.24.2.1	children	57
6.24.2.2	common	57
6.24.2.3	exit_label	57
6.24.2.4	exit_label_ref	57
6.24.2.5	id	57
6.24.2.6	is_entry	57
6.24.2.7	is_global	57
6.24.2.8	is_section	57
6.24.2.9	name	57
6.24.2.10	need_begin	57
6.24.2.11	need_return	58
6.24.2.12	orig_name	58
6.24.2.13	section	58
6.24.2.14	spare	58
6.25	cb_level_78 Struct Reference	59
6.25.1	Detailed Description	60
6.25.2	Member Data Documentation	60
6.25.2.1	fld78	60

6.25.2.2	next	60
6.26	cb_list Struct Reference	60
6.26.1	Detailed Description	61
6.26.2	Member Data Documentation	61
6.26.2.1	chain	61
6.26.2.2	common	61
6.26.2.3	purpose	61
6.26.2.4	sizes	61
6.26.2.5	value	61
6.27	cb_literal Struct Reference	61
6.27.1	Detailed Description	62
6.27.2	Member Data Documentation	62
6.27.2.1	all	62
6.27.2.2	common	62
6.27.2.3	data	62
6.27.2.4	scale	63
6.27.2.5	sign	63
6.27.2.6	size	63
6.27.2.7	spare	63
6.28	cb_locale_name Struct Reference	63
6.28.1	Detailed Description	64
6.28.2	Member Data Documentation	64
6.28.2.1	cname	64
6.28.2.2	common	64
6.28.2.3	list	64
6.28.2.4	name	64
6.29	cb_perform Struct Reference	64
6.29.1	Detailed Description	65
6.29.2	Member Data Documentation	65
6.29.2.1	body	65
6.29.2.2	common	66
6.29.2.3	cycle_label	66
6.29.2.4	data	66
6.29.2.5	exit_label	66

---

6.29.2.6	test	66
6.29.2.7	type	66
6.29.2.8	varying	66
6.30	cb_perform_varying Struct Reference	66
6.30.1	Detailed Description	67
6.30.2	Member Data Documentation	67
6.30.2.1	common	67
6.30.2.2	from	67
6.30.2.3	name	68
6.30.2.4	step	68
6.30.2.5	until	68
6.31	cb_picture Struct Reference	68
6.31.1	Detailed Description	69
6.31.2	Member Data Documentation	69
6.31.2.1	category	69
6.31.2.2	common	69
6.31.2.3	digits	69
6.31.2.4	have_sign	69
6.31.2.5	lenstr	69
6.31.2.6	orig	69
6.31.2.7	scale	69
6.31.2.8	size	70
6.31.2.9	spare	70
6.31.2.10	str	70
6.32	cb_program Struct Reference	70
6.32.1	Detailed Description	73
6.32.2	Member Data Documentation	73
6.32.2.1	alphabet_name_list	73
6.32.2.2	cb_call_params	73
6.32.2.3	cb_return_code	73
6.32.2.4	cb_sort_return	73
6.32.2.5	class_name_list	73
6.32.2.6	class_spec_list	73
6.32.2.7	collating_sequence	73

---

6.32.2.8 crt_status . . . . .	74
6.32.2.9 currency_symbol . . . . .	74
6.32.2.10 cursor_pos . . . . .	74
6.32.2.11 decimal_index . . . . .	74
6.32.2.12 decimal_index_max . . . . .	74
6.32.2.13 decimal_point . . . . .	74
6.32.2.14 entry_list . . . . .	74
6.32.2.15 exec_list . . . . .	74
6.32.2.16 file_list . . . . .	74
6.32.2.17 flag_chained . . . . .	74
6.32.2.18 flag_common . . . . .	75
6.32.2.19 flag_global_use . . . . .	75
6.32.2.20 flag_initial . . . . .	75
6.32.2.21 flag_main . . . . .	75
6.32.2.22 flag_recursive . . . . .	75
6.32.2.23 flag_screen . . . . .	75
6.32.2.24 flag_validated . . . . .	75
6.32.2.25 function_spec_list . . . . .	75
6.32.2.26 gen_decset . . . . .	75
6.32.2.27 gen_file_error . . . . .	75
6.32.2.28 gen_ptrmanip . . . . .	76
6.32.2.29 gen_udecset . . . . .	76
6.32.2.30 global_file_list . . . . .	76
6.32.2.31 global_handler . . . . .	76
6.32.2.32 global_list . . . . .	76
6.32.2.33 interface_spec_list . . . . .	76
6.32.2.34 label_list . . . . .	76
6.32.2.35 linkage_storage . . . . .	76
6.32.2.36 local_file_list . . . . .	76
6.32.2.37 local_storage . . . . .	76
6.32.2.38 local_storage_file . . . . .	77
6.32.2.39 local_storage_name . . . . .	77
6.32.2.40 locale_list . . . . .	77
6.32.2.41 loop_counter . . . . .	77

---

6.32.2.42 nested_level . . . . .	77
6.32.2.43 next_program . . . . .	77
6.32.2.44 numeric_separator . . . . .	77
6.32.2.45 orig_source_name . . . . .	77
6.32.2.46 parameter_list . . . . .	77
6.32.2.47 prog_type . . . . .	77
6.32.2.48 program_id . . . . .	78
6.32.2.49 program_spec_list . . . . .	78
6.32.2.50 property_spec_list . . . . .	78
6.32.2.51 reference_list . . . . .	78
6.32.2.52 returning . . . . .	78
6.32.2.53 screen_storage . . . . .	78
6.32.2.54 source_name . . . . .	78
6.32.2.55 spare . . . . .	78
6.32.2.56 symbolic_list . . . . .	78
6.32.2.57 word_table . . . . .	78
6.32.2.58 working_storage . . . . .	79
6.33 cb_reference Struct Reference . . . . .	79
6.33.1 Detailed Description . . . . .	80
6.33.2 Member Data Documentation . . . . .	80
6.33.2.1 all . . . . .	80
6.33.2.2 chain . . . . .	80
6.33.2.3 check . . . . .	80
6.33.2.4 common . . . . .	80
6.33.2.5 length . . . . .	80
6.33.2.6 offset . . . . .	80
6.33.2.7 subs . . . . .	80
6.33.2.8 type . . . . .	81
6.33.2.9 value . . . . .	81
6.33.2.10 word . . . . .	81
6.34 cb_replace_list Struct Reference . . . . .	81
6.34.1 Detailed Description . . . . .	82
6.34.2 Member Data Documentation . . . . .	82
6.34.2.1 new_text . . . . .	82

---

6.34.2.2	next	82
6.34.2.3	old_text	82
6.35	cb_search Struct Reference	82
6.35.1	Detailed Description	83
6.35.2	Member Data Documentation	83
6.35.2.1	common	83
6.35.2.2	end_stmt	83
6.35.2.3	flag_all	84
6.35.2.4	table	84
6.35.2.5	var	84
6.35.2.6	whens	84
6.36	cb_statement Struct Reference	84
6.36.1	Detailed Description	85
6.36.2	Member Data Documentation	85
6.36.2.1	body	85
6.36.2.2	common	86
6.36.2.3	file	86
6.36.2.4	handler1	86
6.36.2.5	handler2	86
6.36.2.6	handler3	86
6.36.2.7	handler_id	86
6.36.2.8	name	86
6.36.2.9	need_terminator	86
6.36.2.10	null_check	86
6.37	cb_string Struct Reference	87
6.37.1	Detailed Description	87
6.37.2	Member Data Documentation	87
6.37.2.1	common	87
6.37.2.2	data	87
6.37.2.3	size	87
6.38	cb_system_name Struct Reference	88
6.38.1	Detailed Description	88
6.38.2	Member Data Documentation	88
6.38.2.1	category	88



---

6.38.2.2	common	88
6.38.2.3	token	89
6.39	cb_text_list Struct Reference	89
6.39.1	Detailed Description	89
6.39.2	Member Data Documentation	89
6.39.2.1	next	89
6.39.2.2	text	89
6.40	cb_tree_common Struct Reference	90
6.40.1	Detailed Description	90
6.40.2	Member Data Documentation	90
6.40.2.1	category	90
6.40.2.2	source_file	90
6.40.2.3	source_line	90
6.40.2.4	tag	90
6.41	cb_word Struct Reference	90
6.41.1	Detailed Description	91
6.41.2	Member Data Documentation	91
6.41.2.1	count	91
6.41.2.2	error	91
6.41.2.3	items	91
6.41.2.4	name	92
6.41.2.5	next	92
6.42	cob_alloc_cache Struct Reference	92
6.42.1	Detailed Description	92
6.42.2	Member Data Documentation	92
6.42.2.1	cob_pointer	92
6.42.2.2	next	93
6.42.2.3	size	93
6.43	cob_decimal Struct Reference	93
6.43.1	Detailed Description	93
6.43.2	Member Data Documentation	93
6.43.2.1	scale	93
6.43.2.2	value	93
6.44	cob_exception Struct Reference	93

6.44.1	Detailed Description	94
6.44.2	Member Data Documentation	94
6.44.2.1	code	94
6.44.2.2	critical	94
6.44.2.3	name	94
6.45	cob_external Struct Reference	94
6.45.1	Detailed Description	95
6.45.2	Member Data Documentation	95
6.45.2.1	ename	95
6.45.2.2	esize	95
6.45.2.3	ext_alloc	95
6.45.2.4	next	95
6.46	cob_field Struct Reference	95
6.46.1	Detailed Description	96
6.46.2	Member Data Documentation	96
6.46.2.1	attr	96
6.46.2.2	data	96
6.46.2.3	size	96
6.47	cob_field_attr Struct Reference	97
6.47.1	Detailed Description	97
6.47.2	Member Data Documentation	97
6.47.2.1	digits	97
6.47.2.2	flags	97
6.47.2.3	pic	97
6.47.2.4	scale	97
6.47.2.5	type	97
6.48	cob_file Struct Reference	98
6.48.1	Detailed Description	99
6.48.2	Member Data Documentation	99
6.48.2.1	access_mode	99
6.48.2.2	assign	99
6.48.2.3	extfh_ptr	99
6.48.2.4	file	99
6.48.2.5	file_status	100

---

6.48.2.6	file_version	100
6.48.2.7	flag_begin_of_file	100
6.48.2.8	flag_end_of_file	100
6.48.2.9	flag_first_read	100
6.48.2.10	flag_needs_nl	100
6.48.2.11	flag_needs_top	100
6.48.2.12	flag_nonexistent	100
6.48.2.13	flag_optional	100
6.48.2.14	flag_read_done	100
6.48.2.15	flag_select_features	101
6.48.2.16	keys	101
6.48.2.17	last_open_mode	101
6.48.2.18	linorkeyptr	101
6.48.2.19	lock_mode	101
6.48.2.20	nkeys	101
6.48.2.21	open_mode	101
6.48.2.22	organization	101
6.48.2.23	record	101
6.48.2.24	record_max	101
6.48.2.25	record_min	102
6.48.2.26	record_size	102
6.48.2.27	select_name	102
6.48.2.28	sort_collating	102
6.48.2.29	special	102
6.49	cob_file_key Struct Reference	102
6.49.1	Detailed Description	103
6.49.2	Member Data Documentation	103
6.49.2.1	field	103
6.49.2.2	flag	103
6.49.2.3	offset	104
6.50	cob_fileio_funcs Struct Reference	104
6.50.1	Detailed Description	104
6.50.2	Member Data Documentation	104
6.50.2.1	close	104

6.50.2.2	fdelete	104
6.50.2.3	open	104
6.50.2.4	read	105
6.50.2.5	read_next	105
6.50.2.6	rewrite	105
6.50.2.7	start	105
6.50.2.8	write	105
6.51	cob_inp_struct Struct Reference	106
6.51.1	Detailed Description	107
6.51.2	Member Data Documentation	107
6.51.2.1	down_index	107
6.51.2.2	scr	107
6.51.2.3	this_x	107
6.51.2.4	this_y	107
6.51.2.5	up_index	107
6.52	cob_module Struct Reference	107
6.52.1	Detailed Description	108
6.52.2	Member Data Documentation	109
6.52.2.1	cob_procedure_parameters	109
6.52.2.2	collating_sequence	109
6.52.2.3	crt_status	109
6.52.2.4	currency_symbol	109
6.52.2.5	cursor_pos	109
6.52.2.6	decimal_point	109
6.52.2.7	display_sign	109
6.52.2.8	flag_binary_truncate	109
6.52.2.9	flag_filename_mapping	109
6.52.2.10	flag_pretty_display	109
6.52.2.11	next	110
6.52.2.12	numeric_separator	110
6.52.2.13	spare8	110
6.53	cobitem Struct Reference	110
6.53.1	Detailed Description	110
6.53.2	Member Data Documentation	111

---

6.53.2.1	block_byte	111
6.53.2.2	end_of_block	111
6.53.2.3	item	111
6.53.2.4	next	111
6.53.2.5	unique	111
6.54	cobjmp_buf Struct Reference	111
6.54.1	Detailed Description	111
6.54.2	Member Data Documentation	112
6.54.2.1	cbj_int	112
6.54.2.2	cbj_jmp_buf	112
6.54.2.3	cbj_ptr	112
6.54.2.4	cbj_ptr_rest	112
6.55	cobsort Struct Reference	113
6.55.1	Detailed Description	114
6.55.2	Member Data Documentation	114
6.55.2.1	destination_file	114
6.55.2.2	empty	114
6.55.2.3	file	114
6.55.2.4	files_used	114
6.55.2.5	fnstatus	114
6.55.2.6	memory	114
6.55.2.7	pointer	114
6.55.2.8	queue	114
6.55.2.9	r_size	114
6.55.2.10	retrieval_queue	115
6.55.2.11	retrieving	115
6.55.2.12	size	115
6.55.2.13	sort_return	115
6.55.2.14	unique	115
6.55.2.15	w_size	115
6.56	d1m_struct Struct Reference	116
6.56.1	Detailed Description	116
6.56.2	Member Data Documentation	116
6.56.2.1	uns_all	116

---

6.56.2.2	uns_dlm	116
6.57	expr_node Struct Reference	117
6.57.1	Detailed Description	117
6.57.2	Member Data Documentation	117
6.57.2.1	token	117
6.57.2.2	value	117
6.58	file_struct Struct Reference	118
6.58.1	Detailed Description	118
6.58.2	Member Data Documentation	118
6.58.2.1	count	118
6.58.2.2	fp	118
6.59	filename Struct Reference	118
6.59.1	Detailed Description	119
6.59.2	Member Data Documentation	119
6.59.2.1	demangle_source	119
6.59.2.2	localfile	119
6.59.2.3	need_assemble	119
6.59.2.4	need_preprocess	119
6.59.2.5	need_translate	119
6.59.2.6	next	119
6.59.2.7	object	120
6.59.2.8	preprocess	120
6.59.2.9	source	120
6.59.2.10	translate	120
6.59.2.11	trstorage	120
6.60	handler_struct Struct Reference	120
6.60.1	Detailed Description	122
6.60.2	Member Data Documentation	122
6.60.2.1	handler_label	122
6.60.2.2	handler_prog	122
6.61	indexed_file Struct Reference	122
6.61.1	Detailed Description	123
6.61.2	Member Data Documentation	123
6.61.2.1	bdb_file_lock	123

---

6.61.2.2	<a href="#">bdb_lock_id</a>	123
6.61.2.3	<a href="#">bdb_record_lock</a>	123
6.61.2.4	<a href="#">cursor</a>	123
6.61.2.5	<a href="#">data</a>	123
6.61.2.6	<a href="#">db</a>	123
6.61.2.7	<a href="#">filename</a>	123
6.61.2.8	<a href="#">filenamelen</a>	123
6.61.2.9	<a href="#">key</a>	123
6.61.2.10	<a href="#">key_index</a>	124
6.61.2.11	<a href="#">last_dupno</a>	124
6.61.2.12	<a href="#">last_key</a>	124
6.61.2.13	<a href="#">last_readkey</a>	124
6.61.2.14	<a href="#">record_locked</a>	124
6.61.2.15	<a href="#">rewrite_sec_key</a>	124
6.61.2.16	<a href="#">temp_key</a>	124
6.61.2.17	<a href="#">write_cursor_open</a>	124
6.62	<a href="#">linage_struct Struct Reference</a>	124
6.62.1	<a href="#">Detailed Description</a>	125
6.62.2	<a href="#">Member Data Documentation</a>	126
6.62.2.1	<a href="#">latbot</a>	126
6.62.2.2	<a href="#">latfoot</a>	126
6.62.2.3	<a href="#">lattop</a>	126
6.62.2.4	<a href="#">lin_bot</a>	126
6.62.2.5	<a href="#">lin_foot</a>	126
6.62.2.6	<a href="#">lin_lines</a>	126
6.62.2.7	<a href="#">lin_top</a>	126
6.62.2.8	<a href="#">linage</a>	126
6.62.2.9	<a href="#">linage_ctr</a>	126
6.63	<a href="#">local_filename Struct Reference</a>	127
6.63.1	<a href="#">Detailed Description</a>	127
6.63.2	<a href="#">Member Data Documentation</a>	127
6.63.2.1	<a href="#">local_fp</a>	127
6.63.2.2	<a href="#">local_name</a>	127
6.63.2.3	<a href="#">next</a>	127

6.64	memory_struct Struct Reference	128
6.64.1	Detailed Description	128
6.64.2	Member Data Documentation	128
6.64.2.1	count	128
6.64.2.2	first	128
6.64.2.3	last	128
6.65	noreserve Struct Reference	129
6.65.1	Detailed Description	129
6.65.2	Member Data Documentation	129
6.65.2.1	next	129
6.65.2.2	noresword	129
6.66	option Struct Reference	130
6.66.1	Detailed Description	130
6.66.2	Member Data Documentation	130
6.66.2.1	flag	130
6.66.2.2	has_arg	130
6.66.2.3	name	130
6.66.2.4	val	130
6.67	reserved Struct Reference	130
6.67.1	Detailed Description	131
6.67.2	Member Data Documentation	131
6.67.2.1	name	131
6.67.2.2	token	131
6.68	sort_list Struct Reference	131
6.68.1	Detailed Description	131
6.68.2	Member Data Documentation	131
6.68.2.1	next	132
6.69	struct_handle Struct Reference	132
6.69.1	Detailed Description	132
6.69.2	Member Data Documentation	132
6.69.2.1	next	132
6.69.2.2	preload_handle	132
6.70	system_table Struct Reference	133
6.70.1	Detailed Description	133



---

6.70.2	Member Data Documentation	133
6.70.2.1	syst_call	133
6.70.2.2	syst_call	133
6.70.2.3	syst_name	133
6.70.2.4	syst_params	133
6.71	yy_buffer_state Struct Reference	133
6.71.1	Detailed Description	134
6.71.2	Member Data Documentation	134
6.71.2.1	yy_at_bol	134
6.71.2.2	yy_buf_pos	134
6.71.2.3	yy_buf_size	134
6.71.2.4	yy_buffer_status	134
6.71.2.5	yy_ch_buf	134
6.71.2.6	yy_fill_buffer	134
6.71.2.7	yy_input_file	134
6.71.2.8	yy_is_interactive	135
6.71.2.9	yy_is_our_buffer	135
6.71.2.10	yy_n_chars	135
6.72	yyalloc Union Reference	135
6.72.1	Detailed Description	135
6.72.2	Member Data Documentation	135
6.72.2.1	yyss	135
6.72.2.2	yyvs	135
6.73	YYSTYPE Union Reference	136
6.73.1	Detailed Description	136
6.73.2	Member Data Documentation	136
6.73.2.1	l	136
6.73.2.2	r	137
6.73.2.3	s	137
<b>7</b>	<b>File Documentation</b>	<b>139</b>
7.1	bin/cobcrun.c File Reference	139
7.1.1	Function Documentation	139
7.1.1.1	main	140

7.2	cobc/cobc.c File Reference	140
7.2.1	Define Documentation	143
7.2.1.1	CB_FLAG	143
7.2.1.2	CB_FLAG	143
7.2.1.3	CB_FLAG	143
7.2.1.4	CB_WARNDEF	143
7.2.1.5	CB_WARNDEF	143
7.2.1.6	CB_WARNDEF	144
7.2.1.7	CB_WARNDEF	144
7.2.1.8	CB_WARNDEF	144
7.2.1.9	CB_WARNDEF	144
7.2.1.10	COB_EXCEPTION	144
7.2.1.11	COB_NUM_CSYSNS	144
7.2.1.12	PATHSEPS	144
7.2.2	Typedef Documentation	144
7.2.2.1	cob_sighandler_t	144
7.2.3	Enumeration Type Documentation	144
7.2.3.1	cb_compile_level	145
7.2.4	Function Documentation	145
7.2.4.1	cb_text_list_add	145
7.2.4.2	cobc_abort	145
7.2.4.3	cobc_check_valid_name	146
7.2.4.4	cobc_malloc	146
7.2.4.5	cobc_realloc	146
7.2.4.6	cobc_tree_cast_error	147
7.2.4.7	main	147
7.2.5	Variable Documentation	151
7.2.5.1	alt_ebcdic	151
7.2.5.2	cb_attr_id	152
7.2.5.3	cb_depend_file	152
7.2.5.4	cb_depend_list	152
7.2.5.5	cb_depend_target	152
7.2.5.6	cb_display_sign	152
7.2.5.7	cb_exception_table	152

---

7.2.5.8	<code>cb_extension_list</code>	152
7.2.5.9	<code>cb_field_id</code>	152
7.2.5.10	<code>cb_flag_main</code>	152
7.2.5.11	<code>cb_id</code>	152
7.2.5.12	<code>cb_include_list</code>	153
7.2.5.13	<code>cb_listing_file</code>	153
7.2.5.14	<code>cb_literal_id</code>	153
7.2.5.15	<code>cb_oc_build_stamp</code>	153
7.2.5.16	<code>cb_saveargc</code>	153
7.2.5.17	<code>cb_saveargv</code>	153
7.2.5.18	<code>cb_source_file</code>	153
7.2.5.19	<code>cb_source_format</code>	153
7.2.5.20	<code>cb_source_line</code>	153
7.2.5.21	<code>cb_storage_file</code>	153
7.2.5.22	<code>cb_storage_file_name</code>	154
7.2.5.23	<code>cb_storage_id</code>	154
7.2.5.24	<code>cob_config_dir</code>	154
7.2.5.25	<code>demangle_name</code>	154
7.2.5.26	<code>errorcount</code>	154
7.2.5.27	<code>optimize_flag</code>	154
7.2.5.28	<code>source_name</code>	154
7.2.5.29	<code>warningcount</code>	154
7.3	<code>cobc/cobc.h</code> File Reference	154
7.3.1	Define Documentation	157
7.3.1.1	<code>ABORT</code>	157
7.3.1.2	<code>CB_CONFIG_ANY</code>	157
7.3.1.3	<code>CB_CONFIG_BOOLEAN</code>	158
7.3.1.4	<code>CB_CONFIG_INT</code>	158
7.3.1.5	<code>CB_CONFIG_STRING</code>	158
7.3.1.6	<code>CB_CONFIG_SUPPORT</code>	158
7.3.1.7	<code>CB_EXCEPTION_CODE</code>	158
7.3.1.8	<code>CB_EXCEPTION_ENABLE</code>	158
7.3.1.9	<code>CB_EXCEPTION_NAME</code>	158
7.3.1.10	<code>CB_FLAG</code>	158

7.3.1.11	CB_FORMAT_FIXED	158
7.3.1.12	CB_FORMAT_FREE	158
7.3.1.13	CB_WARNDEF	159
7.3.1.14	COB_NON_ALIGNED	159
7.3.2	Enumeration Type Documentation	159
7.3.2.1	cb_assign_clause	159
7.3.2.2	cb_binary_byteorder	159
7.3.2.3	cb_binary_size	159
7.3.2.4	cb_operation_type	160
7.3.2.5	cb_support	160
7.3.3	Function Documentation	161
7.3.3.1	cb_error	161
7.3.3.2	cb_load_conf	161
7.3.3.3	cb_load_std	165
7.3.3.4	cb_text_list_add	165
7.3.3.5	cb_verify	165
7.3.3.6	cb_warning	166
7.3.3.7	cobc_abort	166
7.3.3.8	cobc_check_valid_name	166
7.3.3.9	cobc_malloc	167
7.3.3.10	cobc_realloc	167
7.3.3.11	pp_set_replace_list	167
7.3.3.12	ppcopy	167
7.3.3.13	plex	167
7.3.3.14	ppopen	167
7.3.3.15	ppparse	167
7.3.3.16	yylex	167
7.3.3.17	yyparse	167
7.3.4	Variable Documentation	167
7.3.4.1	alt_ebcdic	168
7.3.4.2	cb_attr_id	168
7.3.4.3	cb_depend_file	168
7.3.4.4	cb_depend_list	168
7.3.4.5	cb_depend_target	168

---

7.3.4.6	<a href="#">cb_display_sign</a>	168
7.3.4.7	<a href="#">cb_exception_table</a>	168
7.3.4.8	<a href="#">cb_extension_list</a>	168
7.3.4.9	<a href="#">cb_field_id</a>	168
7.3.4.10	<a href="#">cb_flag_main</a>	168
7.3.4.11	<a href="#">cb_id</a>	168
7.3.4.12	<a href="#">cb_include_list</a>	169
7.3.4.13	<a href="#">cb_listing_file</a>	169
7.3.4.14	<a href="#">cb_literal_id</a>	169
7.3.4.15	<a href="#">cb_oc_build_stamp</a>	169
7.3.4.16	<a href="#">cb_saveargc</a>	169
7.3.4.17	<a href="#">cb_saveargv</a>	169
7.3.4.18	<a href="#">cb_source_file</a>	169
7.3.4.19	<a href="#">cb_source_format</a>	169
7.3.4.20	<a href="#">cb_source_line</a>	169
7.3.4.21	<a href="#">cb_storage_file</a>	169
7.3.4.22	<a href="#">cb_storage_file_name</a>	170
7.3.4.23	<a href="#">cb_storage_id</a>	170
7.3.4.24	<a href="#">cob_config_dir</a>	170
7.3.4.25	<a href="#">current_paragraph</a>	170
7.3.4.26	<a href="#">current_program</a>	170
7.3.4.27	<a href="#">current_section</a>	170
7.3.4.28	<a href="#">current_statement</a>	170
7.3.4.29	<a href="#">demangle_name</a>	170
7.3.4.30	<a href="#">errorcount</a>	170
7.3.4.31	<a href="#">functions_are_all</a>	170
7.3.4.32	<a href="#">has_external</a>	171
7.3.4.33	<a href="#">norestab</a>	171
7.3.4.34	<a href="#">optimize_flag</a>	171
7.3.4.35	<a href="#">ppin</a>	171
7.3.4.36	<a href="#">ppout</a>	171
7.3.4.37	<a href="#">sending_id</a>	171
7.3.4.38	<a href="#">source_name</a>	171
7.3.4.39	<a href="#">suppress_warn</a>	171

7.3.4.40	warningcount . . . . .	171
7.3.4.41	yyin . . . . .	171
7.3.4.42	yyout . . . . .	171
7.4	cobc/codegen.c File Reference . . . . .	171
7.4.1	Define Documentation . . . . .	173
7.4.1.1	COB_MAX_SUBSCRIPTS . . . . .	173
7.4.1.2	COB_SYSTEM_GEN . . . . .	173
7.4.1.3	COB_USE_SETJMP . . . . .	173
7.4.1.4	INITIALIZE_COMPOUND . . . . .	173
7.4.1.5	INITIALIZE_DEFAULT . . . . .	173
7.4.1.6	INITIALIZE_EXTERNAL . . . . .	173
7.4.1.7	INITIALIZE_NONE . . . . .	173
7.4.1.8	INITIALIZE_ONE . . . . .	173
7.4.2	Function Documentation . . . . .	174
7.4.2.1	codegen . . . . .	174
7.4.3	Variable Documentation . . . . .	184
7.4.3.1	has_external . . . . .	184
7.5	cobc/config.c File Reference . . . . .	184
7.5.1	Define Documentation . . . . .	185
7.5.1.1	CB_CONFIG_ANY . . . . .	185
7.5.1.2	CB_CONFIG_ANY . . . . .	185
7.5.1.3	CB_CONFIG_BOOLEAN . . . . .	185
7.5.1.4	CB_CONFIG_BOOLEAN . . . . .	185
7.5.1.5	CB_CONFIG_INT . . . . .	185
7.5.1.6	CB_CONFIG_INT . . . . .	185
7.5.1.7	CB_CONFIG_STRING . . . . .	186
7.5.1.8	CB_CONFIG_STRING . . . . .	186
7.5.1.9	CB_CONFIG_SUPPORT . . . . .	186
7.5.1.10	CB_CONFIG_SUPPORT . . . . .	186
7.5.2	Enumeration Type Documentation . . . . .	186
7.5.2.1	cb_config_type . . . . .	186
7.5.3	Function Documentation . . . . .	186
7.5.3.1	cb_load_conf . . . . .	186
7.5.3.2	cb_load_std . . . . .	190

---

7.5.4	Variable Documentation	190
7.5.4.1	name	190
7.5.4.2	norestab	190
7.5.4.3	type	191
7.5.4.4	val	191
7.5.4.5	var	191
7.6	cobc/error.c File Reference	191
7.6.1	Function Documentation	192
7.6.1.1	ambiguous_error	192
7.6.1.2	cb_error	193
7.6.1.3	cb_error_x	193
7.6.1.4	cb_verify	193
7.6.1.5	cb_warning	194
7.6.1.6	cb_warning_x	194
7.6.1.7	check_filler_name	195
7.6.1.8	group_error	195
7.6.1.9	level_except_error	195
7.6.1.10	level_redundant_error	195
7.6.1.11	level_require_error	196
7.6.1.12	redefinition_error	196
7.6.1.13	redefinition_warning	196
7.6.1.14	undefined_error	196
7.7	cobc/field.c File Reference	197
7.7.1	Function Documentation	198
7.7.1.1	cb_build_field_tree	198
7.7.1.2	cb_clear_real_field	200
7.7.1.3	cb_get_level	200
7.7.1.4	cb_resolve_redefines	201
7.7.1.5	cb_validate_78_item	202
7.7.1.6	cb_validate_88_item	203
7.7.1.7	cb_validate_field	203
7.7.2	Variable Documentation	204
7.7.2.1	cb_needs_01	204
7.8	cobc/parser.c File Reference	204

---

7.8.1	Define Documentation	224
7.8.1.1	ACCEPT	224
7.8.1.2	ACCESS	224
7.8.1.3	ADD	224
7.8.1.4	ADDRESS	224
7.8.1.5	ADVANCING	224
7.8.1.6	AFTER	224
7.8.1.7	ALL	224
7.8.1.8	ALLOCATE	224
7.8.1.9	ALPHABET	224
7.8.1.10	ALPHABETIC	225
7.8.1.11	ALPHABETIC_LOWER	225
7.8.1.12	ALPHABETIC_UPPER	225
7.8.1.13	ALPHANUMERIC	225
7.8.1.14	ALPHANUMERIC_EDITED	225
7.8.1.15	ALSO	225
7.8.1.16	ALTER	225
7.8.1.17	ALTERNATE	225
7.8.1.18	AND	225
7.8.1.19	ANY	225
7.8.1.20	ARE	226
7.8.1.21	AREA	226
7.8.1.22	ARGUMENT_NUMBER	226
7.8.1.23	ARGUMENT_VALUE	226
7.8.1.24	AS	226
7.8.1.25	ASCENDING	226
7.8.1.26	ASSIGN	226
7.8.1.27	AT	226
7.8.1.28	AUTO	226
7.8.1.29	AUTOMATIC	226
7.8.1.30	BACKGROUND_COLOR	227
7.8.1.31	BASED	227
7.8.1.32	BEFORE	227
7.8.1.33	BELL	227



---

7.8.1.34	BINARY	227
7.8.1.35	BINARY_C_LONG	227
7.8.1.36	BINARY_CHAR	227
7.8.1.37	BINARY_DOUBLE	227
7.8.1.38	BINARY_LONG	227
7.8.1.39	BINARY_SHORT	227
7.8.1.40	BLANK	228
7.8.1.41	BLANK_LINE	228
7.8.1.42	BLANK_SCREEN	228
7.8.1.43	BLINK	228
7.8.1.44	BLOCK	228
7.8.1.45	BOTTOM	228
7.8.1.46	BY	228
7.8.1.47	BYTE_LENGTH	228
7.8.1.48	CALL	228
7.8.1.49	CANCEL	228
7.8.1.50	CH	229
7.8.1.51	CHAINING	229
7.8.1.52	CHARACTER	229
7.8.1.53	CHARACTERS	229
7.8.1.54	CLASS	229
7.8.1.55	CLOSE	229
7.8.1.56	CODE	229
7.8.1.57	CODE_SET	229
7.8.1.58	COL	229
7.8.1.59	COLLATING	229
7.8.1.60	COLS	230
7.8.1.61	COLUMN	230
7.8.1.62	COLUMNS	230
7.8.1.63	COMMA	230
7.8.1.64	COMMA_DELIM	230
7.8.1.65	COMMAND_LINE	230
7.8.1.66	COMMIT	230
7.8.1.67	COMMON	230

---

7.8.1.68	COMP	230
7.8.1.69	COMP_1	230
7.8.1.70	COMP_2	231
7.8.1.71	COMP_3	231
7.8.1.72	COMP_4	231
7.8.1.73	COMP_5	231
7.8.1.74	COMP_X	231
7.8.1.75	COMPUTE	231
7.8.1.76	CONCATENATE_FUNC	231
7.8.1.77	CONFIGURATION	231
7.8.1.78	CONSTANT	231
7.8.1.79	CONTAINS	231
7.8.1.80	CONTENT	232
7.8.1.81	CONTINUE	232
7.8.1.82	CONTROL	232
7.8.1.83	CONTROL_FOOTING	232
7.8.1.84	CONTROL_HEADING	232
7.8.1.85	CONTROLS	232
7.8.1.86	CONVERTING	232
7.8.1.87	CORRESPONDING	232
7.8.1.88	COUNT	232
7.8.1.89	CRT	232
7.8.1.90	CURRENCY	233
7.8.1.91	CURRENT_DATE_FUNC	233
7.8.1.92	CURSOR	233
7.8.1.93	CYCLE	233
7.8.1.94	DATA	233
7.8.1.95	DATE	233
7.8.1.96	DAY	233
7.8.1.97	DAY_OF_WEEK	233
7.8.1.98	DE	233
7.8.1.99	DEBUGGING	233
7.8.1.100	DECIMAL_POINT	234
7.8.1.101	DECLARATIVES	234

---

7.8.1.102 DEFAULT . . . . .	234
7.8.1.103 DELETE . . . . .	234
7.8.1.104 DELIMITED . . . . .	234
7.8.1.105 DELIMITER . . . . .	234
7.8.1.106 DEPENDING . . . . .	234
7.8.1.107 DESCENDING . . . . .	234
7.8.1.108 DETAIL . . . . .	234
7.8.1.109 DISK . . . . .	234
7.8.1.110 DISPLAY . . . . .	235
7.8.1.111 DIVIDE . . . . .	235
7.8.1.112 DIVISION . . . . .	235
7.8.1.113 DOWN . . . . .	235
7.8.1.114 DUPLICATES . . . . .	235
7.8.1.115 DYNAMIC . . . . .	235
7.8.1.116 EBCDIC . . . . .	235
7.8.1.117 ELSE . . . . .	235
7.8.1.118 emit_statement . . . . .	235
7.8.1.119 END . . . . .	235
7.8.1.120 END_ACCEPT . . . . .	236
7.8.1.121 END_ADD . . . . .	236
7.8.1.122 END_CALL . . . . .	236
7.8.1.123 END_COMPUTE . . . . .	236
7.8.1.124 END_DELETE . . . . .	236
7.8.1.125 END_DISPLAY . . . . .	236
7.8.1.126 END_DIVIDE . . . . .	236
7.8.1.127 END_EVALUATE . . . . .	236
7.8.1.128 END_FUNCTION . . . . .	236
7.8.1.129 END_IF . . . . .	236
7.8.1.130 END_MULTIPLY . . . . .	237
7.8.1.131 END_PERFORM . . . . .	237
7.8.1.132 END_PROGRAM . . . . .	237
7.8.1.133 END_READ . . . . .	237
7.8.1.134 END_RETURN . . . . .	237
7.8.1.135 END_REWRITE . . . . .	237

---

7.8.1.136 END_SEARCH . . . . .	237
7.8.1.137 END_START . . . . .	237
7.8.1.138 END_STRING . . . . .	237
7.8.1.139 END_SUBTRACT . . . . .	237
7.8.1.140 END_UNSTRING . . . . .	238
7.8.1.141 END_WRITE . . . . .	238
7.8.1.142 ENTRY . . . . .	238
7.8.1.143 ENVIRONMENT . . . . .	238
7.8.1.144 ENVIRONMENT_NAME . . . . .	238
7.8.1.145 ENVIRONMENT_VALUE . . . . .	238
7.8.1.146 EOL . . . . .	238
7.8.1.147 EOP . . . . .	238
7.8.1.148 EOS . . . . .	238
7.8.1.149 EQUAL . . . . .	238
7.8.1.150 EQUALS . . . . .	239
7.8.1.151 ERASE . . . . .	239
7.8.1.152 ERROR . . . . .	239
7.8.1.153 ESCAPE . . . . .	239
7.8.1.154 EVALUATE . . . . .	239
7.8.1.155 EVENT_STATUS . . . . .	239
7.8.1.156 EXCEPTION . . . . .	239
7.8.1.157 EXCLUSIVE . . . . .	239
7.8.1.158 EXIT . . . . .	239
7.8.1.159 EXTEND . . . . .	239
7.8.1.160 EXTERNAL . . . . .	240
7.8.1.161 FD . . . . .	240
7.8.1.162 FILE_CONTROL . . . . .	240
7.8.1.163 FILE_ID . . . . .	240
7.8.1.164 FILLER . . . . .	240
7.8.1.165 FINAL . . . . .	240
7.8.1.166 FIRST . . . . .	240
7.8.1.167 FOOTING . . . . .	240
7.8.1.168 FOR . . . . .	240
7.8.1.169 FOREGROUND_COLOR . . . . .	240

---

7.8.1.170 FOREVER . . . . .	241
7.8.1.171 FREE . . . . .	241
7.8.1.172 FROM . . . . .	241
7.8.1.173 FULL . . . . .	241
7.8.1.174 FUNCTION . . . . .	241
7.8.1.175 FUNCTION_ID . . . . .	241
7.8.1.176 FUNCTION_NAME . . . . .	241
7.8.1.177 GE . . . . .	241
7.8.1.178 GENERATE . . . . .	241
7.8.1.179 GIVING . . . . .	241
7.8.1.180 GLOBAL . . . . .	242
7.8.1.181 GO . . . . .	242
7.8.1.182 GOBACK . . . . .	242
7.8.1.183 GREATER . . . . .	242
7.8.1.184 GROUP . . . . .	242
7.8.1.185 HEADING . . . . .	242
7.8.1.186 HIGH_VALUE . . . . .	242
7.8.1.187 HIGHLIGHT . . . . .	242
7.8.1.188 I_O . . . . .	242
7.8.1.189 I_O_CONTROL . . . . .	242
7.8.1.190 IDENTIFICATION . . . . .	243
7.8.1.191 IF . . . . .	243
7.8.1.192 IGNORE . . . . .	243
7.8.1.193 IGNORING . . . . .	243
7.8.1.194 IN . . . . .	243
7.8.1.195 INDEX . . . . .	243
7.8.1.196 INDEXED . . . . .	243
7.8.1.197 INDICATE . . . . .	243
7.8.1.198 INITIALIZE . . . . .	243
7.8.1.199 INITIALIZED . . . . .	243
7.8.1.200 INITIATE . . . . .	244
7.8.1.201 INPUT . . . . .	244
7.8.1.202 INPUT_OUTPUT . . . . .	244
7.8.1.203 INSPECT . . . . .	244

---

7.8.1.204 INTO . . . . .	244
7.8.1.205 INTRINSIC . . . . .	244
7.8.1.206 INVALID . . . . .	244
7.8.1.207 INVALID_KEY . . . . .	244
7.8.1.208 IS . . . . .	244
7.8.1.209 JUSTIFIED . . . . .	244
7.8.1.210 KEY . . . . .	245
7.8.1.211 LABEL . . . . .	245
7.8.1.212 LAST . . . . .	245
7.8.1.213 LAST_DETAIL . . . . .	245
7.8.1.214 LE . . . . .	245
7.8.1.215 LEADING . . . . .	245
7.8.1.216 LEFT . . . . .	245
7.8.1.217 LENGTH . . . . .	245
7.8.1.218 LESS . . . . .	245
7.8.1.219 LIMIT . . . . .	245
7.8.1.220 LIMITS . . . . .	246
7.8.1.221 LINAGE . . . . .	246
7.8.1.222 LINAGE_COUNTER . . . . .	246
7.8.1.223 LINE . . . . .	246
7.8.1.224 LINES . . . . .	246
7.8.1.225 LINKAGE . . . . .	246
7.8.1.226 LITERAL . . . . .	246
7.8.1.227 LOCAL_STORAGE . . . . .	246
7.8.1.228 LOCALE . . . . .	246
7.8.1.229 LOCALE_DT_FUNC . . . . .	246
7.8.1.230 LOCK . . . . .	247
7.8.1.231 LOW_VALUE . . . . .	247
7.8.1.232 LOWER_CASE_FUNC . . . . .	247
7.8.1.233 LOWLIGHT . . . . .	247
7.8.1.234 MANUAL . . . . .	247
7.8.1.235 MEMORY . . . . .	247
7.8.1.236 MERGE . . . . .	247
7.8.1.237 MINUS . . . . .	247

---

7.8.1.238 MNEMONIC_NAME . . . . .	247
7.8.1.239 MODE . . . . .	247
7.8.1.240 MOVE . . . . .	248
7.8.1.241 MULTIPLE . . . . .	248
7.8.1.242 MULTIPLY . . . . .	248
7.8.1.243 NATIONAL . . . . .	248
7.8.1.244 NATIONAL_EDITED . . . . .	248
7.8.1.245 NATIVE . . . . .	248
7.8.1.246 NE . . . . .	248
7.8.1.247 NEGATIVE . . . . .	248
7.8.1.248 NEXT . . . . .	248
7.8.1.249 NEXT_SENTENCE . . . . .	248
7.8.1.250 NO . . . . .	249
7.8.1.251 NO_ADVANCING . . . . .	249
7.8.1.252 NOT . . . . .	249
7.8.1.253 NOT_END . . . . .	249
7.8.1.254 NOT_EOP . . . . .	249
7.8.1.255 NOT_EXCEPTION . . . . .	249
7.8.1.256 NOT_INVALID_KEY . . . . .	249
7.8.1.257 NOT_OVERFLOW . . . . .	249
7.8.1.258 NOT_SIZE_ERROR . . . . .	249
7.8.1.259 NUMBER . . . . .	249
7.8.1.260 NUMBERS . . . . .	250
7.8.1.261 NUMERIC . . . . .	250
7.8.1.262 NUMERIC_EDITED . . . . .	250
7.8.1.263 NUMVALC_FUNC . . . . .	250
7.8.1.264 OBJECT_COMPUTER . . . . .	250
7.8.1.265 OCCURS . . . . .	250
7.8.1.266 OF . . . . .	250
7.8.1.267 OFF . . . . .	250
7.8.1.268 OMITTED . . . . .	250
7.8.1.269 ON . . . . .	250
7.8.1.270 ONLY . . . . .	251
7.8.1.271 OPEN . . . . .	251

---

7.8.1.272 OPTIONAL . . . . .	251
7.8.1.273 OR . . . . .	251
7.8.1.274 ORDER . . . . .	251
7.8.1.275 ORGANIZATION . . . . .	251
7.8.1.276 OTHER . . . . .	251
7.8.1.277 OUTPUT . . . . .	251
7.8.1.278 OVERFLOW . . . . .	251
7.8.1.279 OVERLINE . . . . .	251
7.8.1.280 PACKED_DECIMAL . . . . .	252
7.8.1.281 PADDING . . . . .	252
7.8.1.282 PAGE . . . . .	252
7.8.1.283 PAGE_FOOTING . . . . .	252
7.8.1.284 PAGE_HEADING . . . . .	252
7.8.1.285 PARAGRAPH . . . . .	252
7.8.1.286 PENDING . . . . .	252
7.8.1.287 PERFORM . . . . .	252
7.8.1.288 PICTURE . . . . .	252
7.8.1.289 PLUS . . . . .	252
7.8.1.290 POINTER . . . . .	253
7.8.1.291 POSITION . . . . .	253
7.8.1.292 POSITIVE . . . . .	253
7.8.1.293 PRESENT . . . . .	253
7.8.1.294 PREVIOUS . . . . .	253
7.8.1.295 PRINTER . . . . .	253
7.8.1.296 PRINTING . . . . .	253
7.8.1.297 PROCEDURE . . . . .	253
7.8.1.298 PROCEDURES . . . . .	253
7.8.1.299 PROCEED . . . . .	253
7.8.1.300 PROGRAM . . . . .	254
7.8.1.301 PROGRAM_ID . . . . .	254
7.8.1.302 PROGRAM_NAME . . . . .	254
7.8.1.303 PROGRAM_POINTER . . . . .	254
7.8.1.304 PROMPT . . . . .	254
7.8.1.305 push_expr . . . . .	254



---

7.8.1.306 QUOTE . . . . .	254
7.8.1.307 RANDOM . . . . .	254
7.8.1.308 RD . . . . .	254
7.8.1.309 READ . . . . .	254
7.8.1.310 RECORD . . . . .	255
7.8.1.311 RECORDING . . . . .	255
7.8.1.312 RECORDS . . . . .	255
7.8.1.313 RECURSIVE . . . . .	255
7.8.1.314 REDEFINES . . . . .	255
7.8.1.315 REEL . . . . .	255
7.8.1.316 REFERENCE . . . . .	255
7.8.1.317 RELATIVE . . . . .	255
7.8.1.318 RELEASE . . . . .	255
7.8.1.319 REMAINDER . . . . .	255
7.8.1.320 REMOVAL . . . . .	256
7.8.1.321 RENAMES . . . . .	256
7.8.1.322 REPLACING . . . . .	256
7.8.1.323 REPORT . . . . .	256
7.8.1.324 REPORT_FOOTING . . . . .	256
7.8.1.325 REPORT_HEADING . . . . .	256
7.8.1.326 REPORTING . . . . .	256
7.8.1.327 REPORTS . . . . .	256
7.8.1.328 REPOSITORY . . . . .	256
7.8.1.329 REQUIRED . . . . .	256
7.8.1.330 RESERVE . . . . .	257
7.8.1.331 RETURN . . . . .	257
7.8.1.332 RETURNING . . . . .	257
7.8.1.333 REVERSE_FUNC . . . . .	257
7.8.1.334 REVERSE_VIDEO . . . . .	257
7.8.1.335 REWIND . . . . .	257
7.8.1.336 REWRITE . . . . .	257
7.8.1.337 RIGHT . . . . .	257
7.8.1.338 ROLLBACK . . . . .	257
7.8.1.339 ROUNDED . . . . .	257

---

7.8.1.340 RUN . . . . .	258
7.8.1.341 SAME . . . . .	258
7.8.1.342 SCREEN . . . . .	258
7.8.1.343 SCREEN_CONTROL . . . . .	258
7.8.1.344 SCROLL . . . . .	258
7.8.1.345 SD . . . . .	258
7.8.1.346 SEARCH . . . . .	258
7.8.1.347 SECTION . . . . .	258
7.8.1.348 SECURE . . . . .	258
7.8.1.349 SEGMENT_LIMIT . . . . .	258
7.8.1.350 SELECT . . . . .	259
7.8.1.351 SEMI_COLON . . . . .	259
7.8.1.352 SENTENCE . . . . .	259
7.8.1.353 SEPARATE . . . . .	259
7.8.1.354 SEQUENCE . . . . .	259
7.8.1.355 SEQUENTIAL . . . . .	259
7.8.1.356 SET . . . . .	259
7.8.1.357 SHARING . . . . .	259
7.8.1.358 SIGN . . . . .	259
7.8.1.359 SIGNED . . . . .	259
7.8.1.360 SIGNED_INT . . . . .	260
7.8.1.361 SIGNED_LONG . . . . .	260
7.8.1.362 SIGNED_SHORT . . . . .	260
7.8.1.363 SIZE . . . . .	260
7.8.1.364 SIZE_ERROR . . . . .	260
7.8.1.365 SORT . . . . .	260
7.8.1.366 SORT_MERGE . . . . .	260
7.8.1.367 SOURCE . . . . .	260
7.8.1.368 SOURCE_COMPUTER . . . . .	260
7.8.1.369 SPACE . . . . .	260
7.8.1.370 SPECIAL_NAMES . . . . .	261
7.8.1.371 STANDARD . . . . .	261
7.8.1.372 STANDARD_1 . . . . .	261
7.8.1.373 STANDARD_2 . . . . .	261

---

7.8.1.374 START . . . . .	261
7.8.1.375 STATUS . . . . .	261
7.8.1.376 STOP . . . . .	261
7.8.1.377 STRING . . . . .	261
7.8.1.378 SUBSTITUTE_CASE_FUNC . . . . .	261
7.8.1.379 SUBSTITUTE_FUNC . . . . .	261
7.8.1.380 SUBTRACT . . . . .	262
7.8.1.381 SUM . . . . .	262
7.8.1.382 SUPPRESS . . . . .	262
7.8.1.383 SYMBOLIC . . . . .	262
7.8.1.384 SYNCHRONIZED . . . . .	262
7.8.1.385 TALLYING . . . . .	262
7.8.1.386 TAPE . . . . .	262
7.8.1.387 TERM_ACCEPT . . . . .	262
7.8.1.388 TERM_ADD . . . . .	262
7.8.1.389 TERM_CALL . . . . .	262
7.8.1.390 TERM_COMPUTE . . . . .	263
7.8.1.391 TERM_DELETE . . . . .	263
7.8.1.392 TERM_DISPLAY . . . . .	263
7.8.1.393 TERM_DIVIDE . . . . .	263
7.8.1.394 TERM_EVALUATE . . . . .	263
7.8.1.395 TERM_IF . . . . .	263
7.8.1.396 TERM_MAX . . . . .	263
7.8.1.397 TERM_MULTIPLY . . . . .	263
7.8.1.398 TERM_NONE . . . . .	263
7.8.1.399 TERM_PERFORM . . . . .	263
7.8.1.400 TERM_READ . . . . .	264
7.8.1.401 TERM_RECEIVE . . . . .	264
7.8.1.402 TERM_RETURN . . . . .	264
7.8.1.403 TERM_REWRITE . . . . .	264
7.8.1.404 TERM_SEARCH . . . . .	264
7.8.1.405 TERM_START . . . . .	264
7.8.1.406 TERM_STRING . . . . .	264
7.8.1.407 TERM_SUBTRACT . . . . .	264

---

7.8.1.408 TERM_UNSTRING . . . . .	264
7.8.1.409 TERM_WRITE . . . . .	264
7.8.1.410 TERMINATE . . . . .	265
7.8.1.411 TEST . . . . .	265
7.8.1.412 THAN . . . . .	265
7.8.1.413 THEN . . . . .	265
7.8.1.414 THRU . . . . .	265
7.8.1.415 TIME . . . . .	265
7.8.1.416 TIMES . . . . .	265
7.8.1.417 TO . . . . .	265
7.8.1.418 TOK_FALSE . . . . .	265
7.8.1.419 TOK_FILE . . . . .	265
7.8.1.420 TOK_INITIAL . . . . .	266
7.8.1.421 TOK_NULL . . . . .	266
7.8.1.422 TOK_TRUE . . . . .	266
7.8.1.423 TOKEN_EOF . . . . .	266
7.8.1.424 TOP . . . . .	266
7.8.1.425 TRAILING . . . . .	266
7.8.1.426 TRANSFORM . . . . .	266
7.8.1.427 TRIM_FUNCTION . . . . .	266
7.8.1.428 TYPE . . . . .	266
7.8.1.429 UNARY_SIGN . . . . .	266
7.8.1.430 UNDERLINE . . . . .	267
7.8.1.431 UNIT . . . . .	267
7.8.1.432 UNLOCK . . . . .	267
7.8.1.433 UNSIGNED . . . . .	267
7.8.1.434 UNSIGNED_INT . . . . .	267
7.8.1.435 UNSIGNED_LONG . . . . .	267
7.8.1.436 UNSIGNED_SHORT . . . . .	267
7.8.1.437 UNSTRING . . . . .	267
7.8.1.438 UNTIL . . . . .	267
7.8.1.439 UP . . . . .	267
7.8.1.440 UPDATE . . . . .	268
7.8.1.441 UPON . . . . .	268

---

7.8.1.442 UPON_ARGUMENT_NUMBER . . . . .	268
7.8.1.443 UPON_COMMAND_LINE . . . . .	268
7.8.1.444 UPON_ENVIRONMENT_NAME . . . . .	268
7.8.1.445 UPON_ENVIRONMENT_VALUE . . . . .	268
7.8.1.446 UPPER_CASE_FUNC . . . . .	268
7.8.1.447 USAGE . . . . .	268
7.8.1.448 USE . . . . .	268
7.8.1.449 USING . . . . .	268
7.8.1.450 VALUE . . . . .	269
7.8.1.451 VARYING . . . . .	269
7.8.1.452 WAIT . . . . .	269
7.8.1.453 WHEN . . . . .	269
7.8.1.454 WHEN_COMPILED_FUNC . . . . .	269
7.8.1.455 WITH . . . . .	269
7.8.1.456 WORD . . . . .	269
7.8.1.457 WORDS . . . . .	269
7.8.1.458 WORKING_STORAGE . . . . .	269
7.8.1.459 WRITE . . . . .	269
7.8.1.460 YY_REDUCE_PRINT . . . . .	270
7.8.1.461 YY_STACK_PRINT . . . . .	270
7.8.1.462 YYABORT . . . . .	270
7.8.1.463 YYACCEPT . . . . .	270
7.8.1.464 YYBACKUP . . . . .	270
7.8.1.465 YYBISON . . . . .	271
7.8.1.466 yyclearin . . . . .	271
7.8.1.467 YYCOPY . . . . .	271
7.8.1.468 YYDEBUG . . . . .	271
7.8.1.469 YYDPRINTF . . . . .	271
7.8.1.470 YYDSYMPRINT . . . . .	271
7.8.1.471 YYDSYMPRINTF . . . . .	272
7.8.1.472 YYEMPTY . . . . .	272
7.8.1.473 YYEOF . . . . .	272
7.8.1.474 YYERRCODE . . . . .	272
7.8.1.475 yyerrok . . . . .	272

---

7.8.1.476 yyerror . . . . .	272
7.8.1.477 YYERROR . . . . .	272
7.8.1.478 YYERROR_VERBOSE . . . . .	272
7.8.1.479 YYERROR_VERBOSE . . . . .	273
7.8.1.480 YYFAIL . . . . .	273
7.8.1.481 YYFINAL . . . . .	273
7.8.1.482 YYPRINTF . . . . .	273
7.8.1.483 YYINITDEPTH . . . . .	273
7.8.1.484 YYLAST . . . . .	273
7.8.1.485 YYLEX . . . . .	273
7.8.1.486 YYLOC_DEFAULT . . . . .	273
7.8.1.487 YYLSP_NEEDED . . . . .	273
7.8.1.488 YYMAXDEPTH . . . . .	274
7.8.1.489 YYMAXUTOK . . . . .	274
7.8.1.490 YYNNTS . . . . .	274
7.8.1.491 YYNRULES . . . . .	274
7.8.1.492 YYNSTATES . . . . .	274
7.8.1.493 YYNTOKENS . . . . .	274
7.8.1.494 YYPACT_NINF . . . . .	274
7.8.1.495 YYPOPSTACK . . . . .	274
7.8.1.496 YYPURE . . . . .	274
7.8.1.497 YYRECOVERING . . . . .	274
7.8.1.498 YYSIZE_T . . . . .	274
7.8.1.499 YYSKELETON_NAME . . . . .	275
7.8.1.500 Yystack_ALLOC . . . . .	275
7.8.1.501 Yystack_BYTES . . . . .	275
7.8.1.502 Yystack_FREE . . . . .	275
7.8.1.503 Yystack_GAP_MAXIMUM . . . . .	275
7.8.1.504 Yystack_RELOCATE . . . . .	275
7.8.1.505 YYTABLE_NINF . . . . .	275
7.8.1.506 YYTERROR . . . . .	275
7.8.1.507 YYTOKENTYPE . . . . .	276
7.8.1.508 YYTRANSLATE . . . . .	276
7.8.1.509 YYUNDEFTOK . . . . .	276

---

7.8.1.510	YYYYDDD	276
7.8.1.511	YYYYMMDD	276
7.8.1.512	ZERO	276
7.8.2	Typedef Documentation	276
7.8.2.1	yysigned_char	276
7.8.3	Enumeration Type Documentation	276
7.8.3.1	yytokentype	276
7.8.4	Function Documentation	307
7.8.4.1	yyparse	307
7.8.5	Variable Documentation	307
7.8.5.1	current_paragraph	307
7.8.5.2	current_program	307
7.8.5.3	current_section	307
7.8.5.4	current_statement	308
7.8.5.5	functions_are_all	308
7.8.5.6	non_const_word	308
7.8.5.7	yychar	308
7.8.5.8	yydebug	308
7.8.5.9	yylval	308
7.8.5.10	yynerrs	308
7.9	cobc/parser.h File Reference	309
7.9.1	Define Documentation	326
7.9.1.1	ACCEPT	326
7.9.1.2	ACCESS	326
7.9.1.3	ADD	326
7.9.1.4	ADDRESS	326
7.9.1.5	ADVANCING	327
7.9.1.6	AFTER	327
7.9.1.7	ALL	327
7.9.1.8	ALLOCATE	327
7.9.1.9	ALPHABET	327
7.9.1.10	ALPHABETIC	327
7.9.1.11	ALPHABETIC_LOWER	327
7.9.1.12	ALPHABETIC_UPPER	327

---

7.9.1.13	ALPHANUMERIC	327
7.9.1.14	ALPHANUMERIC_EDITED	327
7.9.1.15	ALSO	328
7.9.1.16	ALTER	328
7.9.1.17	ALTERNATE	328
7.9.1.18	AND	328
7.9.1.19	ANY	328
7.9.1.20	ARE	328
7.9.1.21	AREA	328
7.9.1.22	ARGUMENT_NUMBER	328
7.9.1.23	ARGUMENT_VALUE	328
7.9.1.24	AS	328
7.9.1.25	ASCENDING	329
7.9.1.26	ASSIGN	329
7.9.1.27	AT	329
7.9.1.28	AUTO	329
7.9.1.29	AUTOMATIC	329
7.9.1.30	BACKGROUND_COLOR	329
7.9.1.31	BASED	329
7.9.1.32	BEFORE	329
7.9.1.33	BELL	329
7.9.1.34	BINARY	329
7.9.1.35	BINARY_C_LONG	330
7.9.1.36	BINARY_CHAR	330
7.9.1.37	BINARY_DOUBLE	330
7.9.1.38	BINARY_LONG	330
7.9.1.39	BINARY_SHORT	330
7.9.1.40	BLANK	330
7.9.1.41	BLANK_LINE	330
7.9.1.42	BLANK_SCREEN	330
7.9.1.43	BLINK	330
7.9.1.44	BLOCK	330
7.9.1.45	BOTTOM	331
7.9.1.46	BY	331



---

7.9.1.47	BYTE_LENGTH	331
7.9.1.48	CALL	331
7.9.1.49	CANCEL	331
7.9.1.50	CH	331
7.9.1.51	CHAINING	331
7.9.1.52	CHARACTER	331
7.9.1.53	CHARACTERS	331
7.9.1.54	CLASS	331
7.9.1.55	CLOSE	332
7.9.1.56	CODE	332
7.9.1.57	CODE_SET	332
7.9.1.58	COL	332
7.9.1.59	COLLATING	332
7.9.1.60	COLS	332
7.9.1.61	COLUMN	332
7.9.1.62	COLUMNS	332
7.9.1.63	COMMA	332
7.9.1.64	COMMA_DELIM	332
7.9.1.65	COMMAND_LINE	333
7.9.1.66	COMMIT	333
7.9.1.67	COMMON	333
7.9.1.68	COMP	333
7.9.1.69	COMP_1	333
7.9.1.70	COMP_2	333
7.9.1.71	COMP_3	333
7.9.1.72	COMP_4	333
7.9.1.73	COMP_5	333
7.9.1.74	COMP_X	333
7.9.1.75	COMPUTE	334
7.9.1.76	CONCATENATE_FUNC	334
7.9.1.77	CONFIGURATION	334
7.9.1.78	CONSTANT	334
7.9.1.79	CONTAINS	334
7.9.1.80	CONTENT	334

---

7.9.1.81	CONTINUE	334
7.9.1.82	CONTROL	334
7.9.1.83	CONTROL_FOOTING	334
7.9.1.84	CONTROL_HEADING	334
7.9.1.85	CONTROLS	335
7.9.1.86	CONVERTING	335
7.9.1.87	CORRESPONDING	335
7.9.1.88	COUNT	335
7.9.1.89	CRT	335
7.9.1.90	CURRENCY	335
7.9.1.91	CURRENT_DATE_FUNC	335
7.9.1.92	CURSOR	335
7.9.1.93	CYCLE	335
7.9.1.94	DATA	335
7.9.1.95	DATE	336
7.9.1.96	DAY	336
7.9.1.97	DAY_OF_WEEK	336
7.9.1.98	DE	336
7.9.1.99	DEBUGGING	336
7.9.1.100	DECIMAL_POINT	336
7.9.1.101	DECLARATIVES	336
7.9.1.102	DEFAULT	336
7.9.1.103	DELETE	336
7.9.1.104	DELIMITED	336
7.9.1.105	DELIMITER	337
7.9.1.106	DEPENDING	337
7.9.1.107	DESCENDING	337
7.9.1.108	DETAIL	337
7.9.1.109	DISK	337
7.9.1.110	DISPLAY	337
7.9.1.111	DIVIDE	337
7.9.1.112	DIVISION	337
7.9.1.113	DOWN	337
7.9.1.114	DUPLICATES	337

---

7.9.1.115 DYNAMIC . . . . .	338
7.9.1.116 EBCDIC . . . . .	338
7.9.1.117 ELSE . . . . .	338
7.9.1.118 END . . . . .	338
7.9.1.119 END_ACCEPT . . . . .	338
7.9.1.120 END_ADD . . . . .	338
7.9.1.121 END_CALL . . . . .	338
7.9.1.122 END_COMPUTE . . . . .	338
7.9.1.123 END_DELETE . . . . .	338
7.9.1.124 END_DISPLAY . . . . .	338
7.9.1.125 END_DIVIDE . . . . .	339
7.9.1.126 END_EVALUATE . . . . .	339
7.9.1.127 END_FUNCTION . . . . .	339
7.9.1.128 END_IF . . . . .	339
7.9.1.129 END_MULTIPLY . . . . .	339
7.9.1.130 END_PERFORM . . . . .	339
7.9.1.131 END_PROGRAM . . . . .	339
7.9.1.132 END_READ . . . . .	339
7.9.1.133 END_RETURN . . . . .	339
7.9.1.134 END_REWRITE . . . . .	339
7.9.1.135 END_SEARCH . . . . .	340
7.9.1.136 END_START . . . . .	340
7.9.1.137 END_STRING . . . . .	340
7.9.1.138 END_SUBTRACT . . . . .	340
7.9.1.139 END_UNSTRING . . . . .	340
7.9.1.140 END_WRITE . . . . .	340
7.9.1.141 ENTRY . . . . .	340
7.9.1.142 ENVIRONMENT . . . . .	340
7.9.1.143 ENVIRONMENT_NAME . . . . .	340
7.9.1.144 ENVIRONMENT_VALUE . . . . .	340
7.9.1.145 EOL . . . . .	341
7.9.1.146 EOP . . . . .	341
7.9.1.147 EOS . . . . .	341
7.9.1.148 EQUAL . . . . .	341

---

7.9.1.149 EQUALS . . . . .	341
7.9.1.150 ERASE . . . . .	341
7.9.1.151 ERROR . . . . .	341
7.9.1.152 ESCAPE . . . . .	341
7.9.1.153 EVALUATE . . . . .	341
7.9.1.154 EVENT_STATUS . . . . .	341
7.9.1.155 EXCEPTION . . . . .	342
7.9.1.156 EXCLUSIVE . . . . .	342
7.9.1.157 EXIT . . . . .	342
7.9.1.158 EXTEND . . . . .	342
7.9.1.159 EXTERNAL . . . . .	342
7.9.1.160 FD . . . . .	342
7.9.1.161 FILE_CONTROL . . . . .	342
7.9.1.162 FILE_ID . . . . .	342
7.9.1.163 FILLER . . . . .	342
7.9.1.164 FINAL . . . . .	342
7.9.1.165 FIRST . . . . .	343
7.9.1.166 FOOTING . . . . .	343
7.9.1.167 FOR . . . . .	343
7.9.1.168 FOREGROUND_COLOR . . . . .	343
7.9.1.169 FOREVER . . . . .	343
7.9.1.170 FREE . . . . .	343
7.9.1.171 FROM . . . . .	343
7.9.1.172 FULL . . . . .	343
7.9.1.173 FUNCTION . . . . .	343
7.9.1.174 FUNCTION_ID . . . . .	343
7.9.1.175 FUNCTION_NAME . . . . .	344
7.9.1.176 GE . . . . .	344
7.9.1.177 GENERATE . . . . .	344
7.9.1.178 GIVING . . . . .	344
7.9.1.179 GLOBAL . . . . .	344
7.9.1.180 GO . . . . .	344
7.9.1.181 GOBACK . . . . .	344
7.9.1.182 GREATER . . . . .	344

---

7.9.1.183 GROUP . . . . .	344
7.9.1.184 HEADING . . . . .	344
7.9.1.185 HIGH_VALUE . . . . .	345
7.9.1.186 HIGHLIGHT . . . . .	345
7.9.1.187 I_O . . . . .	345
7.9.1.188 I_O_CONTROL . . . . .	345
7.9.1.189 IDENTIFICATION . . . . .	345
7.9.1.190 IF . . . . .	345
7.9.1.191 IGNORE . . . . .	345
7.9.1.192 IGNORING . . . . .	345
7.9.1.193 IN . . . . .	345
7.9.1.194 INDEX . . . . .	345
7.9.1.195 INDEXED . . . . .	346
7.9.1.196 INDICATE . . . . .	346
7.9.1.197 INITIALIZE . . . . .	346
7.9.1.198 INITIALIZED . . . . .	346
7.9.1.199 INITIATE . . . . .	346
7.9.1.200 INPUT . . . . .	346
7.9.1.201 INPUT_OUTPUT . . . . .	346
7.9.1.202 INSPECT . . . . .	346
7.9.1.203 INTO . . . . .	346
7.9.1.204 INTRINSIC . . . . .	346
7.9.1.205 INVALID . . . . .	347
7.9.1.206 INVALID_KEY . . . . .	347
7.9.1.207 IS . . . . .	347
7.9.1.208 JUSTIFIED . . . . .	347
7.9.1.209 KEY . . . . .	347
7.9.1.210 LABEL . . . . .	347
7.9.1.211 LAST . . . . .	347
7.9.1.212 LAST_DETAIL . . . . .	347
7.9.1.213 LE . . . . .	347
7.9.1.214 LEADING . . . . .	347
7.9.1.215 LEFT . . . . .	348
7.9.1.216 LENGTH . . . . .	348

---

7.9.1.217 LESS . . . . .	348
7.9.1.218 LIMIT . . . . .	348
7.9.1.219 LIMITS . . . . .	348
7.9.1.220 LINAGE . . . . .	348
7.9.1.221 LINAGE_COUNTER . . . . .	348
7.9.1.222 LINE . . . . .	348
7.9.1.223 LINES . . . . .	348
7.9.1.224 LINKAGE . . . . .	348
7.9.1.225 LITERAL . . . . .	349
7.9.1.226 LOCAL_STORAGE . . . . .	349
7.9.1.227 LOCALE . . . . .	349
7.9.1.228 LOCALE_DT_FUNC . . . . .	349
7.9.1.229 LOCK . . . . .	349
7.9.1.230 LOW_VALUE . . . . .	349
7.9.1.231 LOWER_CASE_FUNC . . . . .	349
7.9.1.232 LOWLIGHT . . . . .	349
7.9.1.233 MANUAL . . . . .	349
7.9.1.234 MEMORY . . . . .	349
7.9.1.235 MERGE . . . . .	350
7.9.1.236 MINUS . . . . .	350
7.9.1.237 MNEMONIC_NAME . . . . .	350
7.9.1.238 MODE . . . . .	350
7.9.1.239 MOVE . . . . .	350
7.9.1.240 MULTIPLE . . . . .	350
7.9.1.241 MULTIPLY . . . . .	350
7.9.1.242 NATIONAL . . . . .	350
7.9.1.243 NATIONAL_EDITED . . . . .	350
7.9.1.244 NATIVE . . . . .	350
7.9.1.245 NE . . . . .	351
7.9.1.246 NEGATIVE . . . . .	351
7.9.1.247 NEXT . . . . .	351
7.9.1.248 NEXT_SENTENCE . . . . .	351
7.9.1.249 NO . . . . .	351
7.9.1.250 NO_ADVANCING . . . . .	351

---

7.9.1.251 NOT	351
7.9.1.252 NOT_END	351
7.9.1.253 NOT_EOP	351
7.9.1.254 NOT_EXCEPTION	351
7.9.1.255 NOT_INVALID_KEY	352
7.9.1.256 NOT_OVERFLOW	352
7.9.1.257 NOT_SIZE_ERROR	352
7.9.1.258 NUMBER	352
7.9.1.259 NUMBERS	352
7.9.1.260 NUMERIC	352
7.9.1.261 NUMERIC_EDITED	352
7.9.1.262 NUMVALC_FUNC	352
7.9.1.263 OBJECT_COMPUTER	352
7.9.1.264 OCCURS	352
7.9.1.265 OF	353
7.9.1.266 OFF	353
7.9.1.267 OMITTED	353
7.9.1.268 ON	353
7.9.1.269 ONLY	353
7.9.1.270 OPEN	353
7.9.1.271 OPTIONAL	353
7.9.1.272 OR	353
7.9.1.273 ORDER	353
7.9.1.274 ORGANIZATION	353
7.9.1.275 OTHER	354
7.9.1.276 OUTPUT	354
7.9.1.277 OVERFLOW	354
7.9.1.278 OVERLINE	354
7.9.1.279 PACKED_DECIMAL	354
7.9.1.280 PADDING	354
7.9.1.281 PAGE	354
7.9.1.282 PAGE_FOOTING	354
7.9.1.283 PAGE_HEADING	354
7.9.1.284 PARAGRAPH	354

---

7.9.1.285 PERFORM . . . . .	355
7.9.1.286 PICTURE . . . . .	355
7.9.1.287 PLUS . . . . .	355
7.9.1.288 POINTER . . . . .	355
7.9.1.289 POSITION . . . . .	355
7.9.1.290 POSITIVE . . . . .	355
7.9.1.291 PRESENT . . . . .	355
7.9.1.292 PREVIOUS . . . . .	355
7.9.1.293 PRINTER . . . . .	355
7.9.1.294 PRINTING . . . . .	355
7.9.1.295 PROCEDURE . . . . .	356
7.9.1.296 PROCEDURES . . . . .	356
7.9.1.297 PROCEED . . . . .	356
7.9.1.298 PROGRAM . . . . .	356
7.9.1.299 PROGRAM_ID . . . . .	356
7.9.1.300 PROGRAM_NAME . . . . .	356
7.9.1.301 PROGRAM_POINTER . . . . .	356
7.9.1.302 PROMPT . . . . .	356
7.9.1.303 QUOTE . . . . .	356
7.9.1.304 RANDOM . . . . .	356
7.9.1.305 RD . . . . .	357
7.9.1.306 READ . . . . .	357
7.9.1.307 RECORD . . . . .	357
7.9.1.308 RECORDING . . . . .	357
7.9.1.309 RECORDS . . . . .	357
7.9.1.310 RECURSIVE . . . . .	357
7.9.1.311 REDEFINES . . . . .	357
7.9.1.312 REEL . . . . .	357
7.9.1.313 REFERENCE . . . . .	357
7.9.1.314 RELATIVE . . . . .	357
7.9.1.315 RELEASE . . . . .	358
7.9.1.316 REMAINDER . . . . .	358
7.9.1.317 REMOVAL . . . . .	358
7.9.1.318 RENAMES . . . . .	358



---

7.9.1.319 REPLACING . . . . .	358
7.9.1.320 REPORT . . . . .	358
7.9.1.321 REPORT_FOOTING . . . . .	358
7.9.1.322 REPORT_HEADING . . . . .	358
7.9.1.323 REPORTING . . . . .	358
7.9.1.324 REPORTS . . . . .	358
7.9.1.325 REPOSITORY . . . . .	359
7.9.1.326 REQUIRED . . . . .	359
7.9.1.327 RESERVE . . . . .	359
7.9.1.328 RETURN . . . . .	359
7.9.1.329 RETURNING . . . . .	359
7.9.1.330 REVERSE_FUNC . . . . .	359
7.9.1.331 REVERSE_VIDEO . . . . .	359
7.9.1.332 REWIND . . . . .	359
7.9.1.333 REWRITE . . . . .	359
7.9.1.334 RIGHT . . . . .	359
7.9.1.335 ROLLBACK . . . . .	360
7.9.1.336 ROUNDED . . . . .	360
7.9.1.337 RUN . . . . .	360
7.9.1.338 SAME . . . . .	360
7.9.1.339 SCREEN . . . . .	360
7.9.1.340 SCREEN_CONTROL . . . . .	360
7.9.1.341 SCROLL . . . . .	360
7.9.1.342 SD . . . . .	360
7.9.1.343 SEARCH . . . . .	360
7.9.1.344 SECTION . . . . .	360
7.9.1.345 SECURE . . . . .	361
7.9.1.346 SEGMENT_LIMIT . . . . .	361
7.9.1.347 SELECT . . . . .	361
7.9.1.348 SEMI_COLON . . . . .	361
7.9.1.349 SENTENCE . . . . .	361
7.9.1.350 SEPARATE . . . . .	361
7.9.1.351 SEQUENCE . . . . .	361
7.9.1.352 SEQUENTIAL . . . . .	361

---

7.9.1.353 SET	361
7.9.1.354 SHARING	361
7.9.1.355 SIGN	362
7.9.1.356 SIGNED	362
7.9.1.357 SIGNED_INT	362
7.9.1.358 SIGNED_LONG	362
7.9.1.359 SIGNED_SHORT	362
7.9.1.360 SIZE	362
7.9.1.361 SIZE_ERROR	362
7.9.1.362 SORT	362
7.9.1.363 SORT_MERGE	362
7.9.1.364 SOURCE	362
7.9.1.365 SOURCE_COMPUTER	363
7.9.1.366 SPACE	363
7.9.1.367 SPECIAL_NAMES	363
7.9.1.368 STANDARD	363
7.9.1.369 STANDARD_1	363
7.9.1.370 STANDARD_2	363
7.9.1.371 START	363
7.9.1.372 STATUS	363
7.9.1.373 STOP	363
7.9.1.374 STRING	363
7.9.1.375 SUBSTITUTE_CASE_FUNC	364
7.9.1.376 SUBSTITUTE_FUNC	364
7.9.1.377 SUBTRACT	364
7.9.1.378 SUM	364
7.9.1.379 SUPPRESS	364
7.9.1.380 SYMBOLIC	364
7.9.1.381 SYNCHRONIZED	364
7.9.1.382 TALLYING	364
7.9.1.383 TAPE	364
7.9.1.384 TERMINATE	364
7.9.1.385 TEST	365
7.9.1.386 THAN	365

---

7.9.1.387 THEN	365
7.9.1.388 THRU	365
7.9.1.389 TIME	365
7.9.1.390 TIMES	365
7.9.1.391 TO	365
7.9.1.392 TOK_FALSE	365
7.9.1.393 TOK_FILE	365
7.9.1.394 TOK_INITIAL	365
7.9.1.395 TOK_NULL	366
7.9.1.396 TOK_TRUE	366
7.9.1.397 TOKEN_EOF	366
7.9.1.398 TOP	366
7.9.1.399 TRAILING	366
7.9.1.400 TRANSFORM	366
7.9.1.401 TRIM_FUNCTION	366
7.9.1.402 TYPE	366
7.9.1.403 UNARY_SIGN	366
7.9.1.404 UNDERLINE	366
7.9.1.405 UNIT	367
7.9.1.406 UNLOCK	367
7.9.1.407 UNSIGNED	367
7.9.1.408 UNSIGNED_INT	367
7.9.1.409 UNSIGNED_LONG	367
7.9.1.410 UNSIGNED_SHORT	367
7.9.1.411 UNSTRING	367
7.9.1.412 UNTIL	367
7.9.1.413 UP	367
7.9.1.414 UPDATE	367
7.9.1.415 UPON	368
7.9.1.416 UPON_ARGUMENT_NUMBER	368
7.9.1.417 UPON_COMMAND_LINE	368
7.9.1.418 UPON_ENVIRONMENT_NAME	368
7.9.1.419 UPON_ENVIRONMENT_VALUE	368
7.9.1.420 UPPER_CASE_FUNC	368

7.9.1.421	USAGE	368
7.9.1.422	USE	368
7.9.1.423	USING	368
7.9.1.424	VALUE	368
7.9.1.425	VARYING	369
7.9.1.426	WAIT	369
7.9.1.427	WHEN	369
7.9.1.428	WHEN_COMPILED_FUNC	369
7.9.1.429	WITH	369
7.9.1.430	WORD	369
7.9.1.431	WORDS	369
7.9.1.432	WORKING_STORAGE	369
7.9.1.433	WRITE	369
7.9.1.434	yystype	369
7.9.1.435	YYSTYPE_IS_DECLARED	370
7.9.1.436	YYSTYPE_IS_TRIVIAL	370
7.9.1.437	YYYYDDD	370
7.9.1.438	YYYYMMDD	370
7.9.1.439	ZERO	370
7.9.2	Typedef Documentation	370
7.9.2.1	YYSTYPE	370
7.9.3	Enumeration Type Documentation	370
7.9.3.1	ytoken_type	370
7.9.4	Variable Documentation	401
7.9.4.1	yylval	401
7.10	cobc/pplex.c File Reference	401
7.10.1	Define Documentation	404
7.10.1.1	BEGIN	404
7.10.1.2	COPY_STATE	404
7.10.1.3	ECHO	404
7.10.1.4	EOB_ACT_CONTINUE_SCAN	405
7.10.1.5	EOB_ACT_END_OF_FILE	405
7.10.1.6	EOB_ACT_LAST_MATCH	405
7.10.1.7	FLEX_SCANNER	405

---

7.10.1.8	INITIAL	405
7.10.1.9	PROCESS_STATE	405
7.10.1.10	PSEUDO_STATE	405
7.10.1.11	REJECT	405
7.10.1.12	unput	405
7.10.1.13	YY_AT_BOL	405
7.10.1.14	YY_BREAK	406
7.10.1.15	YY_BUF_SIZE	406
7.10.1.16	YY_BUFFER_EOF_PENDING	406
7.10.1.17	YY_BUFFER_NEW	406
7.10.1.18	YY_BUFFER_NORMAL	406
7.10.1.19	yy_create_buffer	406
7.10.1.20	YY_CURRENT_BUFFER	406
7.10.1.21	YY_DECL	406
7.10.1.22	yy_delete_buffer	406
7.10.1.23	YY_DO_BEFORE_ACTION	406
7.10.1.24	YY_END_OF_BUFFER	407
7.10.1.25	YY_END_OF_BUFFER_CHAR	407
7.10.1.26	YY_EXIT_FAILURE	407
7.10.1.27	YY_FATAL_ERROR	407
7.10.1.28	yy_flex_debug	407
7.10.1.29	YY_FLEX_MAJOR_VERSION	407
7.10.1.30	YY_FLEX_MINOR_VERSION	407
7.10.1.31	yy_flush_buffer	407
7.10.1.32	YY_FLUSH_BUFFER	407
7.10.1.33	yy_init_buffer	407
7.10.1.34	YY_INPUT	408
7.10.1.35	yy_load_buffer_state	408
7.10.1.36	YY_MORE_ADJ	408
7.10.1.37	YY_NEVER_INTERACTIVE	408
7.10.1.38	yy_new_buffer	408
7.10.1.39	YY_NEW_FILE	408
7.10.1.40	YY_NO_POP_STATE	408
7.10.1.41	YY_NO_PUSH_STATE	408

---

7.10.1.42 YY_NO_TOP_STATE . . . . .	408
7.10.1.43 YY_NULL . . . . .	408
7.10.1.44 YY_NUM_RULES . . . . .	409
7.10.1.45 YY_PROTO . . . . .	409
7.10.1.46 YY_READ_BUF_SIZE . . . . .	409
7.10.1.47 YY_RESTORE_YY_MORE_OFFSET . . . . .	409
7.10.1.48 YY_RULE_SETUP . . . . .	409
7.10.1.49 YY_SC_TO_UI . . . . .	409
7.10.1.50 yy_scan_buffer . . . . .	409
7.10.1.51 yy_scan_bytes . . . . .	409
7.10.1.52 yy_scan_string . . . . .	409
7.10.1.53 yy_set_bol . . . . .	410
7.10.1.54 yy_set_interactive . . . . .	410
7.10.1.55 YY_SKIP_YYWRAP . . . . .	410
7.10.1.56 YY_START . . . . .	410
7.10.1.57 YY_START_STACK_INCR . . . . .	410
7.10.1.58 YY_STATE_EOF . . . . .	410
7.10.1.59 yy_switch_to_buffer . . . . .	410
7.10.1.60 YY_USER_ACTION . . . . .	410
7.10.1.61 yyconst . . . . .	411
7.10.1.62 yyin . . . . .	411
7.10.1.63 yyleng . . . . .	411
7.10.1.64 yyless . . . . .	411
7.10.1.65 yylex . . . . .	411
7.10.1.66 yylex . . . . .	411
7.10.1.67 yymore . . . . .	412
7.10.1.68 yyout . . . . .	412
7.10.1.69 yyrestart . . . . .	412
7.10.1.70 YYSTATE . . . . .	412
7.10.1.71 yyterminate . . . . .	412
7.10.1.72 yytext . . . . .	412
7.10.1.73 yytext_ptr . . . . .	412
7.10.1.74 yywrap . . . . .	412
7.10.2 Typedef Documentation . . . . .	412

---

7.10.2.1	YY_BUFFER_STATE	412
7.10.2.2	YY_CHAR	412
7.10.2.3	yy_size_t	413
7.10.2.4	yy_state_type	413
7.10.3	Function Documentation	413
7.10.3.1	YY_PROTO	413
7.10.3.2	YY_PROTO	413
7.10.3.3	YY_PROTO	413
7.10.3.4	YY_PROTO	413
7.10.3.5	YY_PROTO	413
7.10.3.6	YY_PROTO	413
7.10.3.7	YY_PROTO	413
7.10.3.8	YY_PROTO	413
7.10.3.9	YY_PROTO	413
7.10.4	Variable Documentation	413
7.10.4.1	file	413
7.10.4.2	len	413
7.10.4.3	size	413
7.10.4.4	yy_bp	413
7.10.4.5	yyin	414
7.10.4.6	yy leng	414
7.10.4.7	yyout	414
7.10.4.8	yytext	414
7.11	cobc/ppparse.c File Reference	414
7.11.1	Define Documentation	424
7.11.1.1	BY	424
7.11.1.2	COPY	424
7.11.1.3	EQQ	424
7.11.1.4	IN	424
7.11.1.5	OF	424
7.11.1.6	OFF	424
7.11.1.7	pperror	425
7.11.1.8	PRINTING	425
7.11.1.9	REPLACE	425

---

7.11.1.10 REPLACING . . . . .	425
7.11.1.11 SUPPRESS . . . . .	425
7.11.1.12 TOKEN . . . . .	425
7.11.1.13 TOKEN_EOF . . . . .	425
7.11.1.14 YY_REDUCE_PRINT . . . . .	425
7.11.1.15 YY_STACK_PRINT . . . . .	425
7.11.1.16 YYABORT . . . . .	426
7.11.1.17 YYACCEPT . . . . .	426
7.11.1.18 YYBACKUP . . . . .	426
7.11.1.19 YBISON . . . . .	426
7.11.1.20 yychar . . . . .	426
7.11.1.21 yyclearin . . . . .	426
7.11.1.22 YYCOPY . . . . .	427
7.11.1.23 yydebug . . . . .	427
7.11.1.24 YYDEBUG . . . . .	427
7.11.1.25 YYDPRINTF . . . . .	427
7.11.1.26 YYDSYMPRINT . . . . .	427
7.11.1.27 YYDSYMPRINTF . . . . .	427
7.11.1.28 YYEMPTY . . . . .	428
7.11.1.29 YYEOF . . . . .	428
7.11.1.30 YYERRCODE . . . . .	428
7.11.1.31 yyerror . . . . .	428
7.11.1.32 yyerror . . . . .	428
7.11.1.33 YYERROR . . . . .	428
7.11.1.34 YYERROR_VERBOSE . . . . .	428
7.11.1.35 YYERROR_VERBOSE . . . . .	428
7.11.1.36 YYFAIL . . . . .	429
7.11.1.37 YYFINAL . . . . .	429
7.11.1.38 YYFPRINTF . . . . .	429
7.11.1.39 YYINITDEPTH . . . . .	429
7.11.1.40 YYLAST . . . . .	429
7.11.1.41 yylex . . . . .	429
7.11.1.42 YYLEX . . . . .	429
7.11.1.43 YYLOC_DEFAULT . . . . .	429



---

7.11.1.44	YYLSP_NEEDED	429
7.11.1.45	yylval	430
7.11.1.46	YYMAXDEPTH	430
7.11.1.47	YYMAXUTOK	430
7.11.1.48	yynerrs	430
7.11.1.49	YYNNTS	430
7.11.1.50	YYNRULES	430
7.11.1.51	YYNSTATES	430
7.11.1.52	YYNTOKENS	430
7.11.1.53	YYPACT_NINF	430
7.11.1.54	yyparse	430
7.11.1.55	YYPOPSTACK	431
7.11.1.56	YYPURE	431
7.11.1.57	YYRECOVERING	431
7.11.1.58	YYSIZE_T	431
7.11.1.59	YYSKELETON_NAME	431
7.11.1.60	YYSTACK_ALLOC	431
7.11.1.61	YYSTACK_BYTES	431
7.11.1.62	YYSTACK_FREE	431
7.11.1.63	YYSTACK_GAP_MAXIMUM	431
7.11.1.64	YYSTACK_RELOCATE	431
7.11.1.65	yystype	432
7.11.1.66	YYSTYPE_IS_DECLARED	432
7.11.1.67	YYSTYPE_IS_TRIVIAL	432
7.11.1.68	YYTABLE_NINF	432
7.11.1.69	YYTERROR	432
7.11.1.70	YYTOKENTYPE	432
7.11.1.71	YYTRANSLATE	432
7.11.1.72	YYUNDEFTOK	432
7.11.2	Typedef Documentation	433
7.11.2.1	yysigned_char	433
7.11.2.2	YYSTYPE	433
7.11.3	Enumeration Type Documentation	433
7.11.3.1	yytokentype	433

---

7.11.4	Function Documentation	457
7.11.4.1	yyparse	457
7.11.5	Variable Documentation	457
7.11.5.1	yychar	457
7.11.5.2	yydebug	457
7.11.5.3	yylval	457
7.11.5.4	yynerrs	457
7.12	cobc/ppparse.h File Reference	458
7.12.1	Define Documentation	466
7.12.1.1	BY	466
7.12.1.2	COPY	466
7.12.1.3	EQEQ	466
7.12.1.4	IN	466
7.12.1.5	OF	466
7.12.1.6	OFF	467
7.12.1.7	PRINTING	467
7.12.1.8	REPLACE	467
7.12.1.9	REPLACING	467
7.12.1.10	SUPPRESS	467
7.12.1.11	TOKEN	467
7.12.1.12	TOKEN_EOF	467
7.12.1.13	yystype	467
7.12.1.14	YYSTYPE_IS_DECLARED	467
7.12.1.15	YYSTYPE_IS_TRIVIAL	467
7.12.2	Typedef Documentation	468
7.12.2.1	YYSTYPE	468
7.12.3	Enumeration Type Documentation	468
7.12.3.1	yytokentype	468
7.12.4	Variable Documentation	492
7.12.4.1	pplval	492
7.13	cobc/reserved.c File Reference	492
7.13.1	Define Documentation	493
7.13.1.1	NUM_INTRINSICS	493
7.13.1.2	NUM_RESERVED_WORDS	493

---

7.13.2	Function Documentation	493
7.13.2.1	cb_init_reserved	493
7.13.2.2	cb_list_intrinsics	493
7.13.2.3	cb_list_mnemonics	494
7.13.2.4	cb_list_reserved	494
7.13.2.5	lookup_intrinsic	495
7.13.2.6	lookup_reserved_word	495
7.13.2.7	lookup_system_name	495
7.13.3	Variable Documentation	496
7.13.3.1	category	496
7.13.3.2	name	496
7.13.3.3	node	496
7.13.3.4	token	496
7.14	cobc/scanner.c File Reference	496
7.14.1	Define Documentation	499
7.14.1.1	BEGIN	499
7.14.1.2	DECIMAL_IS_COMMA	499
7.14.1.3	DECIMAL_IS_PERIOD	499
7.14.1.4	ECHO	499
7.14.1.5	EOB_ACT_CONTINUE_SCAN	499
7.14.1.6	EOB_ACT_END_OF_FILE	500
7.14.1.7	EOB_ACT_LAST_MATCH	500
7.14.1.8	FLEX_SCANNER	500
7.14.1.9	FUNCTION_STATE	500
7.14.1.10	INITIAL	500
7.14.1.11	PICTURE_STATE	500
7.14.1.12	REJECT	500
7.14.1.13	SET_LOCATION	500
7.14.1.14	unput	500
7.14.1.15	YY_AT_BOL	500
7.14.1.16	YY_BREAK	501
7.14.1.17	YY_BUF_SIZE	501
7.14.1.18	YY_BUFFER_EOF_PENDING	501
7.14.1.19	YY_BUFFER_NEW	501

---

7.14.1.20 YY_BUFFER_NORMAL . . . . .	501
7.14.1.21 YY_CURRENT_BUFFER . . . . .	501
7.14.1.22 YY_DECL . . . . .	501
7.14.1.23 YY_DO_BEFORE_ACTION . . . . .	501
7.14.1.24 YY_END_OF_BUFFER . . . . .	501
7.14.1.25 YY_END_OF_BUFFER_CHAR . . . . .	502
7.14.1.26 YY_EXIT_FAILURE . . . . .	502
7.14.1.27 YY_FATAL_ERROR . . . . .	502
7.14.1.28 YY_FLEX_MAJOR_VERSION . . . . .	502
7.14.1.29 YY_FLEX_MINOR_VERSION . . . . .	502
7.14.1.30 YY_FLUSH_BUFFER . . . . .	502
7.14.1.31 YY_INPUT . . . . .	502
7.14.1.32 YY_MORE_ADJ . . . . .	502
7.14.1.33 YY_NEVER_INTERACTIVE . . . . .	503
7.14.1.34 yy_new_buffer . . . . .	503
7.14.1.35 YY_NEW_FILE . . . . .	503
7.14.1.36 YY_NO_POP_STATE . . . . .	503
7.14.1.37 YY_NO_PUSH_STATE . . . . .	503
7.14.1.38 YY_NO_TOP_STATE . . . . .	503
7.14.1.39 YY_NULL . . . . .	503
7.14.1.40 YY_NUM_RULES . . . . .	503
7.14.1.41 YY_PROTO . . . . .	503
7.14.1.42 YY_READ_BUF_SIZE . . . . .	503
7.14.1.43 YY_RESTORE_YY_MORE_OFFSET . . . . .	504
7.14.1.44 YY_RULE_SETUP . . . . .	504
7.14.1.45 YY_SC_TO_UI . . . . .	504
7.14.1.46 yy_set_bol . . . . .	504
7.14.1.47 yy_set_interactive . . . . .	504
7.14.1.48 YY_SKIP_YYWRAP . . . . .	504
7.14.1.49 YY_START . . . . .	505
7.14.1.50 YY_START_STACK_INCR . . . . .	505
7.14.1.51 YY_STATE_EOF . . . . .	505
7.14.1.52 YY_USER_ACTION . . . . .	505
7.14.1.53 yyconst . . . . .	505

---

7.14.1.54	yyless	505
7.14.1.55	yyless	505
7.14.1.56	yymore	506
7.14.1.57	YYSTATE	506
7.14.1.58	yyterminate	506
7.14.1.59	yytext_ptr	506
7.14.1.60	yywrap	506
7.14.2	Typedef Documentation	506
7.14.2.1	YY_BUFFER_STATE	506
7.14.2.2	YY_CHAR	506
7.14.2.3	yy_size_t	506
7.14.2.4	yy_state_type	506
7.14.3	Function Documentation	507
7.14.3.1	YY_PROTO	507
7.14.3.2	YY_PROTO	507
7.14.3.3	YY_PROTO	507
7.14.3.4	YY_PROTO	507
7.14.3.5	YY_PROTO	507
7.14.3.6	YY_PROTO	507
7.14.3.7	YY_PROTO	507
7.14.3.8	YY_PROTO	507
7.14.3.9	YY_PROTO	507
7.14.4	Variable Documentation	507
7.14.4.1	file	507
7.14.4.2	len	507
7.14.4.3	size	507
7.14.4.4	yy_bp	507
7.14.4.5	yyin	507
7.14.4.6	yy leng	507
7.14.4.7	yyout	508
7.14.4.8	yytext	508
7.15	cobc/tree.c File Reference	508
7.15.1	Define Documentation	511
7.15.1.1	PIC_ALPHABETIC	511

---

7.15.1.2	PIC_ALPHABETIC_EDITED . . . . .	511
7.15.1.3	PIC_ALPHANUMERIC . . . . .	511
7.15.1.4	PIC_ALPHANUMERIC_EDITED . . . . .	511
7.15.1.5	PIC_EDITED . . . . .	511
7.15.1.6	PIC_NATIONAL . . . . .	511
7.15.1.7	PIC_NATIONAL_EDITED . . . . .	512
7.15.1.8	PIC_NUMERIC . . . . .	512
7.15.1.9	PIC_NUMERIC_EDITED . . . . .	512
7.15.2	Function Documentation . . . . .	512
7.15.2.1	build_file . . . . .	512
7.15.2.2	build_literal . . . . .	512
7.15.2.3	cb_build_alphabet_name . . . . .	513
7.15.2.4	cb_build_alphanumeric_literal . . . . .	513
7.15.2.5	cb_build_any_intrinsic . . . . .	513
7.15.2.6	cb_build_assign . . . . .	513
7.15.2.7	cb_build_binary_list . . . . .	514
7.15.2.8	cb_build_binary_op . . . . .	514
7.15.2.9	cb_build_call . . . . .	515
7.15.2.10	cb_build_cast . . . . .	515
7.15.2.11	cb_build_class_name . . . . .	516
7.15.2.12	cb_build_constant . . . . .	516
7.15.2.13	cb_build_continue . . . . .	516
7.15.2.14	cb_build_decimal . . . . .	517
7.15.2.15	cb_build_field . . . . .	517
7.15.2.16	cb_build_field_reference . . . . .	517
7.15.2.17	cb_build_filler . . . . .	518
7.15.2.18	cb_build_funcall . . . . .	518
7.15.2.19	cb_build_goto . . . . .	518
7.15.2.20	cb_build_if . . . . .	519
7.15.2.21	cb_build_implicit_field . . . . .	519
7.15.2.22	cb_build_initialize . . . . .	519
7.15.2.23	cb_build_intrinsic . . . . .	519
7.15.2.24	cb_build_label . . . . .	522
7.15.2.25	cb_build_list . . . . .	522

---

7.15.2.26	<a href="#">cb_build_locale_name</a>	523
7.15.2.27	<a href="#">cb_build_numeric_literal</a>	523
7.15.2.28	<a href="#">cb_build_perform</a>	523
7.15.2.29	<a href="#">cb_build_perform_varying</a>	524
7.15.2.30	<a href="#">cb_build_picture</a>	524
7.15.2.31	<a href="#">cb_build_program</a>	529
7.15.2.32	<a href="#">cb_build_reference</a>	530
7.15.2.33	<a href="#">cb_build_search</a>	530
7.15.2.34	<a href="#">cb_build_statement</a>	530
7.15.2.35	<a href="#">cb_build_string</a>	531
7.15.2.36	<a href="#">cb_build_system_name</a>	531
7.15.2.37	<a href="#">cb_concat_literals</a>	531
7.15.2.38	<a href="#">cb_define</a>	532
7.15.2.39	<a href="#">cb_define_system_name</a>	533
7.15.2.40	<a href="#">cb_field</a>	533
7.15.2.41	<a href="#">cb_field_add</a>	533
7.15.2.42	<a href="#">cb_field_founder</a>	534
7.15.2.43	<a href="#">cb_field_size</a>	534
7.15.2.44	<a href="#">cb_field_subordinate</a>	534
7.15.2.45	<a href="#">cb_field_variable_address</a>	535
7.15.2.46	<a href="#">cb_field_variable_size</a>	535
7.15.2.47	<a href="#">cb_fits_int</a>	535
7.15.2.48	<a href="#">cb_fits_long_long</a>	536
7.15.2.49	<a href="#">cb_get_int</a>	537
7.15.2.50	<a href="#">cb_get_long_long</a>	538
7.15.2.51	<a href="#">cb_init_constants</a>	538
7.15.2.52	<a href="#">cb_int</a>	539
7.15.2.53	<a href="#">cb_list_add</a>	539
7.15.2.54	<a href="#">cb_list_append</a>	539
7.15.2.55	<a href="#">cb_list_length</a>	540
7.15.2.56	<a href="#">cb_list_map</a>	540
7.15.2.57	<a href="#">cb_list_reverse</a>	540
7.15.2.58	<a href="#">cb_name</a>	541
7.15.2.59	<a href="#">cb_ref</a>	541

7.15.2.60	<code>cb_tree_category</code>	544
7.15.2.61	<code>cb_tree_class</code>	545
7.15.2.62	<code>cb_tree_type</code>	545
7.15.2.63	<code>finalize_file</code>	546
7.15.2.64	<code>validate_file</code>	548
7.15.3	Variable Documentation	549
7.15.3.1	<code>cb_any</code>	549
7.15.3.2	<code>cb_error_node</code>	549
7.15.3.3	<code>cb_false</code>	549
7.15.3.4	<code>cb_high</code>	549
7.15.3.5	<code>cb_i</code>	549
7.15.3.6	<code>cb_int0</code>	549
7.15.3.7	<code>cb_int1</code>	549
7.15.3.8	<code>cb_int2</code>	549
7.15.3.9	<code>cb_int3</code>	549
7.15.3.10	<code>cb_int4</code>	550
7.15.3.11	<code>cb_int5</code>	550
7.15.3.12	<code>cb_intr_e</code>	550
7.15.3.13	<code>cb_intr_pi</code>	550
7.15.3.14	<code>cb_intr_whencomp</code>	550
7.15.3.15	<code>cb_low</code>	550
7.15.3.16	<code>cb_norm_high</code>	550
7.15.3.17	<code>cb_norm_low</code>	550
7.15.3.18	<code>cb_null</code>	550
7.15.3.19	<code>cb_one</code>	550
7.15.3.20	<code>cb_quote</code>	551
7.15.3.21	<code>cb_space</code>	551
7.15.3.22	<code>cb_standard_error_handler</code>	551
7.15.3.23	<code>cb_true</code>	551
7.15.3.24	<code>cb_zero</code>	551
7.15.3.25	<code>gen_screen_ptr</code>	551
7.16	<code>cobc/tree.h</code> File Reference	551
7.16.1	Define Documentation	564
7.16.1.1	<code>CB_AFTER</code>	564



---

7.16.1.2	CB_ALPHABET_NAME	564
7.16.1.3	CB_ALPHABET_NAME_P	565
7.16.1.4	CB_ASSIGN	565
7.16.1.5	CB_ASSIGN_P	565
7.16.1.6	CB_BEFORE	565
7.16.1.7	CB_BINARY_OP	565
7.16.1.8	CB_BINARY_OP_P	565
7.16.1.9	cb_build_cast_addr_of_addr	565
7.16.1.10	cb_build_cast_address	565
7.16.1.11	cb_build_cast_integer	565
7.16.1.12	cb_build_cast_length	565
7.16.1.13	cb_build_cast_ppointer	566
7.16.1.14	cb_build_funcall_0	566
7.16.1.15	cb_build_funcall_1	566
7.16.1.16	cb_build_funcall_2	566
7.16.1.17	cb_build_funcall_3	566
7.16.1.18	cb_build_funcall_4	566
7.16.1.19	cb_build_funcall_5	566
7.16.1.20	cb_build_funcall_6	566
7.16.1.21	cb_build_funcall_7	566
7.16.1.22	cb_build_negation	567
7.16.1.23	cb_build_pair	567
7.16.1.24	cb_build_parenthesis	567
7.16.1.25	cb_build_string0	567
7.16.1.26	CB_CALL	567
7.16.1.27	CB_CALL_BY_CONTENT	567
7.16.1.28	CB_CALL_BY_REFERENCE	567
7.16.1.29	CB_CALL_BY_VALUE	567
7.16.1.30	CB_CALL_P	567
7.16.1.31	CB_CAST	567
7.16.1.32	CB_CAST_P	568
7.16.1.33	CB_CHAIN	568
7.16.1.34	CB_CLASS_NAME	568
7.16.1.35	CB_CLASS_NAME_P	568

---

7.16.1.36	cb_cons	568
7.16.1.37	CB_CONST	568
7.16.1.38	CB_CONST_P	568
7.16.1.39	CB_CONTINUE	568
7.16.1.40	CB_CONTINUE_P	568
7.16.1.41	CB_DECIMAL	568
7.16.1.42	CB_DECIMAL_P	569
7.16.1.43	CB_FIELD	569
7.16.1.44	CB_FIELD_P	569
7.16.1.45	CB_FILE	569
7.16.1.46	CB_FILE_P	569
7.16.1.47	CB_FUNCALL	569
7.16.1.48	CB_FUNCALL_P	569
7.16.1.49	CB_FUNCTION_TYPE	569
7.16.1.50	CB_GOTO	569
7.16.1.51	CB_GOTO_P	569
7.16.1.52	CB_IF	570
7.16.1.53	CB_IF_P	570
7.16.1.54	CB_INDEX_P	570
7.16.1.55	CB_INITIALIZE	570
7.16.1.56	CB_INITIALIZE_P	570
7.16.1.57	CB_INTEGER	570
7.16.1.58	CB_INTEGER_P	570
7.16.1.59	CB_INTRINSIC	570
7.16.1.60	CB_INTRINSIC_P	570
7.16.1.61	CB_LABEL	571
7.16.1.62	CB_LABEL_P	571
7.16.1.63	CB_LIST	571
7.16.1.64	cb_list_init	571
7.16.1.65	CB_LIST_P	571
7.16.1.66	CB_LITERAL	571
7.16.1.67	CB_LITERAL_P	571
7.16.1.68	CB_LOCALE_NAME	571
7.16.1.69	CB_LOCALE_NAME_P	571

---

7.16.1.70 CB_NAME	571
7.16.1.71 CB_NUMERIC_LITERAL_P	572
7.16.1.72 CB_PAIR_P	572
7.16.1.73 CB_PAIR_X	572
7.16.1.74 CB_PAIR_Y	572
7.16.1.75 CB_PERFORM	572
7.16.1.76 CB_PERFORM_P	572
7.16.1.77 CB_PERFORM_VARYING	572
7.16.1.78 CB_PICTURE	572
7.16.1.79 CB_PICTURE_P	572
7.16.1.80 CB_PREFIX_ATTR	572
7.16.1.81 CB_PREFIX_BASE	573
7.16.1.82 CB_PREFIX_CONST	573
7.16.1.83 CB_PREFIX_DECIMAL	573
7.16.1.84 CB_PREFIX_FIELD	573
7.16.1.85 CB_PREFIX_FILE	573
7.16.1.86 CB_PREFIX_KEYS	573
7.16.1.87 CB_PREFIX_LABEL	573
7.16.1.88 CB_PREFIX_SEQUENCE	573
7.16.1.89 CB_PROGRAM_TYPE	573
7.16.1.90 CB_PURPOSE	573
7.16.1.91 CB_PURPOSE_INT	574
7.16.1.92 CB_REF_OR_FIELD_P	574
7.16.1.93 CB_REFERENCE	574
7.16.1.94 CB_REFERENCE_P	574
7.16.1.95 CB_SEARCH	574
7.16.1.96 CB_SEARCH_P	574
7.16.1.97 CB_SIZE_1	574
7.16.1.98 CB_SIZE_2	574
7.16.1.99 CB_SIZE_4	574
7.16.1.100 CB_SIZE_8	574
7.16.1.101 CB_SIZE_AUTO	575
7.16.1.102 CB_SIZE_UNSIGNED	575
7.16.1.103 CB_SIZES	575

7.16.1.104	CB_SIZES_INT	575
7.16.1.105	CB_SIZES_INT_UNSIGNED	575
7.16.1.106	CB_STATEMENT	575
7.16.1.107	CB_STATEMENT_P	575
7.16.1.108	CB_STRING	575
7.16.1.109	CB_STRING_P	575
7.16.1.110	CB_SYSTEM_NAME	575
7.16.1.111	CB_SYSTEM_NAME_P	576
7.16.1.112	CB_TREE	576
7.16.1.113	CB_TREE_CAST	576
7.16.1.114	CB_TREE_CATEGORY	576
7.16.1.115	CB_TREE_CLASS	576
7.16.1.116	CB_TREE_TAG	576
7.16.1.117	CB_VALUE	576
7.16.1.118	CB_WORD_HASH_SIZE	576
7.16.1.119	COB_MAX_SUBSCRIPTS	576
7.16.1.120	YYSTYPE	576
7.16.2	Typedef Documentation	577
7.16.2.1	cb_tree	577
7.16.3	Enumeration Type Documentation	577
7.16.3.1	cb_alphabet_name_type	577
7.16.3.2	cb_cast_type	577
7.16.3.3	cb_category	578
7.16.3.4	cb_class	578
7.16.3.5	cb_device_name	579
7.16.3.6	cb_feature_name	579
7.16.3.7	cb_intr_enum	580
7.16.3.8	cb_operand_type	584
7.16.3.9	cb_perform_type	584
7.16.3.10	cb_storage	585
7.16.3.11	cb_switch_name	585
7.16.3.12	cb_system_name_category	586
7.16.3.13	cb_tag	586
7.16.3.14	cb_usage	588

---

7.16.4	Function Documentation	589
7.16.4.1	ambiguous_error	589
7.16.4.2	build_file	590
7.16.4.3	build_literal	590
7.16.4.4	cb_add_78	591
7.16.4.5	cb_build_add	591
7.16.4.6	cb_build_address	591
7.16.4.7	cb_build_alphabet_name	592
7.16.4.8	cb_build_alphanumeric_literal	592
7.16.4.9	cb_build_any_intrinsic	592
7.16.4.10	cb_build_assign	592
7.16.4.11	cb_build_assignment_name	593
7.16.4.12	cb_build_binary_list	594
7.16.4.13	cb_build_binary_op	594
7.16.4.14	cb_build_call	595
7.16.4.15	cb_build_cast	596
7.16.4.16	cb_build_class_name	596
7.16.4.17	cb_build_cond	596
7.16.4.18	cb_build_const_length	599
7.16.4.19	cb_build_constant	599
7.16.4.20	cb_build_continue	599
7.16.4.21	cb_build_converting	600
7.16.4.22	cb_build_decimal	600
7.16.4.23	cb_build_display_upon	600
7.16.4.24	cb_build_display_upon_direct	601
7.16.4.25	cb_build_expr	601
7.16.4.26	cb_build_field	602
7.16.4.27	cb_build_field_reference	603
7.16.4.28	cb_build_field_tree	603
7.16.4.29	cb_build_filler	606
7.16.4.30	cb_build_funcall	606
7.16.4.31	cb_build_goto	606
7.16.4.32	cb_build_identifier	607
7.16.4.33	cb_build_if	610

---

7.16.4.34	cb_build_implicit_field	610
7.16.4.35	cb_build_index	610
7.16.4.36	cb_build_initialize	611
7.16.4.37	cb_build_inspect_region	611
7.16.4.38	cb_build_inspect_region_start	611
7.16.4.39	cb_build_intrinsic	612
7.16.4.40	cb_build_label	614
7.16.4.41	cb_build_length	615
7.16.4.42	cb_build_list	615
7.16.4.43	cb_build_locale_name	616
7.16.4.44	cb_build_move	616
7.16.4.45	cb_build_numeric_literal	617
7.16.4.46	cb_build_perform	617
7.16.4.47	cb_build_perform_exit	618
7.16.4.48	cb_build_perform_forever	618
7.16.4.49	cb_build_perform_once	618
7.16.4.50	cb_build_perform_times	619
7.16.4.51	cb_build_perform_until	619
7.16.4.52	cb_build_perform_varying	619
7.16.4.53	cb_build_picture	619
7.16.4.54	cb_build_ppointer	625
7.16.4.55	cb_build_program	625
7.16.4.56	cb_build_program_id	626
7.16.4.57	cb_build_reference	627
7.16.4.58	cb_build_registers	627
7.16.4.59	cb_build_replacing_all	628
7.16.4.60	cb_build_replacing_characters	628
7.16.4.61	cb_build_replacing_first	628
7.16.4.62	cb_build_replacing_leading	629
7.16.4.63	cb_build_replacing_trailing	629
7.16.4.64	cb_build_search	629
7.16.4.65	cb_build_section_name	629
7.16.4.66	cb_build_statement	630
7.16.4.67	cb_build_string	630

---

7.16.4.68	cb_build_sub	630
7.16.4.69	cb_build_system_name	631
7.16.4.70	cb_build_tarrying_all	631
7.16.4.71	cb_build_tarrying_characters	632
7.16.4.72	cb_build_tarrying_data	632
7.16.4.73	cb_build_tarrying_leading	632
7.16.4.74	cb_build_tarrying_trailing	632
7.16.4.75	cb_build_tarrying_value	633
7.16.4.76	cb_build_unstring_delimited	633
7.16.4.77	cb_build_unstring_into	633
7.16.4.78	cb_build_write_advancing_lines	633
7.16.4.79	cb_build_write_advancing_mnemonic	634
7.16.4.80	cb_build_write_advancing_page	634
7.16.4.81	cb_check_numeric_value	634
7.16.4.82	cb_clear_real_field	635
7.16.4.83	cb_concat_literals	635
7.16.4.84	cb_define	636
7.16.4.85	cb_define_switch_name	636
7.16.4.86	cb_define_system_name	637
7.16.4.87	cb_emit_accept	637
7.16.4.88	cb_emit_accept_arg_number	639
7.16.4.89	cb_emit_accept_arg_value	639
7.16.4.90	cb_emit_accept_command_line	639
7.16.4.91	cb_emit_accept_date	640
7.16.4.92	cb_emit_accept_date_yyyymmdd	640
7.16.4.93	cb_emit_accept_day	640
7.16.4.94	cb_emit_accept_day_of_week	640
7.16.4.95	cb_emit_accept_day_yyyddd	640
7.16.4.96	cb_emit_accept_environment	641
7.16.4.97	cb_emit_accept_line_or_col	641
7.16.4.98	cb_emit_accept_mnemonic	641
7.16.4.99	cb_emit_accept_name	642
7.16.4.100	cb_emit_accept_time	642
7.16.4.101	cb_emit_allocate	642

---

7.16.4.102cb_emit_arg_number . . . . .	643
7.16.4.103cb_emit_arithmetic . . . . .	643
7.16.4.104cb_emit_call . . . . .	645
7.16.4.105cb_emit_cancel . . . . .	646
7.16.4.106cb_emit_close . . . . .	646
7.16.4.107cb_emit_command_line . . . . .	646
7.16.4.108cb_emit_commit . . . . .	647
7.16.4.109cb_emit_continue . . . . .	647
7.16.4.110cb_emit_corresponding . . . . .	647
7.16.4.111cb_emit_delete . . . . .	647
7.16.4.112cb_emit_display . . . . .	648
7.16.4.113cb_emit_divide . . . . .	650
7.16.4.114cb_emit_env_name . . . . .	650
7.16.4.115cb_emit_env_value . . . . .	651
7.16.4.116cb_emit_evaluate . . . . .	651
7.16.4.117cb_emit_exit . . . . .	651
7.16.4.118cb_emit_free . . . . .	651
7.16.4.119cb_emit_get_environment . . . . .	652
7.16.4.120cb_emit_goto . . . . .	652
7.16.4.121cb_emit_if . . . . .	653
7.16.4.122cb_emit_initialize . . . . .	653
7.16.4.123cb_emit_inspect . . . . .	653
7.16.4.124cb_emit_move . . . . .	654
7.16.4.125cb_emit_move_corresponding . . . . .	654
7.16.4.126cb_emit_open . . . . .	655
7.16.4.127cb_emit_perform . . . . .	655
7.16.4.128cb_emit_read . . . . .	656
7.16.4.129cb_emit_release . . . . .	657
7.16.4.130cb_emit_return . . . . .	657
7.16.4.131cb_emit_rewrite . . . . .	658
7.16.4.132cb_emit_rollback . . . . .	658
7.16.4.133cb_emit_search . . . . .	659
7.16.4.134cb_emit_search_all . . . . .	659
7.16.4.135cb_emit_set_false . . . . .	659



---

7.16.4.136	cb_emit_set_on_off	660
7.16.4.137	cb_emit_set_to	660
7.16.4.138	cb_emit_set_true	662
7.16.4.139	cb_emit_set_up_down	662
7.16.4.140	cb_emit_setenv	663
7.16.4.141	cb_emit_sort_finish	663
7.16.4.142	cb_emit_sort_giving	663
7.16.4.143	cb_emit_sort_init	664
7.16.4.144	cb_emit_sort_input	665
7.16.4.145	cb_emit_sort_output	665
7.16.4.146	cb_emit_sort_using	665
7.16.4.147	cb_emit_start	665
7.16.4.148	cb_emit_stop_run	666
7.16.4.149	cb_emit_string	666
7.16.4.150	cb_emit_unlock	666
7.16.4.151	cb_emit_unstring	667
7.16.4.152	cb_emit_write	667
7.16.4.153	cb_encode_program_id	668
7.16.4.154	cb_error_x	669
7.16.4.155	cb_field	669
7.16.4.156	cb_field_add	670
7.16.4.157	cb_field_founder	670
7.16.4.158	cb_field_size	670
7.16.4.159	cb_field_subordinate	671
7.16.4.160	cb_field_variable_address	671
7.16.4.161	cb_field_variable_size	671
7.16.4.162	cb_fits_int	672
7.16.4.163	cb_fits_long_long	673
7.16.4.164	cb_get_int	673
7.16.4.165	cb_get_level	674
7.16.4.166	cb_get_long_long	675
7.16.4.167	cb_init_constants	675
7.16.4.168	cb_init_reserved	676
7.16.4.169	cb_init_tarrying	676

---

7.16.4.170cb_int . . . . .	676
7.16.4.171cb_list_add . . . . .	677
7.16.4.172cb_list_append . . . . .	677
7.16.4.173cb_list_intrinsics . . . . .	677
7.16.4.174cb_list_length . . . . .	678
7.16.4.175cb_list_map . . . . .	678
7.16.4.176cb_list_mnemonics . . . . .	678
7.16.4.177cb_list_reserved . . . . .	679
7.16.4.178cb_list_reverse . . . . .	679
7.16.4.179cb_name . . . . .	679
7.16.4.180cb_ref . . . . .	680
7.16.4.181cb_reset_78 . . . . .	683
7.16.4.182cb_reset_in_procedure . . . . .	683
7.16.4.183cb_resolve_redefines . . . . .	683
7.16.4.184cb_set_in_procedure . . . . .	684
7.16.4.185cb_tree_category . . . . .	684
7.16.4.186cb_tree_class . . . . .	685
7.16.4.187cb_tree_type . . . . .	685
7.16.4.188cb_validate_78_item . . . . .	686
7.16.4.189cb_validate_88_item . . . . .	687
7.16.4.190cb_validate_field . . . . .	687
7.16.4.191cb_validate_program_body . . . . .	688
7.16.4.192cb_validate_program_data . . . . .	688
7.16.4.193cb_validate_program_environment . . . . .	691
7.16.4.194cb_warning_x . . . . .	694
7.16.4.195check_filler_name . . . . .	695
7.16.4.196check_level_78 . . . . .	695
7.16.4.197cobc_tree_cast_error . . . . .	695
7.16.4.198codegen . . . . .	695
7.16.4.199finalize_file . . . . .	705
7.16.4.200group_error . . . . .	707
7.16.4.201level_except_error . . . . .	707
7.16.4.202level_redundant_error . . . . .	708
7.16.4.203level_require_error . . . . .	708

---

7.16.4.204	lookup_intrinsic	708
7.16.4.205	lookup_reserved_word	709
7.16.4.206	lookup_system_name	709
7.16.4.207	redefinition_error	709
7.16.4.208	redefinition_warning	710
7.16.4.209	undefined_error	710
7.16.4.210	validate_file	710
7.16.4.211	validate_move	711
7.16.5	Variable Documentation	719
7.16.5.1	cb_any	719
7.16.5.2	cb_error_node	719
7.16.5.3	cb_false	719
7.16.5.4	cb_high	719
7.16.5.5	cb_i	720
7.16.5.6	cb_int0	720
7.16.5.7	cb_int1	720
7.16.5.8	cb_int2	720
7.16.5.9	cb_int3	720
7.16.5.10	cb_int4	720
7.16.5.11	cb_int5	720
7.16.5.12	cb_intr_e	720
7.16.5.13	cb_intr_pi	720
7.16.5.14	cb_intr_whencomp	720
7.16.5.15	cb_low	721
7.16.5.16	cb_needs_01	721
7.16.5.17	cb_norm_high	721
7.16.5.18	cb_norm_low	721
7.16.5.19	cb_null	721
7.16.5.20	cb_one	721
7.16.5.21	cb_quote	721
7.16.5.22	cb_space	721
7.16.5.23	cb_standard_error_handler	721
7.16.5.24	cb_true	721
7.16.5.25	cb_zero	722

7.16.5.26	gen_screen_ptr	722
7.16.5.27	non_const_word	722
7.17	cobc/typeck.c File Reference	722
7.17.1	Define Documentation	726
7.17.1.1	cb_emit	726
7.17.1.2	cb_emit_list	726
7.17.1.3	COB_SYSTEM_GEN	726
7.17.1.4	dpush	726
7.17.1.5	START_STACK_SIZE	726
7.17.1.6	TOKEN	726
7.17.1.7	VALUE	726
7.17.2	Function Documentation	727
7.17.2.1	cb_build_add	727
7.17.2.2	cb_build_address	727
7.17.2.3	cb_build_assignment_name	728
7.17.2.4	cb_build_cond	729
7.17.2.5	cb_build_const_length	731
7.17.2.6	cb_build_converting	731
7.17.2.7	cb_build_display_upon	732
7.17.2.8	cb_build_display_upon_direct	732
7.17.2.9	cb_build_expr	733
7.17.2.10	cb_build_identifier	734
7.17.2.11	cb_build_index	737
7.17.2.12	cb_build_inspect_region	737
7.17.2.13	cb_build_inspect_region_start	738
7.17.2.14	cb_build_length	738
7.17.2.15	cb_build_move	739
7.17.2.16	cb_build_perform_exit	740
7.17.2.17	cb_build_perform_forever	740
7.17.2.18	cb_build_perform_once	740
7.17.2.19	cb_build_perform_times	741
7.17.2.20	cb_build_perform_until	741
7.17.2.21	cb_build_ppointer	741
7.17.2.22	cb_build_program_id	742

7.17.2.23	<a href="#">cb_build_registers</a>	742
7.17.2.24	<a href="#">cb_build_replacing_all</a>	744
7.17.2.25	<a href="#">cb_build_replacing_characters</a>	744
7.17.2.26	<a href="#">cb_build_replacing_first</a>	744
7.17.2.27	<a href="#">cb_build_replacing_leading</a>	744
7.17.2.28	<a href="#">cb_build_replacing_trailing</a>	744
7.17.2.29	<a href="#">cb_build_section_name</a>	745
7.17.2.30	<a href="#">cb_build_sub</a>	745
7.17.2.31	<a href="#">cb_build_tarrying_all</a>	746
7.17.2.32	<a href="#">cb_build_tarrying_characters</a>	746
7.17.2.33	<a href="#">cb_build_tarrying_data</a>	746
7.17.2.34	<a href="#">cb_build_tarrying_leading</a>	746
7.17.2.35	<a href="#">cb_build_tarrying_trailing</a>	747
7.17.2.36	<a href="#">cb_build_tarrying_value</a>	747
7.17.2.37	<a href="#">cb_build_unstring_delimited</a>	747
7.17.2.38	<a href="#">cb_build_unstring_into</a>	747
7.17.2.39	<a href="#">cb_build_write_advancing_lines</a>	748
7.17.2.40	<a href="#">cb_build_write_advancing_mnemonic</a>	748
7.17.2.41	<a href="#">cb_build_write_advancing_page</a>	748
7.17.2.42	<a href="#">cb_check_numeric_value</a>	749
7.17.2.43	<a href="#">cb_define_switch_name</a>	749
7.17.2.44	<a href="#">cb_emit_accept</a>	749
7.17.2.45	<a href="#">cb_emit_accept_arg_number</a>	751
7.17.2.46	<a href="#">cb_emit_accept_arg_value</a>	751
7.17.2.47	<a href="#">cb_emit_accept_command_line</a>	751
7.17.2.48	<a href="#">cb_emit_accept_date</a>	752
7.17.2.49	<a href="#">cb_emit_accept_date_yyyymmdd</a>	752
7.17.2.50	<a href="#">cb_emit_accept_day</a>	752
7.17.2.51	<a href="#">cb_emit_accept_day_of_week</a>	752
7.17.2.52	<a href="#">cb_emit_accept_day_yyyddd</a>	753
7.17.2.53	<a href="#">cb_emit_accept_environment</a>	753
7.17.2.54	<a href="#">cb_emit_accept_line_or_col</a>	753
7.17.2.55	<a href="#">cb_emit_accept_mnemonic</a>	753
7.17.2.56	<a href="#">cb_emit_accept_name</a>	754

---

7.17.2.57	cb_emit_accept_time	754
7.17.2.58	cb_emit_allocate	754
7.17.2.59	cb_emit_arg_number	755
7.17.2.60	cb_emit_arithmetic	755
7.17.2.61	cb_emit_call	757
7.17.2.62	cb_emit_cancel	758
7.17.2.63	cb_emit_close	758
7.17.2.64	cb_emit_command_line	758
7.17.2.65	cb_emit_commit	759
7.17.2.66	cb_emit_continue	759
7.17.2.67	cb_emit_corresponding	759
7.17.2.68	cb_emit_delete	759
7.17.2.69	cb_emit_display	760
7.17.2.70	cb_emit_divide	762
7.17.2.71	cb_emit_env_name	762
7.17.2.72	cb_emit_env_value	763
7.17.2.73	cb_emit_evaluate	763
7.17.2.74	cb_emit_exit	763
7.17.2.75	cb_emit_free	763
7.17.2.76	cb_emit_get_environment	764
7.17.2.77	cb_emit_goto	764
7.17.2.78	cb_emit_if	765
7.17.2.79	cb_emit_initialize	765
7.17.2.80	cb_emit_inspect	765
7.17.2.81	cb_emit_move	766
7.17.2.82	cb_emit_move_corresponding	766
7.17.2.83	cb_emit_open	767
7.17.2.84	cb_emit_perform	767
7.17.2.85	cb_emit_read	768
7.17.2.86	cb_emit_release	769
7.17.2.87	cb_emit_return	769
7.17.2.88	cb_emit_rewrite	770
7.17.2.89	cb_emit_rollback	770
7.17.2.90	cb_emit_search	771

---

7.17.2.91	cb_emit_search_all	771
7.17.2.92	cb_emit_set_false	771
7.17.2.93	cb_emit_set_on_off	772
7.17.2.94	cb_emit_set_to	772
7.17.2.95	cb_emit_set_true	774
7.17.2.96	cb_emit_set_up_down	774
7.17.2.97	cb_emit_setenv	775
7.17.2.98	cb_emit_sort_finish	775
7.17.2.99	cb_emit_sort_giving	775
7.17.2.100	cb_emit_sort_init	776
7.17.2.101	cb_emit_sort_input	777
7.17.2.102	cb_emit_sort_output	777
7.17.2.103	cb_emit_sort_using	777
7.17.2.104	cb_emit_start	777
7.17.2.105	cb_emit_stop_run	778
7.17.2.106	cb_emit_string	778
7.17.2.107	cb_emit_unlock	778
7.17.2.108	cb_emit_unstring	779
7.17.2.109	cb_emit_write	779
7.17.2.110	cb_encode_program_id	780
7.17.2.111	cb_init_tarrying	781
7.17.2.112	cb_validate_program_body	781
7.17.2.113	cb_validate_program_data	782
7.17.2.114	cb_validate_program_environment	784
7.17.2.115	validate_move	788
7.17.3	Variable Documentation	796
7.17.3.1	sending_id	796
7.17.3.2	suppress_warn	796
7.18	config.h File Reference	797
7.18.1	Define Documentation	798
7.18.1.1	__USE_STRING_INLINES	798
7.18.1.2	COB_EXEEXT	798
7.18.1.3	COB_EXPORT_DYN	799
7.18.1.4	COB_EXTRA_FLAGS	799

---

7.18.1.5	COB_HAS_INLINE	799
7.18.1.6	COB_LI_IS_LL	799
7.18.1.7	COB_PIC_FLAGS	799
7.18.1.8	COB_SHARED_OPT	799
7.18.1.9	COB_STRFTIME	799
7.18.1.10	COB_STRIP_CMD	799
7.18.1.11	ENABLE_NLS	799
7.18.1.12	HAVE_ALLOCA	799
7.18.1.13	HAVE_ALLOCA_H	800
7.18.1.14	HAVE_ATTRIBUTE_ALIGNED	800
7.18.1.15	HAVE_COLOR_SET	800
7.18.1.16	HAVE_DCGETTEXT	800
7.18.1.17	HAVE_DLFCN_H	800
7.18.1.18	HAVE_FCNTL	800
7.18.1.19	HAVE_FCNTL_H	800
7.18.1.20	HAVE_GETOPT_H	800
7.18.1.21	HAVE_GETTEXT	800
7.18.1.22	HAVE_GETTIMEOFDAY	800
7.18.1.23	HAVE_GMP_H	801
7.18.1.24	HAVE_ICONV	801
7.18.1.25	HAVE_INTTYPES_H	801
7.18.1.26	HAVE_LANGINFO_CODESET	801
7.18.1.27	HAVE_LIBGMP	801
7.18.1.28	HAVE_LIBNCURSES	801
7.18.1.29	HAVE_LOCALE_H	801
7.18.1.30	HAVE_MALLOC_H	801
7.18.1.31	HAVE_MEMMOVE	801
7.18.1.32	HAVE_MEMORY_H	801
7.18.1.33	HAVE_MEMSET	802
7.18.1.34	HAVE_NCURSES_H	802
7.18.1.35	HAVE_PSIGN_OPT	802
7.18.1.36	HAVE_SETLOCALE	802
7.18.1.37	HAVE_SIGNAL_H	802
7.18.1.38	HAVE_STDDEF_H	802



---

7.18.1.39 HAVE_STDINT_H . . . . .	802
7.18.1.40 HAVE_STDLIB_H . . . . .	802
7.18.1.41 HAVE_STRCASECMP . . . . .	802
7.18.1.42 HAVE_STRCHR . . . . .	802
7.18.1.43 HAVE_STRDUP . . . . .	803
7.18.1.44 HAVE_STRERROR . . . . .	803
7.18.1.45 HAVE_STRING_H . . . . .	803
7.18.1.46 HAVE_STRINGS_H . . . . .	803
7.18.1.47 HAVE_STRRCHR . . . . .	803
7.18.1.48 HAVE_STRSTR . . . . .	803
7.18.1.49 HAVE_STRTOL . . . . .	803
7.18.1.50 HAVE_SYS_STAT_H . . . . .	803
7.18.1.51 HAVE_SYS_TIME_H . . . . .	803
7.18.1.52 HAVE_SYS_TYPES_H . . . . .	803
7.18.1.53 HAVE_TIMEZONE . . . . .	804
7.18.1.54 HAVE_UNISTD_H . . . . .	804
7.18.1.55 HAVE_VPRINTF . . . . .	804
7.18.1.56 HAVE_WCHAR_H . . . . .	804
7.18.1.57 ICONV_CONST . . . . .	804
7.18.1.58 PACKAGE . . . . .	804
7.18.1.59 PACKAGE_BUGREPORT . . . . .	804
7.18.1.60 PACKAGE_NAME . . . . .	804
7.18.1.61 PACKAGE_STRING . . . . .	804
7.18.1.62 PACKAGE_TARNAME . . . . .	804
7.18.1.63 PACKAGE_VERSION . . . . .	805
7.18.1.64 PATCH_LEVEL . . . . .	805
7.18.1.65 STDC_HEADERS . . . . .	805
7.18.1.66 USE_DB41 . . . . .	805
7.18.1.67 USE_LIBDL . . . . .	805
7.18.1.68 VERSION . . . . .	805
7.18.1.69 WITH_DB . . . . .	805
7.18.1.70 WITH_VARSEQ . . . . .	805
7.19 cpucheck.c File Reference . . . . .	805
7.19.1 Function Documentation . . . . .	806

---

7.19.1.1	main	806
7.20	defaults.h File Reference	807
7.20.1	Define Documentation	808
7.20.1.1	COB_CC	808
7.20.1.2	COB_CFLAGS	808
7.20.1.3	COB_CONFIG_DIR	808
7.20.1.4	COB_COPY_DIR	808
7.20.1.5	COB_LDFLAGS	808
7.20.1.6	COB_LIBRARY_PATH	808
7.20.1.7	COB_LIBS	808
7.20.1.8	COB_MODULE_EXT	808
7.20.1.9	LOCALEDIR	809
7.21	lib/dummymac.c File Reference	809
7.21.1	Function Documentation	809
7.21.1.1	dummymacfix	809
7.22	lib/getopt.c File Reference	809
7.22.1	Define Documentation	811
7.22.1.1	_	811
7.22.1.2	_NO_PROTO	811
7.22.1.3	attribute_hidden	811
7.22.1.4	const	811
7.22.1.5	GETOPT_INTERFACE_VERSION	811
7.22.1.6	NOOPTION_P	811
7.22.1.7	SWAP_FLAGS	811
7.22.2	Enumeration Type Documentation	811
7.22.2.1	"@2	811
7.22.3	Function Documentation	812
7.22.3.1	_getopt_internal	812
7.22.3.2	getenv	823
7.22.3.3	getopt	823
7.22.4	Variable Documentation	823
7.22.4.1	attribute_hidden	823
7.22.4.2	optarg	823
7.22.4.3	opterr	823

---

7.22.4.4	optind	823
7.22.4.5	optopt	823
7.23	lib/getopt.h File Reference	824
7.23.1	Define Documentation	825
7.23.1.1	_GETOPT_H	825
7.23.1.2	no_argument	825
7.23.1.3	optional_argument	825
7.23.1.4	required_argument	825
7.23.2	Function Documentation	825
7.23.2.1	_getopt_internal	825
7.23.2.2	getopt	825
7.23.2.3	getopt_long	825
7.23.2.4	getopt_long_only	826
7.23.3	Variable Documentation	826
7.23.3.1	optarg	826
7.23.3.2	opterr	826
7.23.3.3	optind	826
7.23.3.4	optopt	826
7.24	lib/getopt1.c File Reference	826
7.24.1	Define Documentation	827
7.24.1.1	const	827
7.24.1.2	GETOPT_INTERFACE_VERSION	827
7.24.1.3	NULL	828
7.24.2	Function Documentation	828
7.24.2.1	getopt_long	828
7.24.2.2	getopt_long_only	828
7.25	lib/gettext.h File Reference	828
7.25.1	Define Documentation	830
7.25.1.1	_	830
7.25.1.2	_	830
7.25.1.3	_	830
7.25.1.4	_	830
7.25.1.5	_	830
7.25.1.6	_	830

---

7.25.1.7	_	830
7.25.1.8	_	830
7.25.1.9	bind_textdomain_codeset	830
7.25.1.10	bindtextdomain	830
7.25.1.11	dcgettext	830
7.25.1.12	dcngettext	830
7.25.1.13	dgettext	830
7.25.1.14	dngettext	830
7.25.1.15	gettext	831
7.25.1.16	gettext_noop	831
7.25.1.17	gettext_noop	831
7.25.1.18	gettext_noop	831
7.25.1.19	gettext_noop	831
7.25.1.20	gettext_noop	831
7.25.1.21	gettext_noop	831
7.25.1.22	gettext_noop	831
7.25.1.23	gettext_noop	831
7.25.1.24	N_	831
7.25.1.25	N_	831
7.25.1.26	N_	831
7.25.1.27	N_	831
7.25.1.28	N_	831
7.25.1.29	N_	831
7.25.1.30	N_	831
7.25.1.31	N_	831
7.25.1.32	ngettext	831
7.25.1.33	textdomain	832
7.26	libcob.h File Reference	832
7.27	libcob/byteswap.h File Reference	833
7.27.1	Define Documentation	833
7.27.1.1	COB_BSWAP_16	833
7.27.1.2	COB_BSWAP_16_CONSTANT	833
7.27.1.3	COB_BSWAP_32	833
7.27.1.4	COB_BSWAP_32_CONSTANT	834

---

7.27.1.5	COB_BSWAP_64	834
7.27.1.6	COB_BSWAP_64_CONSTANT	834
7.28	libcob/call.c File Reference	834
7.28.1	Define Documentation	836
7.28.1.1	__USE_GNU	836
7.28.1.2	CALL_BUFF_SIZE	836
7.28.1.3	CALL_FILEBUFF_SIZE	836
7.28.1.4	COB_MAX_COBCALL_PARMS	836
7.28.1.5	COB_SYSTEM_GEN	836
7.28.1.6	HASH_SIZE	836
7.28.1.7	lt_dlclose	837
7.28.1.8	lt_dLError	837
7.28.1.9	lt_dlhandle	837
7.28.1.10	lt_dlopen	837
7.28.1.11	lt_dlsym	837
7.28.1.12	PATHSEPC	837
7.28.1.13	PATHSEPS	837
7.28.2	Function Documentation	837
7.28.2.1	cob_call_error	837
7.28.2.2	cob_call_resolve	838
7.28.2.3	cob_call_resolve_1	838
7.28.2.4	cob_field_cancel	838
7.28.2.5	cob_init_call	838
7.28.2.6	cob_resolve	840
7.28.2.7	cob_resolve_1	842
7.28.2.8	cob_resolve_error	842
7.28.2.9	cob_set_cancel	842
7.28.2.10	cobcall	843
7.28.2.11	cobcancel	844
7.28.2.12	cobfunc	844
7.28.2.13	coblongjmp	844
7.28.2.14	cobsavenv	845
7.28.2.15	cobsavenv2	845
7.29	libcob/call.h File Reference	845

7.29.1	Define Documentation	847
7.29.1.1	cobsetjmp	847
7.29.2	Function Documentation	847
7.29.2.1	cob_call_error	847
7.29.2.2	cob_call_resolve	847
7.29.2.3	cob_call_resolve_1	848
7.29.2.4	cob_field_cancel	848
7.29.2.5	cob_resolve	848
7.29.2.6	cob_resolve_1	850
7.29.2.7	cob_resolve_error	851
7.29.2.8	cob_set_cancel	851
7.29.2.9	cobcall	851
7.29.2.10	cobcancel	852
7.29.2.11	cobfunc	852
7.29.2.12	coblongjmp	853
7.29.2.13	cobsavenv	853
7.29.2.14	cobsavenv2	853
7.30	libcob/coblocal.h File Reference	854
7.30.1	Define Documentation	855
7.30.1.1	COB_ATTR_INIT	855
7.30.1.2	COB_CHK_PARMS	855
7.30.1.3	COB_HIDDEN	855
7.30.2	Function Documentation	855
7.30.2.1	cob_exit_fileio	855
7.30.2.2	cob_field_to_string	856
7.30.2.3	cob_get_long_long	856
7.30.2.4	cob_init_call	856
7.30.2.5	cob_init_fileio	858
7.30.2.6	cob_init_intrinsic	859
7.30.2.7	cob_init_move	859
7.30.2.8	cob_init_numeric	859
7.30.2.9	cob_init_strings	860
7.30.2.10	cob_init_termio	860
7.30.2.11	cob_memcpy	860

---

7.30.2.12	<a href="#">cob_real_get_sign</a>	860
7.30.2.13	<a href="#">cob_real_put_sign</a>	860
7.30.2.14	<a href="#">cob_screen_set_mode</a>	860
7.30.2.15	<a href="#">cob_screen_terminate</a>	861
7.30.3	Variable Documentation	861
7.30.3.1	<a href="#">cob_got_exception</a>	861
7.30.3.2	<a href="#">cob_orig_line</a>	861
7.30.3.3	<a href="#">cob_orig_paragraph</a>	861
7.30.3.4	<a href="#">cob_orig_program_id</a>	861
7.30.3.5	<a href="#">cob_orig_section</a>	861
7.30.3.6	<a href="#">cob_orig_statement</a>	861
7.30.3.7	<a href="#">cob_screen_initialized</a>	861
7.31	libcob/codegen.h File Reference	862
7.31.1	Define Documentation	865
7.31.1.1	<a href="#">COB_STATIC</a>	865
7.31.2	Function Documentation	865
7.31.2.1	<a href="#">cob_add_align_s16_binary</a>	865
7.31.2.2	<a href="#">cob_add_align_s32_binary</a>	865
7.31.2.3	<a href="#">cob_add_align_s64_binary</a>	865
7.31.2.4	<a href="#">cob_add_align_u16_binary</a>	865
7.31.2.5	<a href="#">cob_add_align_u32_binary</a>	865
7.31.2.6	<a href="#">cob_add_align_u64_binary</a>	865
7.31.2.7	<a href="#">cob_add_packed_int</a>	865
7.31.2.8	<a href="#">cob_add_s16_binary</a>	865
7.31.2.9	<a href="#">cob_add_s24_binary</a>	865
7.31.2.10	<a href="#">cob_add_s32_binary</a>	865
7.31.2.11	<a href="#">cob_add_s40_binary</a>	865
7.31.2.12	<a href="#">cob_add_s48_binary</a>	866
7.31.2.13	<a href="#">cob_add_s56_binary</a>	866
7.31.2.14	<a href="#">cob_add_s64_binary</a>	866
7.31.2.15	<a href="#">cob_add_s8_binary</a>	866
7.31.2.16	<a href="#">cob_add_u16_binary</a>	866
7.31.2.17	<a href="#">cob_add_u24_binary</a>	866
7.31.2.18	<a href="#">cob_add_u32_binary</a>	866

---

7.31.2.19	cob_add_u40_binary	866
7.31.2.20	cob_add_u48_binary	866
7.31.2.21	cob_add_u56_binary	866
7.31.2.22	cob_add_u64_binary	866
7.31.2.23	cob_add_u8_binary	866
7.31.2.24	cob_addswp_s16_binary	866
7.31.2.25	cob_addswp_s24_binary	866
7.31.2.26	cob_addswp_s32_binary	866
7.31.2.27	cob_addswp_s40_binary	866
7.31.2.28	cob_addswp_s48_binary	866
7.31.2.29	cob_addswp_s56_binary	866
7.31.2.30	cob_addswp_s64_binary	866
7.31.2.31	cob_addswp_u16_binary	866
7.31.2.32	cob_addswp_u24_binary	866
7.31.2.33	cob_addswp_u32_binary	866
7.31.2.34	cob_addswp_u40_binary	866
7.31.2.35	cob_addswp_u48_binary	866
7.31.2.36	cob_addswp_u56_binary	867
7.31.2.37	cob_addswp_u64_binary	867
7.31.2.38	cob_cmp_align_s16_binary	867
7.31.2.39	cob_cmp_align_s32_binary	867
7.31.2.40	cob_cmp_align_s64_binary	867
7.31.2.41	cob_cmp_align_u16_binary	867
7.31.2.42	cob_cmp_align_u32_binary	867
7.31.2.43	cob_cmp_align_u64_binary	867
7.31.2.44	cob_cmp_packed_int	867
7.31.2.45	cob_cmp_s16_binary	867
7.31.2.46	cob_cmp_s24_binary	867
7.31.2.47	cob_cmp_s32_binary	867
7.31.2.48	cob_cmp_s40_binary	867
7.31.2.49	cob_cmp_s48_binary	867
7.31.2.50	cob_cmp_s56_binary	867
7.31.2.51	cob_cmp_s64_binary	867
7.31.2.52	cob_cmp_s8_binary	867



---

7.31.2.53	cob_cmp_u16_binary	867
7.31.2.54	cob_cmp_u24_binary	867
7.31.2.55	cob_cmp_u32_binary	867
7.31.2.56	cob_cmp_u40_binary	867
7.31.2.57	cob_cmp_u48_binary	867
7.31.2.58	cob_cmp_u56_binary	867
7.31.2.59	cob_cmp_u64_binary	867
7.31.2.60	cob_cmp_u8_binary	868
7.31.2.61	cob_cmpswp_align_s16_binary	868
7.31.2.62	cob_cmpswp_align_s32_binary	868
7.31.2.63	cob_cmpswp_align_s64_binary	868
7.31.2.64	cob_cmpswp_align_u16_binary	868
7.31.2.65	cob_cmpswp_align_u32_binary	868
7.31.2.66	cob_cmpswp_align_u64_binary	868
7.31.2.67	cob_cmpswp_s16_binary	868
7.31.2.68	cob_cmpswp_s24_binary	868
7.31.2.69	cob_cmpswp_s32_binary	868
7.31.2.70	cob_cmpswp_s40_binary	868
7.31.2.71	cob_cmpswp_s48_binary	868
7.31.2.72	cob_cmpswp_s56_binary	868
7.31.2.73	cob_cmpswp_s64_binary	868
7.31.2.74	cob_cmpswp_u16_binary	868
7.31.2.75	cob_cmpswp_u24_binary	868
7.31.2.76	cob_cmpswp_u32_binary	868
7.31.2.77	cob_cmpswp_u40_binary	868
7.31.2.78	cob_cmpswp_u48_binary	868
7.31.2.79	cob_cmpswp_u56_binary	868
7.31.2.80	cob_cmpswp_u64_binary	868
7.31.2.81	cob_get_numdisp	868
7.31.2.82	cob_get_packed_int	868
7.31.2.83	cob_setswp_s16_binary	868
7.31.2.84	cob_setswp_s24_binary	869
7.31.2.85	cob_setswp_s32_binary	869
7.31.2.86	cob_setswp_s40_binary	869

---

7.31.2.87	cob_setswp_s48_binary	869
7.31.2.88	cob_setswp_s56_binary	869
7.31.2.89	cob_setswp_s64_binary	869
7.31.2.90	cob_setswp_u16_binary	869
7.31.2.91	cob_setswp_u24_binary	869
7.31.2.92	cob_setswp_u32_binary	869
7.31.2.93	cob_setswp_u40_binary	869
7.31.2.94	cob_setswp_u48_binary	869
7.31.2.95	cob_setswp_u56_binary	869
7.31.2.96	cob_setswp_u64_binary	869
7.31.2.97	cob_sub_align_s16_binary	869
7.31.2.98	cob_sub_align_s32_binary	869
7.31.2.99	cob_sub_align_s64_binary	869
7.31.2.100	cob_sub_align_u16_binary	869
7.31.2.101	cob_sub_align_u32_binary	869
7.31.2.102	cob_sub_align_u64_binary	869
7.31.2.103	cob_sub_s16_binary	869
7.31.2.104	cob_sub_s24_binary	869
7.31.2.105	cob_sub_s32_binary	869
7.31.2.106	cob_sub_s40_binary	869
7.31.2.107	cob_sub_s48_binary	869
7.31.2.108	cob_sub_s56_binary	870
7.31.2.109	cob_sub_s64_binary	870
7.31.2.110	cob_sub_s8_binary	870
7.31.2.111	cob_sub_u16_binary	870
7.31.2.112	cob_sub_u24_binary	870
7.31.2.113	cob_sub_u32_binary	870
7.31.2.114	cob_sub_u40_binary	870
7.31.2.115	cob_sub_u48_binary	870
7.31.2.116	cob_sub_u56_binary	870
7.31.2.117	cob_sub_u64_binary	870
7.31.2.118	cob_sub_u8_binary	870
7.31.2.119	cob_subswp_s16_binary	870
7.31.2.120	cob_subswp_s24_binary	870

---

7.31.2.121	cob_subswp_s32_binary	870
7.31.2.122	cob_subswp_s40_binary	870
7.31.2.123	cob_subswp_s48_binary	870
7.31.2.124	cob_subswp_s56_binary	870
7.31.2.125	cob_subswp_s64_binary	870
7.31.2.126	cob_subswp_u16_binary	870
7.31.2.127	cob_subswp_u24_binary	870
7.31.2.128	cob_subswp_u32_binary	870
7.31.2.129	cob_subswp_u40_binary	870
7.31.2.130	cob_subswp_u48_binary	870
7.31.2.131	cob_subswp_u56_binary	870
7.31.2.132	cob_subswp_u64_binary	871
7.32	libcob/common.c File Reference	871
7.32.1	Define Documentation	872
7.32.1.1	COB_ERRBUF_SIZE	872
7.32.1.2	COB_EXCEPTION	873
7.32.1.3	COB_EXCEPTION	873
7.32.1.4	EXCEPTION_TAB_SIZE	873
7.32.2	Typedef Documentation	873
7.32.2.1	cob_sighandler_t	873
7.32.3	Variable Documentation	873
7.32.3.1	cob_call_params	873
7.32.3.2	cob_current_module	873
7.32.3.3	cob_exception_code	873
7.32.3.4	cob_got_exception	873
7.32.3.5	cob_high	873
7.32.3.6	cob_initial_external	874
7.32.3.7	cob_initialized	874
7.32.3.8	cob_low	874
7.32.3.9	cob_one	874
7.32.3.10	cob_orig_line	874
7.32.3.11	cob_orig_paragraph	874
7.32.3.12	cob_orig_program_id	874
7.32.3.13	cob_orig_section	874

---

7.32.3.14	<code>cob_orig_statement</code>	874
7.32.3.15	<code>cob_quote</code>	874
7.32.3.16	<code>cob_save_call_params</code>	875
7.32.3.17	<code>cob_space</code>	875
7.32.3.18	<code>cob_zero</code>	875
7.33	<code>libcob/common.h</code> File Reference	875
7.33.1	Define Documentation	880
7.33.1.1	<code>cob_d2i</code>	880
7.33.1.2	<code>COB_DISPLAY_SIGN_ASCII</code>	880
7.33.1.3	<code>COB_DISPLAY_SIGN_EBCDIC</code>	880
7.33.1.4	<code>COB_EXCEPTION</code>	880
7.33.1.5	<code>COB_FERROR_CHAINING</code>	880
7.33.1.6	<code>COB_FERROR_CODEGEN</code>	880
7.33.1.7	<code>COB_FERROR_INITIALIZED</code>	880
7.33.1.8	<code>COB_FERROR_STACK</code>	881
7.33.1.9	<code>COB_FIELD_BINARY_SWAP</code>	881
7.33.1.10	<code>COB_FIELD_BLANK_ZERO</code>	881
7.33.1.11	<code>COB_FIELD_DATA</code>	881
7.33.1.12	<code>COB_FIELD_DIGITS</code>	881
7.33.1.13	<code>COB_FIELD_FLAGS</code>	881
7.33.1.14	<code>COB_FIELD_HAVE_SIGN</code>	881
7.33.1.15	<code>COB_FIELD_IS_NUMERIC</code>	881
7.33.1.16	<code>COB_FIELD_IS_POINTER</code>	881
7.33.1.17	<code>COB_FIELD_JUSTIFIED</code>	881
7.33.1.18	<code>COB_FIELD_PIC</code>	882
7.33.1.19	<code>COB_FIELD_REAL_BINARY</code>	882
7.33.1.20	<code>COB_FIELD_SCALE</code>	882
7.33.1.21	<code>COB_FIELD_SIGN_LEADING</code>	882
7.33.1.22	<code>COB_FIELD_SIGN_SEPARATE</code>	882
7.33.1.23	<code>COB_FIELD_SIZE</code>	882
7.33.1.24	<code>COB_FIELD_TYPE</code>	882
7.33.1.25	<code>COB_FLAG_BINARY_SWAP</code>	882
7.33.1.26	<code>COB_FLAG_BLANK_ZERO</code>	882
7.33.1.27	<code>COB_FLAG_HAVE_SIGN</code>	882

---

7.33.1.28	COB_FLAG_IS_POINTER	883
7.33.1.29	COB_FLAG_JUSTIFIED	883
7.33.1.30	COB_FLAG_REAL_BINARY	883
7.33.1.31	COB_FLAG_SIGN_LEADING	883
7.33.1.32	COB_FLAG_SIGN_SEPARATE	883
7.33.1.33	cob_get_sign	883
7.33.1.34	cob_i2d	883
7.33.1.35	COB_INLINE	883
7.33.1.36	COB_LARGE_BUFF	883
7.33.1.37	COB_LARGE_MAX	883
7.33.1.38	COB_MAX_FIELD_PARAMS	884
7.33.1.39	COB_MEDIUM_BUFF	884
7.33.1.40	COB_MEDIUM_MAX	884
7.33.1.41	COB_MINI_BUFF	884
7.33.1.42	COB_MINI_MAX	884
7.33.1.43	COB_NOINLINE	884
7.33.1.44	COB_NORMAL_BUFF	884
7.33.1.45	COB_NORMAL_MAX	884
7.33.1.46	cob_put_sign	884
7.33.1.47	COB_SMALL_BUFF	884
7.33.1.48	COB_SMALL_MAX	885
7.33.1.49	COB_STACK_SIZE	885
7.33.1.50	COB_TYPE_ALPHANUMERIC	885
7.33.1.51	COB_TYPE_ALPHANUMERIC_ALL	885
7.33.1.52	COB_TYPE_ALPHANUMERIC_EDITED	885
7.33.1.53	COB_TYPE_BOOLEAN	885
7.33.1.54	COB_TYPE_GROUP	885
7.33.1.55	COB_TYPE_NATIONAL	885
7.33.1.56	COB_TYPE_NATIONAL_EDITED	885
7.33.1.57	COB_TYPE_NUMERIC	885
7.33.1.58	COB_TYPE_NUMERIC_BINARY	886
7.33.1.59	COB_TYPE_NUMERIC_DISPLAY	886
7.33.1.60	COB_TYPE_NUMERIC_DOUBLE	886
7.33.1.61	COB_TYPE_NUMERIC_EDITED	886

---

7.33.1.62	COB_TYPE_NUMERIC_FLOAT	886
7.33.1.63	COB_TYPE_NUMERIC_PACKED	886
7.33.1.64	COB_TYPE_UNKNOWN	886
7.33.1.65	DLL_EXPIMP	886
7.33.1.66	GET_SIGN_ASCII	886
7.33.1.67	likely	886
7.33.1.68	PUT_SIGN_ASCII	887
7.33.1.69	unlikely	887
7.33.2	Typedef Documentation	887
7.33.2.1	ucharptr	887
7.33.3	Enumeration Type Documentation	887
7.33.3.1	cob_exception_id	887
7.33.4	Function Documentation	887
7.33.4.1	CBL_AND	887
7.33.4.2	CBL_EQ	887
7.33.4.3	CBL_ERROR_PROC	887
7.33.4.4	CBL_EXIT_PROC	887
7.33.4.5	CBL_IMP	887
7.33.4.6	CBL_NIMP	887
7.33.4.7	CBL_NOR	887
7.33.4.8	CBL_NOT	888
7.33.4.9	CBL_OC_NANOSLEEP	888
7.33.4.10	CBL_OR	888
7.33.4.11	CBL_TOLOWER	888
7.33.4.12	CBL_TOUPPER	888
7.33.4.13	CBL_X91	888
7.33.4.14	CBL_XF4	888
7.33.4.15	CBL_XF5	888
7.33.4.16	CBL_XOR	888
7.33.4.17	cob_accept_arg_number	888
7.33.4.18	cob_accept_arg_value	888
7.33.4.19	cob_accept_command_line	888
7.33.4.20	cob_accept_date	888
7.33.4.21	cob_accept_date_yyyymmdd	888

---

7.33.4.22	<a href="#">cob_accept_day</a>	888
7.33.4.23	<a href="#">cob_accept_day_of_week</a>	888
7.33.4.24	<a href="#">cob_accept_day_yyyyddd</a>	888
7.33.4.25	<a href="#">cob_accept_environment</a>	888
7.33.4.26	<a href="#">cob_accept_time</a>	888
7.33.4.27	<a href="#">cob_acuw_justify</a>	888
7.33.4.28	<a href="#">cob_acuw_sleep</a>	888
7.33.4.29	<a href="#">cob_allocate</a>	888
7.33.4.30	<a href="#">cob_chain_setup</a>	888
7.33.4.31	<a href="#">cob_check_based</a>	888
7.33.4.32	<a href="#">cob_check_numeric</a>	889
7.33.4.33	<a href="#">cob_check_odo</a>	889
7.33.4.34	<a href="#">cob_check_ref_mod</a>	889
7.33.4.35	<a href="#">cob_check_subscript</a>	889
7.33.4.36	<a href="#">cob_check_version</a>	889
7.33.4.37	<a href="#">cob_cmp</a>	889
7.33.4.38	<a href="#">cob_display_arg_number</a>	889
7.33.4.39	<a href="#">cob_display_command_line</a>	889
7.33.4.40	<a href="#">cob_display_env_value</a>	889
7.33.4.41	<a href="#">cob_display_environment</a>	889
7.33.4.42	<a href="#">cob_external_addr</a>	889
7.33.4.43	<a href="#">cob_fatal_error</a>	889
7.33.4.44	<a href="#">cob_free_alloc</a>	889
7.33.4.45	<a href="#">cob_get_environment</a>	889
7.33.4.46	<a href="#">cob_get_exception_name</a>	889
7.33.4.47	<a href="#">cob_get_pointer</a>	889
7.33.4.48	<a href="#">cob_get_prog_pointer</a>	889
7.33.4.49	<a href="#">cob_get_switch</a>	889
7.33.4.50	<a href="#">cob_init</a>	889
7.33.4.51	<a href="#">cob_is_alpha</a>	889
7.33.4.52	<a href="#">cob_is_lower</a>	889
7.33.4.53	<a href="#">cob_is_numeric</a>	889
7.33.4.54	<a href="#">cob_is_omitted</a>	889
7.33.4.55	<a href="#">cob_is_upper</a>	889

7.33.4.56	cob_malloc	890
7.33.4.57	cob_module_enter	890
7.33.4.58	cob_module_leave	890
7.33.4.59	cob_numeric_cmp	890
7.33.4.60	cob_parameter_size	890
7.33.4.61	cob_ready_trace	890
7.33.4.62	cob_reset_trace	890
7.33.4.63	cob_return_args	890
7.33.4.64	cob_runtime_error	890
7.33.4.65	cob_set_environment	890
7.33.4.66	cob_set_exception	890
7.33.4.67	cob_set_location	890
7.33.4.68	cob_set_switch	890
7.33.4.69	cob_stop_run	890
7.33.4.70	cob_table_sort	890
7.33.4.71	cob_table_sort_init	890
7.33.4.72	cob_table_sort_init_key	890
7.33.4.73	cobcommandline	890
7.33.4.74	cobexit	890
7.33.4.75	cobgetenv	891
7.33.4.76	cobinit	891
7.33.4.77	cobputenv	891
7.33.4.78	cobtidy	891
7.33.4.79	SYSTEM	891
7.33.5	Variable Documentation	891
7.33.5.1	cob_call_params	891
7.33.5.2	cob_current_module	891
7.33.5.3	cob_exception_code	891
7.33.5.4	cob_high	891
7.33.5.5	cob_initial_external	891
7.33.5.6	cob_initialized	891
7.33.5.7	cob_low	891
7.33.5.8	cob_one	891
7.33.5.9	cob_quote	892



---

7.33.5.10	cob_save_call_params	892
7.33.5.11	cob_space	892
7.33.5.12	cob_zero	892
7.34	libcob/fileio.c File Reference	892
7.34.1	Define Documentation	895
7.34.1.1	_FILE_OFFSET_BITS	895
7.34.1.2	_LARGEFILE64_SOURCE	895
7.34.1.3	_LFS64_LARGEFILE	895
7.34.1.4	_LFS64_STDIO	895
7.34.1.5	cob_dbtsize_t	895
7.34.1.6	COBSORTABORT	895
7.34.1.7	COBSORTEND	895
7.34.1.8	COBSORTFILEERR	895
7.34.1.9	COBSORTNOTOPEN	895
7.34.1.10	DB_CLOSE	896
7.34.1.11	DB_DEL	896
7.34.1.12	DB_GET	896
7.34.1.13	DB_PUT	896
7.34.1.14	DB_SEQ	896
7.34.1.15	DB_SYNC	896
7.34.1.16	DBT_SET	896
7.34.1.17	INITIAL_FLAGS	896
7.34.1.18	NUM_PREFIX	896
7.34.1.19	O_BINARY	896
7.34.1.20	O_LARGEFILE	897
7.34.1.21	RETURN_STATUS	897
7.34.1.22	SEEK_INIT	897
7.34.2	Function Documentation	897
7.34.2.1	CBL_CHANGE_DIR	897
7.34.2.2	CBL_CHECK_FILE_EXIST	897
7.34.2.3	CBL_CLOSE_FILE	898
7.34.2.4	CBL_COPY_FILE	898
7.34.2.5	CBL_CREATE_DIR	899
7.34.2.6	CBL_CREATE_FILE	900

---

7.34.2.7	CBL_DELETE_DIR	900
7.34.2.8	CBL_DELETE_FILE	900
7.34.2.9	CBL_FLUSH_FILE	901
7.34.2.10	CBL_GET_CURRENT_DIR	901
7.34.2.11	CBL_OPEN_FILE	902
7.34.2.12	CBL_READ_FILE	902
7.34.2.13	CBL_RENAME_FILE	903
7.34.2.14	CBL_WRITE_FILE	903
7.34.2.15	cob_acuw_chdir	904
7.34.2.16	cob_acuw_copyfile	904
7.34.2.17	cob_acuw_file_delete	905
7.34.2.18	cob_acuw_file_info	905
7.34.2.19	cob_acuw_mkdir	906
7.34.2.20	cob_close	906
7.34.2.21	cob_commit	907
7.34.2.22	cob_default_error_handle	907
7.34.2.23	cob_delete	909
7.34.2.24	cob_exit_fileio	909
7.34.2.25	cob_file_release	910
7.34.2.26	cob_file_return	910
7.34.2.27	cob_file_sort_close	911
7.34.2.28	cob_file_sort_giving	911
7.34.2.29	cob_file_sort_init	912
7.34.2.30	cob_file_sort_init_key	913
7.34.2.31	cob_file_sort_using	913
7.34.2.32	cob_init_fileio	913
7.34.2.33	cob_open	914
7.34.2.34	cob_read	919
7.34.2.35	cob_rewrite	920
7.34.2.36	cob_rollback	921
7.34.2.37	cob_start	921
7.34.2.38	cob_unlock_file	922
7.34.2.39	cob_write	922
7.34.3	Variable Documentation	923

---

---

7.34.3.1	cob_error_file	923
7.35	libcob/fileio.h File Reference	923
7.35.1	Define Documentation	927
7.35.1.1	COB_ACCESS_DYNAMIC	927
7.35.1.2	COB_ACCESS_RANDOM	927
7.35.1.3	COB_ACCESS_SEQUENTIAL	927
7.35.1.4	COB_ASCENDING	927
7.35.1.5	COB_CLOSE_LOCK	927
7.35.1.6	COB_CLOSE_NO_REWIND	927
7.35.1.7	COB_CLOSE_NORMAL	927
7.35.1.8	COB_CLOSE_UNIT	927
7.35.1.9	COB_CLOSE_UNIT_REMOVAL	928
7.35.1.10	COB_DESCENDING	928
7.35.1.11	COB_EQ	928
7.35.1.12	COB_FILE_MODE	928
7.35.1.13	COB_FILE_VERSION	928
7.35.1.14	COB_GE	928
7.35.1.15	COB_GT	928
7.35.1.16	COB_LE	928
7.35.1.17	COB_LINAGE_INVALID	928
7.35.1.18	COB_LOCK_AUTOMATIC	928
7.35.1.19	COB_LOCK_EXCLUSIVE	929
7.35.1.20	COB_LOCK_MANUAL	929
7.35.1.21	COB_LOCK_MASK	929
7.35.1.22	COB_LOCK_MULTIPLE	929
7.35.1.23	COB_LT	929
7.35.1.24	COB_NE	929
7.35.1.25	COB_NOT_CONFIGURED	929
7.35.1.26	COB_OPEN_CLOSED	929
7.35.1.27	COB_OPEN_EXTEND	929
7.35.1.28	COB_OPEN_I_O	929
7.35.1.29	COB_OPEN_INPUT	930
7.35.1.30	COB_OPEN_LOCKED	930
7.35.1.31	COB_OPEN_OUTPUT	930

---

7.35.1.32 COB_ORG_INDEXED . . . . .	930
7.35.1.33 COB_ORG_LINE_SEQUENTIAL . . . . .	930
7.35.1.34 COB_ORG_MAX . . . . .	930
7.35.1.35 COB_ORG_RELATIVE . . . . .	930
7.35.1.36 COB_ORG_SEQUENTIAL . . . . .	930
7.35.1.37 COB_ORG_SORT . . . . .	930
7.35.1.38 COB_READ_FIRST . . . . .	930
7.35.1.39 COB_READ_IGNORE_LOCK . . . . .	931
7.35.1.40 COB_READ_KEPT_LOCK . . . . .	931
7.35.1.41 COB_READ_LAST . . . . .	931
7.35.1.42 COB_READ_LOCK . . . . .	931
7.35.1.43 COB_READ_NEXT . . . . .	931
7.35.1.44 COB_READ_NO_LOCK . . . . .	931
7.35.1.45 COB_READ_PREVIOUS . . . . .	931
7.35.1.46 COB_READ_WAIT_LOCK . . . . .	931
7.35.1.47 COB_SELECT_EXTERNAL . . . . .	931
7.35.1.48 COB_SELECT_FILE_STATUS . . . . .	931
7.35.1.49 COB_SELECT_LINAGE . . . . .	932
7.35.1.50 COB_SELECT_SPLITKEY . . . . .	932
7.35.1.51 COB_STATUS_00_SUCCESS . . . . .	932
7.35.1.52 COB_STATUS_02_SUCCESS_DUPLICATE . . . . .	932
7.35.1.53 COB_STATUS_04_SUCCESS_INCOMPLETE . . . . .	932
7.35.1.54 COB_STATUS_05_SUCCESS_OPTIONAL . . . . .	932
7.35.1.55 COB_STATUS_07_SUCCESS_NO_UNIT . . . . .	932
7.35.1.56 COB_STATUS_10_END_OF_FILE . . . . .	932
7.35.1.57 COB_STATUS_14_OUT_OF_KEY_RANGE . . . . .	932
7.35.1.58 COB_STATUS_21_KEY_INVALID . . . . .	932
7.35.1.59 COB_STATUS_22_KEY_EXISTS . . . . .	933
7.35.1.60 COB_STATUS_23_KEY_NOT_EXISTS . . . . .	933
7.35.1.61 COB_STATUS_30_PERMANENT_ERROR . . . . .	933
7.35.1.62 COB_STATUS_31_INCONSISTENT_FILENAME . . . . .	933
7.35.1.63 COB_STATUS_34_BOUNDARY_VIOLATION . . . . .	933
7.35.1.64 COB_STATUS_35_NOT_EXISTS . . . . .	933
7.35.1.65 COB_STATUS_37_PERMISSION_DENIED . . . . .	933

---

7.35.1.66	COB_STATUS_38_CLOSED_WITH_LOCK	933
7.35.1.67	COB_STATUS_39_CONFLICT_ATTRIBUTE	933
7.35.1.68	COB_STATUS_41_ALREADY_OPEN	933
7.35.1.69	COB_STATUS_42_NOT_OPEN	934
7.35.1.70	COB_STATUS_43_READ_NOT_DONE	934
7.35.1.71	COB_STATUS_44_RECORD_OVERFLOW	934
7.35.1.72	COB_STATUS_46_READ_ERROR	934
7.35.1.73	COB_STATUS_47_INPUT_DENIED	934
7.35.1.74	COB_STATUS_48_OUTPUT_DENIED	934
7.35.1.75	COB_STATUS_49_I_O_DENIED	934
7.35.1.76	COB_STATUS_51_RECORD_LOCKED	934
7.35.1.77	COB_STATUS_52_EOP	934
7.35.1.78	COB_STATUS_57_I_O_LINAGE	934
7.35.1.79	COB_STATUS_61_FILE_SHARING	935
7.35.1.80	COB_STATUS_91_NOT_AVAILABLE	935
7.35.1.81	COB_WRITE_AFTER	935
7.35.1.82	COB_WRITE_BEFORE	935
7.35.1.83	COB_WRITE_CHANNEL	935
7.35.1.84	COB_WRITE_EOP	935
7.35.1.85	COB_WRITE_LINES	935
7.35.1.86	COB_WRITE_LOCK	935
7.35.1.87	COB_WRITE_MASK	935
7.35.1.88	COB_WRITE_PAGE	935
7.35.2	Function Documentation	936
7.35.2.1	CBL_CHANGE_DIR	936
7.35.2.2	CBL_CHECK_FILE_EXIST	936
7.35.2.3	CBL_CLOSE_FILE	937
7.35.2.4	CBL_COPY_FILE	937
7.35.2.5	CBL_CREATE_DIR	938
7.35.2.6	CBL_CREATE_FILE	939
7.35.2.7	CBL_DELETE_DIR	939
7.35.2.8	CBL_DELETE_FILE	939
7.35.2.9	CBL_FLUSH_FILE	940
7.35.2.10	CBL_GET_CURRENT_DIR	940

7.35.2.11 CBL_OPEN_FILE . . . . .	940
7.35.2.12 CBL_READ_FILE . . . . .	941
7.35.2.13 CBL_RENAME_FILE . . . . .	941
7.35.2.14 CBL_WRITE_FILE . . . . .	942
7.35.2.15 cob_acuw_chdir . . . . .	943
7.35.2.16 cob_acuw_copyfile . . . . .	943
7.35.2.17 cob_acuw_file_delete . . . . .	943
7.35.2.18 cob_acuw_file_info . . . . .	944
7.35.2.19 cob_acuw_mkdir . . . . .	945
7.35.2.20 cob_close . . . . .	945
7.35.2.21 cob_commit . . . . .	946
7.35.2.22 cob_default_error_handle . . . . .	946
7.35.2.23 cob_delete . . . . .	947
7.35.2.24 cob_file_release . . . . .	948
7.35.2.25 cob_file_return . . . . .	948
7.35.2.26 cob_file_sort_close . . . . .	949
7.35.2.27 cob_file_sort_giving . . . . .	949
7.35.2.28 cob_file_sort_init . . . . .	950
7.35.2.29 cob_file_sort_init_key . . . . .	951
7.35.2.30 cob_file_sort_using . . . . .	951
7.35.2.31 cob_open . . . . .	951
7.35.2.32 cob_read . . . . .	956
7.35.2.33 cob_rewrite . . . . .	957
7.35.2.34 cob_rollback . . . . .	958
7.35.2.35 cob_start . . . . .	958
7.35.2.36 cob_unlock_file . . . . .	959
7.35.2.37 cob_write . . . . .	959
7.35.3 Variable Documentation . . . . .	960
7.35.3.1 cob_error_file . . . . .	960
7.36 libcob/intrinsic.c File Reference . . . . .	960
7.36.1 Define Documentation . . . . .	963
7.36.1.1 COB_FIELD_INIT . . . . .	963
7.36.1.2 DEPTH_LEVEL . . . . .	963
7.36.2 Function Documentation . . . . .	963

---

7.36.2.1	<a href="#">cob_init_intrinsic</a>	963
7.36.2.2	<a href="#">cob_intr_abs</a>	964
7.36.2.3	<a href="#">cob_intr_acos</a>	964
7.36.2.4	<a href="#">cob_intr_annuity</a>	964
7.36.2.5	<a href="#">cob_intr_asin</a>	965
7.36.2.6	<a href="#">cob_intr_atan</a>	966
7.36.2.7	<a href="#">cob_intr_binop</a>	966
7.36.2.8	<a href="#">cob_intr_char</a>	967
7.36.2.9	<a href="#">cob_intr_combined_datetime</a>	968
7.36.2.10	<a href="#">cob_intr_concatenate</a>	968
7.36.2.11	<a href="#">cob_intr_cos</a>	969
7.36.2.12	<a href="#">cob_intr_current_date</a>	970
7.36.2.13	<a href="#">cob_intr_date_of_integer</a>	971
7.36.2.14	<a href="#">cob_intr_date_to_yyyymmdd</a>	972
7.36.2.15	<a href="#">cob_intr_day_of_integer</a>	973
7.36.2.16	<a href="#">cob_intr_day_to_yyyydd</a>	974
7.36.2.17	<a href="#">cob_intr_exception_file</a>	975
7.36.2.18	<a href="#">cob_intr_exception_location</a>	976
7.36.2.19	<a href="#">cob_intr_exception_statement</a>	976
7.36.2.20	<a href="#">cob_intr_exception_status</a>	977
7.36.2.21	<a href="#">cob_intr_exp</a>	977
7.36.2.22	<a href="#">cob_intr_exp10</a>	977
7.36.2.23	<a href="#">cob_intr_factorial</a>	978
7.36.2.24	<a href="#">cob_intr_fraction_part</a>	978
7.36.2.25	<a href="#">cob_intr_integer</a>	979
7.36.2.26	<a href="#">cob_intr_integer_of_date</a>	979
7.36.2.27	<a href="#">cob_intr_integer_of_day</a>	980
7.36.2.28	<a href="#">cob_intr_integer_part</a>	981
7.36.2.29	<a href="#">cob_intr_lcl_time_from_secs</a>	982
7.36.2.30	<a href="#">cob_intr_length</a>	984
7.36.2.31	<a href="#">cob_intr_locale_date</a>	984
7.36.2.32	<a href="#">cob_intr_locale_time</a>	986
7.36.2.33	<a href="#">cob_intr_log</a>	989
7.36.2.34	<a href="#">cob_intr_log10</a>	989

---

7.36.2.35	<code>cob_intr_lower_case</code>	989
7.36.2.36	<code>cob_intr_max</code>	990
7.36.2.37	<code>cob_intr_mean</code>	990
7.36.2.38	<code>cob_intr_median</code>	991
7.36.2.39	<code>cob_intr_midrange</code>	992
7.36.2.40	<code>cob_intr_min</code>	992
7.36.2.41	<code>cob_intr_mod</code>	993
7.36.2.42	<code>cob_intr_numval</code>	993
7.36.2.43	<code>cob_intr_numval_c</code>	995
7.36.2.44	<code>cob_intr_ord</code>	997
7.36.2.45	<code>cob_intr_ord_max</code>	997
7.36.2.46	<code>cob_intr_ord_min</code>	997
7.36.2.47	<code>cob_intr_present_value</code>	998
7.36.2.48	<code>cob_intr_random</code>	999
7.36.2.49	<code>cob_intr_range</code>	1000
7.36.2.50	<code>cob_intr_rem</code>	1001
7.36.2.51	<code>cob_intr_reverse</code>	1001
7.36.2.52	<code>cob_intr_seconds_from_formatted_time</code>	1001
7.36.2.53	<code>cob_intr_seconds_past_midnight</code>	1003
7.36.2.54	<code>cob_intr_sign</code>	1003
7.36.2.55	<code>cob_intr_sin</code>	1003
7.36.2.56	<code>cob_intr_sqrt</code>	1004
7.36.2.57	<code>cob_intr_standard_deviation</code>	1004
7.36.2.58	<code>cob_intr_stored_char_length</code>	1006
7.36.2.59	<code>cob_intr_substitute</code>	1006
7.36.2.60	<code>cob_intr_substitute_case</code>	1008
7.36.2.61	<code>cob_intr_sum</code>	1009
7.36.2.62	<code>cob_intr_tan</code>	1010
7.36.2.63	<code>cob_intr_test_date_yyyymmdd</code>	1010
7.36.2.64	<code>cob_intr_test_day_yyyyddd</code>	1011
7.36.2.65	<code>cob_intr_trim</code>	1012
7.36.2.66	<code>cob_intr_upper_case</code>	1013
7.36.2.67	<code>cob_intr_variance</code>	1013
7.36.2.68	<code>cob_intr_when_compiled</code>	1014



---

7.36.2.69	<a href="#">cob_intr_year_to_yyyy</a>	1014
7.36.3	<a href="#">Variable Documentation</a>	1015
7.36.3.1	<a href="#">d2</a>	1016
7.36.3.2	<a href="#">d3</a>	1016
7.36.3.3	<a href="#">d4</a>	1016
7.36.3.4	<a href="#">d5</a>	1016
7.37	<a href="#">libcob/intrinsic.h File Reference</a>	1016
7.37.1	<a href="#">Function Documentation</a>	1018
7.37.1.1	<a href="#">cob_intr_abs</a>	1018
7.37.1.2	<a href="#">cob_intr_acos</a>	1019
7.37.1.3	<a href="#">cob_intr_annuity</a>	1019
7.37.1.4	<a href="#">cob_intr_asin</a>	1020
7.37.1.5	<a href="#">cob_intr_atan</a>	1020
7.37.1.6	<a href="#">cob_intr_binop</a>	1021
7.37.1.7	<a href="#">cob_intr_char</a>	1022
7.37.1.8	<a href="#">cob_intr_combined_datetime</a>	1023
7.37.1.9	<a href="#">cob_intr_concatenate</a>	1023
7.37.1.10	<a href="#">cob_intr_cos</a>	1024
7.37.1.11	<a href="#">cob_intr_current_date</a>	1024
7.37.1.12	<a href="#">cob_intr_date_of_integer</a>	1026
7.37.1.13	<a href="#">cob_intr_date_to_yyyymmdd</a>	1027
7.37.1.14	<a href="#">cob_intr_day_of_integer</a>	1028
7.37.1.15	<a href="#">cob_intr_day_to_yyyddd</a>	1029
7.37.1.16	<a href="#">cob_intr_exception_file</a>	1030
7.37.1.17	<a href="#">cob_intr_exception_location</a>	1030
7.37.1.18	<a href="#">cob_intr_exception_statement</a>	1031
7.37.1.19	<a href="#">cob_intr_exception_status</a>	1032
7.37.1.20	<a href="#">cob_intr_exp</a>	1032
7.37.1.21	<a href="#">cob_intr_exp10</a>	1032
7.37.1.22	<a href="#">cob_intr_factorial</a>	1033
7.37.1.23	<a href="#">cob_intr_fraction_part</a>	1033
7.37.1.24	<a href="#">cob_intr_integer</a>	1034
7.37.1.25	<a href="#">cob_intr_integer_of_date</a>	1034
7.37.1.26	<a href="#">cob_intr_integer_of_day</a>	1035

---

7.37.1.27	<code>cob_intr_integer_part</code>	1036
7.37.1.28	<code>cob_intr_lcl_time_from_secs</code>	1036
7.37.1.29	<code>cob_intr_length</code>	1038
7.37.1.30	<code>cob_intr_locale_date</code>	1039
7.37.1.31	<code>cob_intr_locale_time</code>	1041
7.37.1.32	<code>cob_intr_log</code>	1043
7.37.1.33	<code>cob_intr_log10</code>	1044
7.37.1.34	<code>cob_intr_lower_case</code>	1044
7.37.1.35	<code>cob_intr_max</code>	1045
7.37.1.36	<code>cob_intr_mean</code>	1045
7.37.1.37	<code>cob_intr_median</code>	1046
7.37.1.38	<code>cob_intr_midrange</code>	1047
7.37.1.39	<code>cob_intr_min</code>	1047
7.37.1.40	<code>cob_intr_mod</code>	1048
7.37.1.41	<code>cob_intr_numval</code>	1048
7.37.1.42	<code>cob_intr_numval_c</code>	1049
7.37.1.43	<code>cob_intr_ord</code>	1051
7.37.1.44	<code>cob_intr_ord_max</code>	1051
7.37.1.45	<code>cob_intr_ord_min</code>	1052
7.37.1.46	<code>cob_intr_present_value</code>	1053
7.37.1.47	<code>cob_intr_random</code>	1054
7.37.1.48	<code>cob_intr_range</code>	1054
7.37.1.49	<code>cob_intr_rem</code>	1055
7.37.1.50	<code>cob_intr_reverse</code>	1056
7.37.1.51	<code>cob_intr_seconds_from_formatted_time</code>	1056
7.37.1.52	<code>cob_intr_seconds_past_midnight</code>	1057
7.37.1.53	<code>cob_intr_sign</code>	1058
7.37.1.54	<code>cob_intr_sin</code>	1058
7.37.1.55	<code>cob_intr_sqrt</code>	1059
7.37.1.56	<code>cob_intr_standard_deviation</code>	1059
7.37.1.57	<code>cob_intr_stored_char_length</code>	1060
7.37.1.58	<code>cob_intr_substitute</code>	1061
7.37.1.59	<code>cob_intr_substitute_case</code>	1062
7.37.1.60	<code>cob_intr_sum</code>	1064

7.37.1.61	cob_intr_tan	1065
7.37.1.62	cob_intr_test_date_yyyymmdd	1065
7.37.1.63	cob_intr_test_day_yyyydd	1066
7.37.1.64	cob_intr_trim	1066
7.37.1.65	cob_intr_upper_case	1067
7.37.1.66	cob_intr_variance	1067
7.37.1.67	cob_intr_when_compiled	1069
7.37.1.68	cob_intr_year_to_yyyy	1069
7.38	libcob/move.c File Reference	1070
7.38.1	Function Documentation	1071
7.38.1.1	cob_get_int	1071
7.38.1.2	cob_get_long_long	1072
7.38.1.3	cob_init_move	1072
7.38.1.4	cob_move	1072
7.38.1.5	cob_set_int	1075
7.39	libcob/move.h File Reference	1075
7.39.1	Function Documentation	1076
7.39.1.1	cob_get_int	1076
7.39.1.2	cob_move	1076
7.39.1.3	cob_set_int	1079
7.40	libcob/numeric.c File Reference	1079
7.40.1	Define Documentation	1081
7.40.1.1	COB_LIB_INCLUDE	1081
7.40.1.2	COB_MAX_BINARY	1081
7.40.1.3	DECIMAL_CHECK	1081
7.40.1.4	DECIMAL_NAN	1081
7.40.2	Function Documentation	1081
7.40.2.1	cob_add	1081
7.40.2.2	cob_add_int	1082
7.40.2.3	cob_cmp_int	1082
7.40.2.4	cob_cmp_long_numdisp	1082
7.40.2.5	cob_cmp_long_sign_numdisp	1083
7.40.2.6	cob_cmp_numdisp	1083
7.40.2.7	cob_cmp_packed	1084

7.40.2.8	<a href="#">cob_cmp_sign_numdisp</a>	1085
7.40.2.9	<a href="#">cob_cmp_uint</a>	1085
7.40.2.10	<a href="#">cob_decimal_add</a>	1085
7.40.2.11	<a href="#">cob_decimal_cmp</a>	1086
7.40.2.12	<a href="#">cob_decimal_div</a>	1086
7.40.2.13	<a href="#">cob_decimal_get_field</a>	1086
7.40.2.14	<a href="#">cob_decimal_init</a>	1087
7.40.2.15	<a href="#">cob_decimal_mul</a>	1088
7.40.2.16	<a href="#">cob_decimal_pow</a>	1088
7.40.2.17	<a href="#">cob_decimal_set_field</a>	1088
7.40.2.18	<a href="#">cob_decimal_sub</a>	1089
7.40.2.19	<a href="#">cob_div_quotient</a>	1089
7.40.2.20	<a href="#">cob_div_remainder</a>	1090
7.40.2.21	<a href="#">cob_init_numeric</a>	1090
7.40.2.22	<a href="#">cob_numeric_cmp</a>	1090
7.40.2.23	<a href="#">cob_set_packed_int</a>	1090
7.40.2.24	<a href="#">cob_set_packed_zero</a>	1091
7.40.2.25	<a href="#">cob_sub</a>	1091
7.40.2.26	<a href="#">cob_sub_int</a>	1091
7.41	<a href="#">libcob/numeric.h File Reference</a>	1092
7.41.1	<a href="#">Define Documentation</a>	1093
7.41.1.1	<a href="#">COB_STORE_KEEP_ON_OVERFLOW</a>	1093
7.41.1.2	<a href="#">COB_STORE_ROUND</a>	1094
7.41.1.3	<a href="#">COB_STORE_TRUNC_ON_OVERFLOW</a>	1094
7.41.2	<a href="#">Function Documentation</a>	1094
7.41.2.1	<a href="#">cob_add</a>	1094
7.41.2.2	<a href="#">cob_add_int</a>	1094
7.41.2.3	<a href="#">cob_cmp_int</a>	1095
7.41.2.4	<a href="#">cob_cmp_long_numdisp</a>	1095
7.41.2.5	<a href="#">cob_cmp_long_sign_numdisp</a>	1095
7.41.2.6	<a href="#">cob_cmp_numdisp</a>	1096
7.41.2.7	<a href="#">cob_cmp_packed</a>	1096
7.41.2.8	<a href="#">cob_cmp_sign_numdisp</a>	1097
7.41.2.9	<a href="#">cob_cmp_uint</a>	1098

---

7.41.2.10	<a href="#">cob_decimal_add</a>	1098
7.41.2.11	<a href="#">cob_decimal_cmp</a>	1098
7.41.2.12	<a href="#">cob_decimal_div</a>	1098
7.41.2.13	<a href="#">cob_decimal_get_field</a>	1099
7.41.2.14	<a href="#">cob_decimal_init</a>	1100
7.41.2.15	<a href="#">cob_decimal_mul</a>	1100
7.41.2.16	<a href="#">cob_decimal_pow</a>	1100
7.41.2.17	<a href="#">cob_decimal_set_field</a>	1101
7.41.2.18	<a href="#">cob_decimal_sub</a>	1101
7.41.2.19	<a href="#">cob_div_quotient</a>	1101
7.41.2.20	<a href="#">cob_div_remainder</a>	1102
7.41.2.21	<a href="#">cob_set_packed_int</a>	1102
7.41.2.22	<a href="#">cob_set_packed_zero</a>	1103
7.41.2.23	<a href="#">cob_sub</a>	1103
7.41.2.24	<a href="#">cob_sub_int</a>	1103
7.42	<a href="#">libcob/screenio.c File Reference</a>	1104
7.42.1	<a href="#">Define Documentation</a>	1105
7.42.1.1	<a href="#">_XOPEN_SOURCE_EXTENDED</a>	1105
7.42.1.2	<a href="#">COB_GEN_SCREENIO</a>	1105
7.42.1.3	<a href="#">COB_INP_SIZE</a>	1105
7.42.2	<a href="#">Function Documentation</a>	1105
7.42.2.1	<a href="#">cob_field_accept</a>	1105
7.42.2.2	<a href="#">cob_field_display</a>	1108
7.42.2.3	<a href="#">cob_screen_accept</a>	1109
7.42.2.4	<a href="#">cob_screen_display</a>	1110
7.42.2.5	<a href="#">cob_screen_line_col</a>	1111
7.42.2.6	<a href="#">cob_screen_set_mode</a>	1111
7.42.2.7	<a href="#">cob_screen_terminate</a>	1111
7.42.3	<a href="#">Variable Documentation</a>	1111
7.42.3.1	<a href="#">cob_screen_initialized</a>	1111
7.42.3.2	<a href="#">cob_screen_mode</a>	1112
7.43	<a href="#">libcob/screenio.h File Reference</a>	1112
7.43.1	<a href="#">Define Documentation</a>	1114
7.43.1.1	<a href="#">COB_SCREEN_AUTO</a>	1114

---

7.43.1.2	COB_SCREEN_BELL	1114
7.43.1.3	COB_SCREEN_BLACK	1114
7.43.1.4	COB_SCREEN_BLANK_LINE	1114
7.43.1.5	COB_SCREEN_BLANK_SCREEN	1114
7.43.1.6	COB_SCREEN_BLINK	1114
7.43.1.7	COB_SCREEN_BLUE	1114
7.43.1.8	COB_SCREEN_COLUMN_MINUS	1115
7.43.1.9	COB_SCREEN_COLUMN_PLUS	1115
7.43.1.10	COB_SCREEN_CYAN	1115
7.43.1.11	COB_SCREEN_ERASE_EOL	1115
7.43.1.12	COB_SCREEN_ERASE_EOS	1115
7.43.1.13	COB_SCREEN_FULL	1115
7.43.1.14	COB_SCREEN_GREEN	1115
7.43.1.15	COB_SCREEN_HIGHLIGHT	1115
7.43.1.16	COB_SCREEN_INPUT	1115
7.43.1.17	COB_SCREEN_LINE_MINUS	1115
7.43.1.18	COB_SCREEN_LINE_PLUS	1116
7.43.1.19	COB_SCREEN_LOWLIGHT	1116
7.43.1.20	COB_SCREEN_MAGENTA	1116
7.43.1.21	COB_SCREEN_OVERLINE	1116
7.43.1.22	COB_SCREEN_PROMPT	1116
7.43.1.23	COB_SCREEN_RED	1116
7.43.1.24	COB_SCREEN_REQUIRED	1116
7.43.1.25	COB_SCREEN_REVERSE	1116
7.43.1.26	COB_SCREEN_SCROLL_DOWN	1116
7.43.1.27	COB_SCREEN_SECURE	1116
7.43.1.28	COB_SCREEN_TYPE_ATTRIBUTE	1117
7.43.1.29	COB_SCREEN_TYPE_FIELD	1117
7.43.1.30	COB_SCREEN_TYPE_GROUP	1117
7.43.1.31	COB_SCREEN_TYPE_VALUE	1117
7.43.1.32	COB_SCREEN_UNDERLINE	1117
7.43.1.33	COB_SCREEN_UPDATE	1117
7.43.1.34	COB_SCREEN_WHITE	1117
7.43.1.35	COB_SCREEN_YELLOW	1117

---

7.43.2	Typedef Documentation	1117
7.43.2.1	cob_screen	1117
7.43.3	Function Documentation	1117
7.43.3.1	cob_field_accept	1118
7.43.3.2	cob_field_display	1121
7.43.3.3	cob_screen_accept	1121
7.43.3.4	cob_screen_display	1122
7.43.3.5	cob_screen_line_col	1123
7.43.4	Variable Documentation	1123
7.43.4.1	cob_screen_mode	1123
7.44	libcob/strings.c File Reference	1123
7.44.1	Define Documentation	1125
7.44.1.1	DLM_DEFAULT_NUM	1125
7.44.1.2	INSPECT_ALL	1125
7.44.1.3	INSPECT_FIRST	1125
7.44.1.4	INSPECT_LEADING	1125
7.44.1.5	INSPECT_TRAILING	1125
7.44.2	Function Documentation	1126
7.44.2.1	cob_init_strings	1126
7.44.2.2	cob_inspect_after	1126
7.44.2.3	cob_inspect_all	1126
7.44.2.4	cob_inspect_before	1126
7.44.2.5	cob_inspect_characters	1127
7.44.2.6	cob_inspect_converting	1127
7.44.2.7	cob_inspect_finish	1128
7.44.2.8	cob_inspect_first	1128
7.44.2.9	cob_inspect_init	1128
7.44.2.10	cob_inspect_leading	1129
7.44.2.11	cob_inspect_start	1129
7.44.2.12	cob_inspect_trailing	1129
7.44.2.13	cob_string_append	1129
7.44.2.14	cob_string_delimited	1130
7.44.2.15	cob_string_finish	1130
7.44.2.16	cob_string_init	1130

---

7.44.2.17	cob_unstring_delimited	1131
7.44.2.18	cob_unstring_finish	1131
7.44.2.19	cob_unstring_init	1131
7.44.2.20	cob_unstring_into	1132
7.44.2.21	cob_unstring_tallying	1134
7.45	libcob/strings.h File Reference	1134
7.45.1	Function Documentation	1135
7.45.1.1	cob_inspect_after	1135
7.45.1.2	cob_inspect_all	1136
7.45.1.3	cob_inspect_before	1136
7.45.1.4	cob_inspect_characters	1136
7.45.1.5	cob_inspect_converting	1137
7.45.1.6	cob_inspect_finish	1137
7.45.1.7	cob_inspect_first	1137
7.45.1.8	cob_inspect_init	1138
7.45.1.9	cob_inspect_leading	1138
7.45.1.10	cob_inspect_start	1138
7.45.1.11	cob_inspect_trailing	1139
7.45.1.12	cob_string_append	1139
7.45.1.13	cob_string_delimited	1139
7.45.1.14	cob_string_finish	1140
7.45.1.15	cob_string_init	1140
7.45.1.16	cob_unstring_delimited	1140
7.45.1.17	cob_unstring_finish	1140
7.45.1.18	cob_unstring_init	1141
7.45.1.19	cob_unstring_into	1142
7.45.1.20	cob_unstring_tallying	1143
7.46	libcob/termio.c File Reference	1143
7.46.1	Function Documentation	1144
7.46.1.1	cob_accept	1144
7.46.1.2	cob_display	1145
7.46.1.3	cob_init_termio	1145
7.47	libcob/termio.h File Reference	1146
7.47.1	Function Documentation	1147



## CONTENTS

cxix

---

7.47.1.1	<a href="#">cob_accept</a>	1147
7.47.1.2	<a href="#">cob_display</a>	1147
7.48	<a href="#">mainpage.h File Reference</a>	1148
7.49	<a href="#">tarstamp.h File Reference</a>	1148



# Chapter 1

## OpenCOBOL 1.1pre-rel source tree

### Authors

Keisuke Nishida, Roger While

### 1.1 Introduction

This is the OpenCOBOL compiler suite source tree, beautified by Doxygen. All rights remain the authors, Keisuke Nishida and Roger While.

See the Classes (try [cb\\_field](#) for it's dependency graph) and Files, where:

1. src/cobc Contains the compiler sources
2. src/libcob Contains the runtime support sources
3. src/lib and
4. src/bin contain build support files

As you prepare to develop code with OpenCOBOL, please be sure to visit [OpenCOBOL.org](http://OpenCOBOL.org)

Many thanks to Dimitri for Doxygen. This is sweet.

Many thanks to Roger and Keisuke. Copyrights for OpenCOBOL are Roger While and Keisuke Nishida.



## Chapter 2

# Directory Hierarchy

### 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

bin . . . . .	9
cobc . . . . .	10
lib . . . . .	11
libcob . . . . .	12



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">__cob_screen</a>	13
<a href="#">call_hash</a>	15
<a href="#">cb_alphabet_name</a>	16
<a href="#">cb_alt_key</a>	18
<a href="#">cb_assign</a>	19
<a href="#">cb_binary_op</a>	21
<a href="#">cb_call</a>	22
<a href="#">cb_cast</a>	24
<a href="#">cb_class_name</a>	25
<a href="#">cb_const</a>	26
<a href="#">cb_continue</a>	27
<a href="#">cb_decimal</a>	28
<a href="#">cb_exception</a>	29
<a href="#">cb_field</a>	30
<a href="#">cb_file</a>	39
<a href="#">cb_funcall</a>	44
<a href="#">cb_goto</a>	46
<a href="#">cb_if</a>	47
<a href="#">cb_initialize</a>	48
<a href="#">cb_integer</a>	50
<a href="#">cb_intrinsic</a>	51
<a href="#">cb_intrinsic_table</a>	53
<a href="#">cb_field::cb_key</a>	54
<a href="#">cb_label</a>	55
<a href="#">cb_level_78</a>	59
<a href="#">cb_list</a>	60
<a href="#">cb_literal</a>	61
<a href="#">cb_locale_name</a>	63
<a href="#">cb_perform</a>	64

<a href="#">cb_perform_varying</a>	66
<a href="#">cb_picture</a>	68
<a href="#">cb_program</a>	70
<a href="#">cb_reference</a>	79
<a href="#">cb_replace_list</a>	81
<a href="#">cb_search</a>	82
<a href="#">cb_statement</a>	84
<a href="#">cb_string</a>	87
<a href="#">cb_system_name</a>	88
<a href="#">cb_text_list</a>	89
<a href="#">cb_tree_common</a>	90
<a href="#">cb_word</a>	90
<a href="#">cob_alloc_cache</a>	92
<a href="#">cob_decimal</a>	93
<a href="#">cob_exception</a>	93
<a href="#">cob_external</a>	94
<a href="#">cob_field</a>	95
<a href="#">cob_field_attr</a>	97
<a href="#">cob_file</a>	98
<a href="#">cob_file_key</a>	102
<a href="#">cob_fileio_funcs</a>	104
<a href="#">cob_inp_struct</a>	106
<a href="#">cob_module</a>	107
<a href="#">cobitem</a>	110
<a href="#">cobjmp_buf</a>	111
<a href="#">cobsort</a>	113
<a href="#">dlim_struct</a>	116
<a href="#">expr_node</a>	117
<a href="#">file_struct</a>	118
<a href="#">filename</a>	118
<a href="#">handler_struct</a>	120
<a href="#">indexed_file</a>	122
<a href="#">linage_struct</a>	124
<a href="#">local_filename</a>	127
<a href="#">memory_struct</a>	128
<a href="#">noreserve</a>	129
<a href="#">option</a>	130
<a href="#">reserved</a>	130
<a href="#">sort_list</a>	131
<a href="#">struct_handle</a>	132
<a href="#">system_table</a>	133
<a href="#">yy_buffer_state</a>	133
<a href="#">yyalloc</a>	135
<a href="#">YYSTYPE</a>	136



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">config.h</a>	797
<a href="#">cpucheck.c</a>	805
<a href="#">defaults.h</a>	807
<a href="#">libcob.h</a>	832
<a href="#">mainpage.h</a>	1148
<a href="#">tarstamp.h</a>	1148
<a href="#">bin/cobcrun.c</a>	139
<a href="#">cobc/cobc.c</a>	140
<a href="#">cobc/cobc.h</a>	154
<a href="#">cobc/codegen.c</a>	171
<a href="#">cobc/config.c</a>	184
<a href="#">cobc/error.c</a>	191
<a href="#">cobc/field.c</a>	197
<a href="#">cobc/parser.c</a>	204
<a href="#">cobc/parser.h</a>	309
<a href="#">cobc/pplex.c</a>	401
<a href="#">cobc/ppparse.c</a>	414
<a href="#">cobc/ppparse.h</a>	458
<a href="#">cobc/reserved.c</a>	492
<a href="#">cobc/scanner.c</a>	496
<a href="#">cobc/tree.c</a>	508
<a href="#">cobc/tree.h</a>	551
<a href="#">cobc/typeck.c</a>	722
<a href="#">lib/dummysmac.c</a>	809
<a href="#">lib/getopt.c</a>	809
<a href="#">lib/getopt.h</a>	824
<a href="#">lib/getopt1.c</a>	826
<a href="#">lib/gettext.h</a>	828
<a href="#">libcob/byteswap.h</a>	833

---

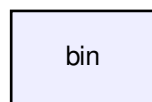
libcob/call.c	834
libcob/call.h	845
libcob/coblocal.h	854
libcob/codegen.h	862
libcob/common.c	871
libcob/common.h	875
libcob/fileio.c	892
libcob/fileio.h	923
libcob/intrinsic.c	960
libcob/intrinsic.h	1016
libcob/move.c	1070
libcob/move.h	1075
libcob/numeric.c	1079
libcob/numeric.h	1092
libcob/screenio.c	1104
libcob/screenio.h	1112
libcob/strings.c	1123
libcob/strings.h	1134
libcob/termio.c	1143
libcob/termio.h	1146

## Chapter 5

# Directory Documentation

### 5.1 bin/ Directory Reference

Directory dependency graph for bin/:

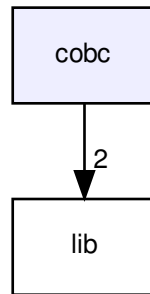


#### Files

- file [cobcrun.c](#)

## 5.2 cobc/ Directory Reference

Directory dependency graph for cobc/:

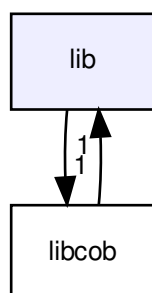


### Files

- file [cobc.c](#)
- file [cobc.h](#)
- file [codegen.c](#)
- file [config.c](#)
- file [error.c](#)
- file [field.c](#)
- file [parser.c](#)
- file [parser.h](#)
- file [plex.c](#)
- file [ppparse.c](#)
- file [ppparse.h](#)
- file [reserved.c](#)
- file [scanner.c](#)
- file [tree.c](#)
- file [tree.h](#)
- file [typeck.c](#)

### 5.3 lib/ Directory Reference

Directory dependency graph for lib/:

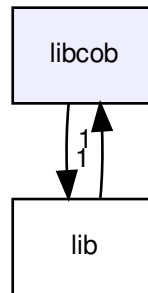


#### Files

- file [dummymac.c](#)
- file [getopt.c](#)
- file [getopt.h](#)
- file [getopt1.c](#)
- file [gettext.h](#)

## 5.4 libcob/ Directory Reference

Directory dependency graph for libcob/:



### Files

- file [byteswap.h](#)
- file [call.c](#)
- file [call.h](#)
- file [coblocal.h](#)
- file [codegen.h](#)
- file [common.c](#)
- file [common.h](#)
- file [fileio.c](#)
- file [fileio.h](#)
- file [intrinsic.c](#)
- file [intrinsic.h](#)
- file [move.c](#)
- file [move.h](#)
- file [numeric.c](#)
- file [numeric.h](#)
- file [screenio.c](#)
- file [screenio.h](#)
- file [strings.c](#)
- file [strings.h](#)
- file [termio.c](#)
- file [termio.h](#)

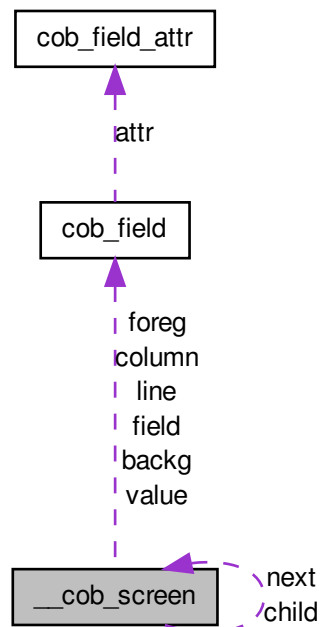
## Chapter 6

# Class Documentation

### 6.1 \_\_cob\_screen Struct Reference

```
#include <screenio.h>
```

Collaboration diagram for \_\_cob\_screen:



## Public Attributes

- [cob\\_screen](#) \* next
- [cob\\_screen](#) \* child
- [cob\\_field](#) \* field
- [cob\\_field](#) \* value
- [cob\\_field](#) \* line
- [cob\\_field](#) \* column
- [cob\\_field](#) \* foreg
- [cob\\_field](#) \* backg
- int type
- int occurs
- int attr

### 6.1.1 Detailed Description

Definition at line 67 of file screenio.h.

### 6.1.2 Member Data Documentation

#### 6.1.2.1 int \_\_cob\_screen::attr

Definition at line 78 of file screenio.h.

#### 6.1.2.2 cob\_field\* \_\_cob\_screen::backg

Definition at line 75 of file screenio.h.

#### 6.1.2.3 cob\_screen\* \_\_cob\_screen::child

Definition at line 69 of file screenio.h.

#### 6.1.2.4 cob\_field\* \_\_cob\_screen::column

Definition at line 73 of file screenio.h.

#### 6.1.2.5 cob\_field\* \_\_cob\_screen::field

Definition at line 70 of file screenio.h.

#### 6.1.2.6 cob\_field\* \_\_cob\_screen::foreg

Definition at line 74 of file screenio.h.



### 6.1.2.7 cob\_field\* \_\_cob\_screen::line

Definition at line 72 of file screenio.h.

### 6.1.2.8 cob\_screen\* \_\_cob\_screen::next

Definition at line 68 of file screenio.h.

### 6.1.2.9 int \_\_cob\_screen::occurs

Definition at line 77 of file screenio.h.

### 6.1.2.10 int \_\_cob\_screen::type

Definition at line 76 of file screenio.h.

### 6.1.2.11 cob\_field\* \_\_cob\_screen::value

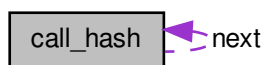
Definition at line 71 of file screenio.h.

The documentation for this struct was generated from the following file:

- [libcob/screenio.h](#)

## 6.2 call\_hash Struct Reference

Collaboration diagram for call\_hash:



### Public Attributes

- struct [call\\_hash](#) \* [next](#)
- const char \* [name](#)
- void \* [func](#)
- void \* [cancel](#)

- [size\\_t flag\\_is\\_active](#)

### 6.2.1 Detailed Description

Definition at line 132 of file call.c.

### 6.2.2 Member Data Documentation

#### 6.2.2.1 void\* call\_hash::cancel

Definition at line 136 of file call.c.

#### 6.2.2.2 size\_t call\_hash::flag\_is\_active

Definition at line 137 of file call.c.

#### 6.2.2.3 void\* call\_hash::func

Definition at line 135 of file call.c.

#### 6.2.2.4 const char\* call\_hash::name

Definition at line 134 of file call.c.

#### 6.2.2.5 struct call\_hash\* call\_hash::next

Definition at line 133 of file call.c.

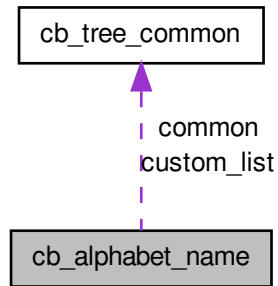
The documentation for this struct was generated from the following file:

- [libcob/call.c](#)

## 6.3 cb\_alphabet\_name Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_alphabet_name`:



### Public Attributes

- struct `cb_tree_common` `common`
- const char \* `name`
- char \* `cname`
- `cb_tree` `custom_list`
- enum `cb_alphabet_name_type` `type`
- int `low_val_char`
- int `high_val_char`

### 6.3.1 Detailed Description

Definition at line 328 of file `tree.h`.

### 6.3.2 Member Data Documentation

#### 6.3.2.1 char\* `cb_alphabet_name::cname`

Definition at line 331 of file `tree.h`.

#### 6.3.2.2 struct `cb_tree_common` `cb_alphabet_name::common`

Definition at line 329 of file `tree.h`.

### 6.3.2.3 `cb_tree` `cb_alphabet_name::custom_list`

Definition at line 332 of file tree.h.

### 6.3.2.4 `int` `cb_alphabet_name::high_val_char`

Definition at line 335 of file tree.h.

### 6.3.2.5 `int` `cb_alphabet_name::low_val_char`

Definition at line 334 of file tree.h.

### 6.3.2.6 `const char*` `cb_alphabet_name::name`

Definition at line 330 of file tree.h.

### 6.3.2.7 `enum` `cb_alphabet_name_type` `cb_alphabet_name::type`

Definition at line 333 of file tree.h.

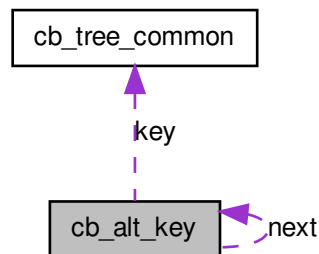
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.4 `cb_alt_key` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_alt_key`:



## Public Attributes

- struct `cb_alt_key` \* `next`
- `cb_tree` `key`
- int `duplicates`
- int `offset`

### 6.4.1 Detailed Description

Definition at line 592 of file `tree.h`.

### 6.4.2 Member Data Documentation

#### 6.4.2.1 int `cb_alt_key::duplicates`

Definition at line 595 of file `tree.h`.

#### 6.4.2.2 `cb_tree` `cb_alt_key::key`

Definition at line 594 of file `tree.h`.

#### 6.4.2.3 struct `cb_alt_key`\* `cb_alt_key::next`

Definition at line 593 of file `tree.h`.

#### 6.4.2.4 int `cb_alt_key::offset`

Definition at line 596 of file `tree.h`.

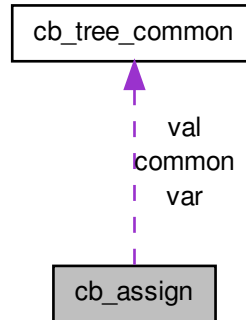
The documentation for this struct was generated from the following file:

- `cobc/tree.h`

## 6.5 `cb_assign` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_assign`:



### Public Attributes

- struct `cb_tree_common` `common`
- `cb_tree` `var`
- `cb_tree` `val`

#### 6.5.1 Detailed Description

Definition at line 793 of file `tree.h`.

#### 6.5.2 Member Data Documentation

##### 6.5.2.1 struct `cb_tree_common` `cb_assign::common`

Definition at line 794 of file `tree.h`.

##### 6.5.2.2 `cb_tree` `cb_assign::val`

Definition at line 796 of file `tree.h`.

##### 6.5.2.3 `cb_tree` `cb_assign::var`

Definition at line 795 of file `tree.h`.

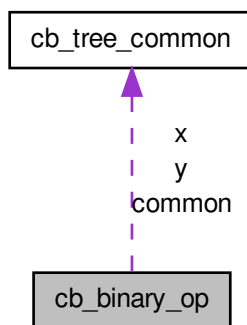
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.6 `cb_binary_op` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_binary_op`:



### Public Attributes

- struct [cb\\_tree\\_common](#) `common`
- int `op`
- [cb\\_tree](#) `x`
- [cb\\_tree](#) `y`

### 6.6.1 Detailed Description

Definition at line 704 of file `tree.h`.

### 6.6.2 Member Data Documentation

#### 6.6.2.1 struct `cb_tree_common` `cb_binary_op::common`

Definition at line 705 of file `tree.h`.

### 6.6.2.2 int `cb_binary_op::op`

Definition at line 706 of file `tree.h`.

### 6.6.2.3 `cb_tree` `cb_binary_op::x`

Definition at line 707 of file `tree.h`.

### 6.6.2.4 `cb_tree` `cb_binary_op::y`

Definition at line 708 of file `tree.h`.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.7 `cb_call` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_call`:



### Public Attributes

- struct `cb_tree_common` `common`



- [cb\\_tree name](#)
- [cb\\_tree args](#)
- [cb\\_tree stmt1](#)
- [cb\\_tree stmt2](#)
- [cb\\_tree returning](#)
- [int is\\_system](#)

### 6.7.1 Detailed Description

Definition at line 974 of file tree.h.

### 6.7.2 Member Data Documentation

#### 6.7.2.1 `cb_tree cb_call::args`

Definition at line 977 of file tree.h.

#### 6.7.2.2 `struct cb_tree_common cb_call::common`

Definition at line 975 of file tree.h.

#### 6.7.2.3 `int cb_call::is_system`

Definition at line 981 of file tree.h.

#### 6.7.2.4 `cb_tree cb_call::name`

Definition at line 976 of file tree.h.

#### 6.7.2.5 `cb_tree cb_call::returning`

Definition at line 980 of file tree.h.

#### 6.7.2.6 `cb_tree cb_call::stmt1`

Definition at line 978 of file tree.h.

#### 6.7.2.7 `cb_tree cb_call::stmt2`

Definition at line 979 of file tree.h.

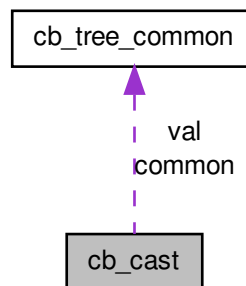
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.8 `cb_cast` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_cast`:



### Public Attributes

- struct `cb_tree_common` `common`
- enum `cb_cast_type` `type`
- `cb_tree` `val`

### 6.8.1 Detailed Description

Definition at line 771 of file `tree.h`.

### 6.8.2 Member Data Documentation

#### 6.8.2.1 struct `cb_tree_common` `cb_cast::common`

Definition at line 772 of file `tree.h`.

#### 6.8.2.2 enum `cb_cast_type` `cb_cast::type`

Definition at line 773 of file `tree.h`.

### 6.8.2.3 `cb_tree` `cb_cast::val`

Definition at line 774 of file `tree.h`.

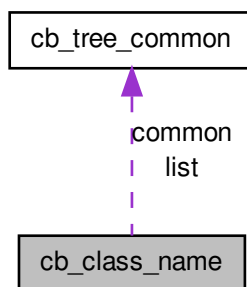
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.9 `cb_class_name` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_class_name`:



### Public Attributes

- struct `cb_tree_common` `common`
- `const char *` `name`
- `char *` `cname`
- `cb_tree` `list`

### 6.9.1 Detailed Description

Definition at line 348 of file `tree.h`.

### 6.9.2 Member Data Documentation

### 6.9.2.1 `char* cb_class_name::cname`

Definition at line 351 of file tree.h.

### 6.9.2.2 `struct cb_tree_common cb_class_name::common`

Definition at line 349 of file tree.h.

### 6.9.2.3 `cb_tree cb_class_name::list`

Definition at line 352 of file tree.h.

### 6.9.2.4 `const char* cb_class_name::name`

Definition at line 350 of file tree.h.

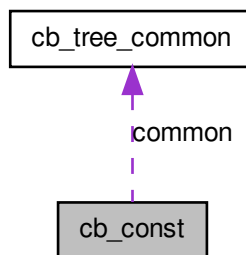
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.10 `cb_const` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_const`:



### Public Attributes

- struct [cb\\_tree\\_common](#) `common`
- `const char * val`

### 6.10.1 Detailed Description

Definition at line 280 of file `tree.h`.

### 6.10.2 Member Data Documentation

#### 6.10.2.1 `struct cb_tree_common cb_const::common`

Definition at line 281 of file `tree.h`.

#### 6.10.2.2 `const char* cb_const::val`

Definition at line 282 of file `tree.h`.

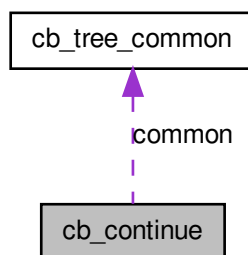
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.11 `cb_continue` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_continue`:



### Public Attributes

- struct [cb\\_tree\\_common common](#)

### 6.11.1 Detailed Description

Definition at line 1090 of file tree.h.

### 6.11.2 Member Data Documentation

#### 6.11.2.1 struct `cb_tree_common` `cb_continue::common`

Definition at line 1091 of file tree.h.

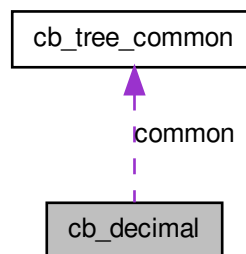
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.12 `cb_decimal` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_decimal`:



### Public Attributes

- struct `cb_tree_common` `common`
- int `id`

### 6.12.1 Detailed Description

Definition at line 421 of file tree.h.

### 6.12.2 Member Data Documentation

#### 6.12.2.1 `struct cb_tree_common cb_decimal::common`

Definition at line 422 of file `tree.h`.

#### 6.12.2.2 `int cb_decimal::id`

Definition at line 423 of file `tree.h`.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.13 `cb_exception` Struct Reference

```
#include <cobc.h>
```

### Public Attributes

- `const char * name`
- `const int code`
- `int enable`

### 6.13.1 Detailed Description

Definition at line 46 of file `cobc.h`.

### 6.13.2 Member Data Documentation

#### 6.13.2.1 `const int cb_exception::code`

Definition at line 48 of file `cobc.h`.

#### 6.13.2.2 `int cb_exception::enable`

Definition at line 49 of file `cobc.h`.

#### 6.13.2.3 `const char* cb_exception::name`

Definition at line 47 of file `cobc.h`.

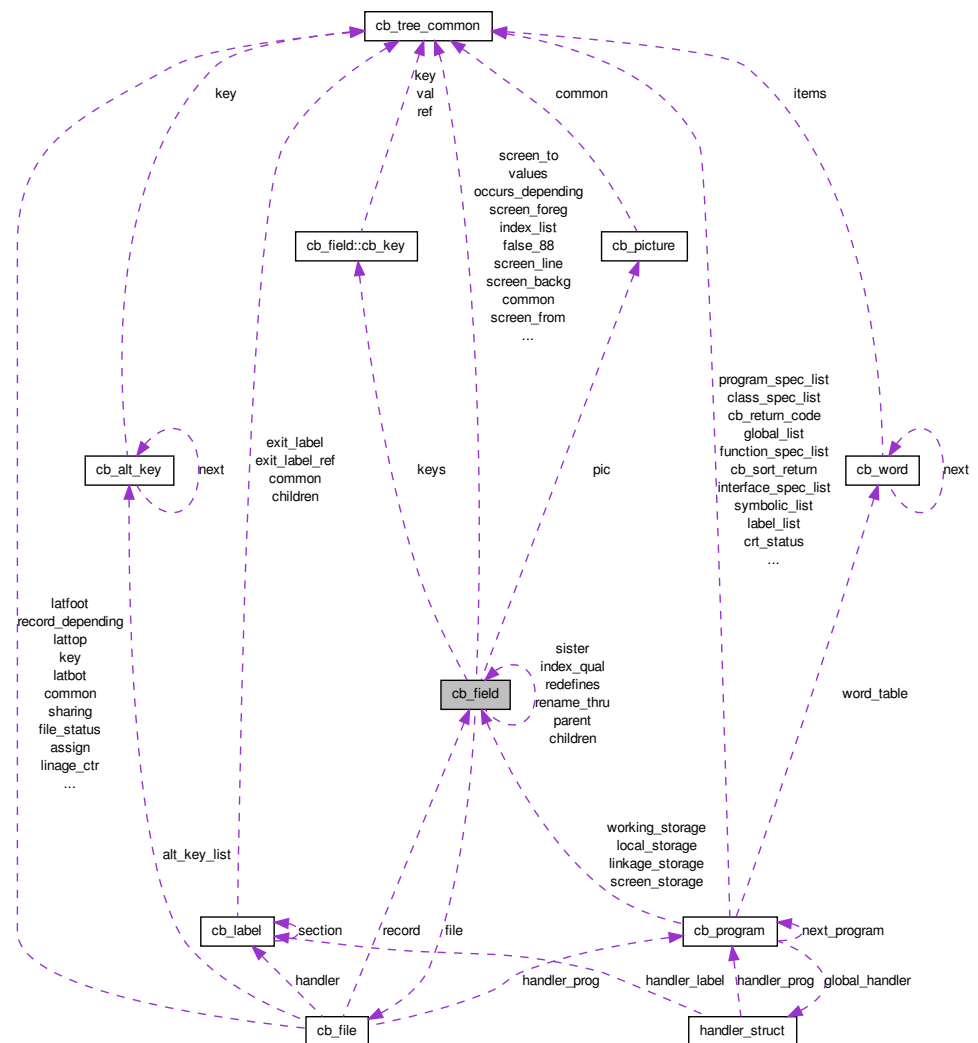
The documentation for this struct was generated from the following file:

- [cobc/cobc.h](#)

## 6.14 cb\_field Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_field`:





## Classes

- struct `cb_key`

## Public Attributes

- struct `cb_tree_common` `common`
- int `id`
- int `storage_id`
- const char \* `name`
- const char \* `ename`
- int `size`
- int `memory_size`
- int `offset`
- int `level`
- int `occurs_min`
- int `occurs_max`
- int `indexes`
- int `count`
- `cb_tree` `occurs Depending`
- enum `cb_storage` `storage`
- enum `cb_usage` `usage`
- `cb_tree` `values`
- `cb_tree` `false_88`
- `cb_tree` `index_list`
- struct `cb_field` \* `parent`
- struct `cb_field` \* `children`
- struct `cb_field` \* `sister`
- struct `cb_field` \* `redefines`
- struct `cb_field` \* `rename_thru`
- struct `cb_field` \* `index_qual`
- struct `cb_file` \* `file`
- struct `cb_field::cb_key` \* `keys`
- int `nkeys`
- int `param_num`
- struct `cb_picture` \* `pic`
- `cb_tree` `screen_line`
- `cb_tree` `screen_column`
- `cb_tree` `screen_from`
- `cb_tree` `screen_to`
- `cb_tree` `screen_foreg`
- `cb_tree` `screen_backg`
- int `screen_flag`
- unsigned int `flag_external`: 1
- unsigned int `flag_blank_zero`: 1
- unsigned int `flag_justified`: 1

- unsigned int `flag_sign_leading`: 1
- unsigned int `flag_sign_separate`: 1
- unsigned int `flag_synchronized`: 1
- unsigned int `flag_occurs`: 1
- unsigned int `flag_invalid`: 1
- unsigned int `flag_binary_swap`: 1
- unsigned int `flag_local`: 1
- unsigned int `flag_base`: 1
- unsigned int `flag_field`: 1
- unsigned int `flag_item_external`: 1
- unsigned int `flag_chained`: 1
- unsigned int `flag_real_binary`: 1
- unsigned int `flag_item_based`: 1
- unsigned int `flag_item_78`: 1
- unsigned int `flag_any_length`: 1
- unsigned int `flag_anylen_done`: 1
- unsigned int `flag_indexed_by`: 1
- unsigned int `flag_is_pointer`: 1
- unsigned int `flag_is_verified`: 1
- unsigned int `flag_is_global`: 1
- unsigned int `flag_is_c_long`: 1
- unsigned int `flag_is_pdiv_parm`: 1
- unsigned int `flag_local_allocated`: 1
- unsigned int `flag_no_init`: 1
- unsigned int `flag_spare`: 5

### 6.14.1 Detailed Description

Definition at line 459 of file tree.h.

### 6.14.2 Member Data Documentation

#### 6.14.2.1 struct `cb_field*` `cb_field::children`

Definition at line 480 of file tree.h.

#### 6.14.2.2 struct `cb_tree_common` `cb_field::common`

Definition at line 460 of file tree.h.

#### 6.14.2.3 int `cb_field::count`

Definition at line 472 of file tree.h.

**6.14.2.4 const char\* cb\_field::ename**

Definition at line 464 of file tree.h.

**6.14.2.5 cb\_tree cb\_field::false\_88**

Definition at line 477 of file tree.h.

**6.14.2.6 struct cb\_file\* cb\_field::file**

Definition at line 485 of file tree.h.

**6.14.2.7 unsigned int cb\_field::flag\_any\_length**

Definition at line 521 of file tree.h.

**6.14.2.8 unsigned int cb\_field::flag\_anylen\_done**

Definition at line 522 of file tree.h.

**6.14.2.9 unsigned int cb\_field::flag\_base**

Definition at line 514 of file tree.h.

**6.14.2.10 unsigned int cb\_field::flag\_binary\_swap**

Definition at line 512 of file tree.h.

**6.14.2.11 unsigned int cb\_field::flag\_blank\_zero**

Definition at line 505 of file tree.h.

**6.14.2.12 unsigned int cb\_field::flag\_chained**

Definition at line 517 of file tree.h.

**6.14.2.13 unsigned int cb\_field::flag\_external**

Definition at line 504 of file tree.h.

**6.14.2.14 unsigned int cb\_field::flag\_field**

Definition at line 515 of file tree.h.

**6.14.2.15 unsigned int cb\_field::flag\_indexed\_by**

Definition at line 523 of file tree.h.

**6.14.2.16 unsigned int cb\_field::flag\_invalid**

Definition at line 511 of file tree.h.

**6.14.2.17 unsigned int cb\_field::flag\_is\_c\_long**

Definition at line 527 of file tree.h.

**6.14.2.18 unsigned int cb\_field::flag\_is\_global**

Definition at line 526 of file tree.h.

**6.14.2.19 unsigned int cb\_field::flag\_is\_pdiv\_parm**

Definition at line 528 of file tree.h.

**6.14.2.20 unsigned int cb\_field::flag\_is\_pointer**

Definition at line 524 of file tree.h.

**6.14.2.21 unsigned int cb\_field::flag\_is\_verified**

Definition at line 525 of file tree.h.

**6.14.2.22 unsigned int cb\_field::flag\_item\_78**

Definition at line 520 of file tree.h.

**6.14.2.23 unsigned int cb\_field::flag\_item\_based**

Definition at line 519 of file tree.h.

**6.14.2.24 unsigned int cb\_field::flag\_item\_external**

Definition at line 516 of file tree.h.

**6.14.2.25 unsigned int cb\_field::flag\_justified**

Definition at line 506 of file tree.h.

**6.14.2.26 unsigned int cb\_field::flag\_local**

Definition at line 513 of file tree.h.

**6.14.2.27 unsigned int cb\_field::flag\_local\_allocated**

Definition at line 529 of file tree.h.

**6.14.2.28 unsigned int cb\_field::flag\_no\_init**

Definition at line 530 of file tree.h.

**6.14.2.29 unsigned int cb\_field::flag\_occurs**

Definition at line 510 of file tree.h.

**6.14.2.30 unsigned int cb\_field::flag\_real\_binary**

Definition at line 518 of file tree.h.

**6.14.2.31 unsigned int cb\_field::flag\_sign\_leading**

Definition at line 507 of file tree.h.

**6.14.2.32 unsigned int cb\_field::flag\_sign\_separate**

Definition at line 508 of file tree.h.

**6.14.2.33 unsigned int cb\_field::flag\_spare**

Definition at line 531 of file tree.h.

**6.14.2.34 unsigned int cb\_field::flag\_synchronized**

Definition at line 509 of file tree.h.

**6.14.2.35 int cb\_field::id**

Definition at line 461 of file tree.h.

**6.14.2.36 cb\_tree cb\_field::index\_list**

Definition at line 478 of file tree.h.

**6.14.2.37 struct cb\_field\* cb\_field::index\_qual**

Definition at line 484 of file tree.h.

**6.14.2.38 int cb\_field::indexes**

Definition at line 471 of file tree.h.

**6.14.2.39 struct cb\_field::cb\_key \* cb\_field::keys****6.14.2.40 int cb\_field::level**

Definition at line 468 of file tree.h.

**6.14.2.41 int cb\_field::memory\_size**

Definition at line 466 of file tree.h.

**6.14.2.42 const char\* cb\_field::name**

Definition at line 463 of file tree.h.

**6.14.2.43 int cb\_field::nkeys**

Definition at line 492 of file tree.h.

**6.14.2.44 cb\_tree cb\_field::occursDepending**

Definition at line 473 of file tree.h.

**6.14.2.45 int cb\_field::occurs\_max**

Definition at line 470 of file tree.h.

**6.14.2.46 int cb\_field::occurs\_min**

Definition at line 469 of file tree.h.

**6.14.2.47 int cb\_field::offset**

Definition at line 467 of file tree.h.

**6.14.2.48 int cb\_field::param\_num**

Definition at line 493 of file tree.h.

**6.14.2.49 struct cb\_field\* cb\_field::parent**

Definition at line 479 of file tree.h.

**6.14.2.50 struct cb\_picture\* cb\_field::pic**

Definition at line 494 of file tree.h.

**6.14.2.51 struct cb\_field\* cb\_field::redefines**

Definition at line 482 of file tree.h.

**6.14.2.52 struct cb\_field\* cb\_field::rename\_thru**

Definition at line 483 of file tree.h.

**6.14.2.53 cb\_tree cb\_field::screen\_backg**

Definition at line 501 of file tree.h.

**6.14.2.54 cb\_tree cb\_field::screen\_column**

Definition at line 497 of file tree.h.

**6.14.2.55 int cb\_field::screen\_flag**

Definition at line 502 of file tree.h.

**6.14.2.56 cb\_tree cb\_field::screen\_foreg**

Definition at line 500 of file tree.h.

**6.14.2.57 cb\_tree cb\_field::screen\_from**

Definition at line 498 of file tree.h.

**6.14.2.58 cb\_tree cb\_field::screen\_line**

Definition at line 496 of file tree.h.

**6.14.2.59 cb\_tree cb\_field::screen\_to**

Definition at line 499 of file tree.h.

**6.14.2.60 struct cb\_field\* cb\_field::sister**

Definition at line 481 of file tree.h.

**6.14.2.61 int cb\_field::size**

Definition at line 465 of file tree.h.

**6.14.2.62 enum cb\_storage cb\_field::storage**

Definition at line 474 of file tree.h.

**6.14.2.63 int cb\_field::storage\_id**

Definition at line 462 of file tree.h.

**6.14.2.64 enum cb\_usage cb\_field::usage**

Definition at line 475 of file tree.h.



**6.14.2.65 cb\_tree cb\_field::values**

Definition at line 476 of file tree.h.

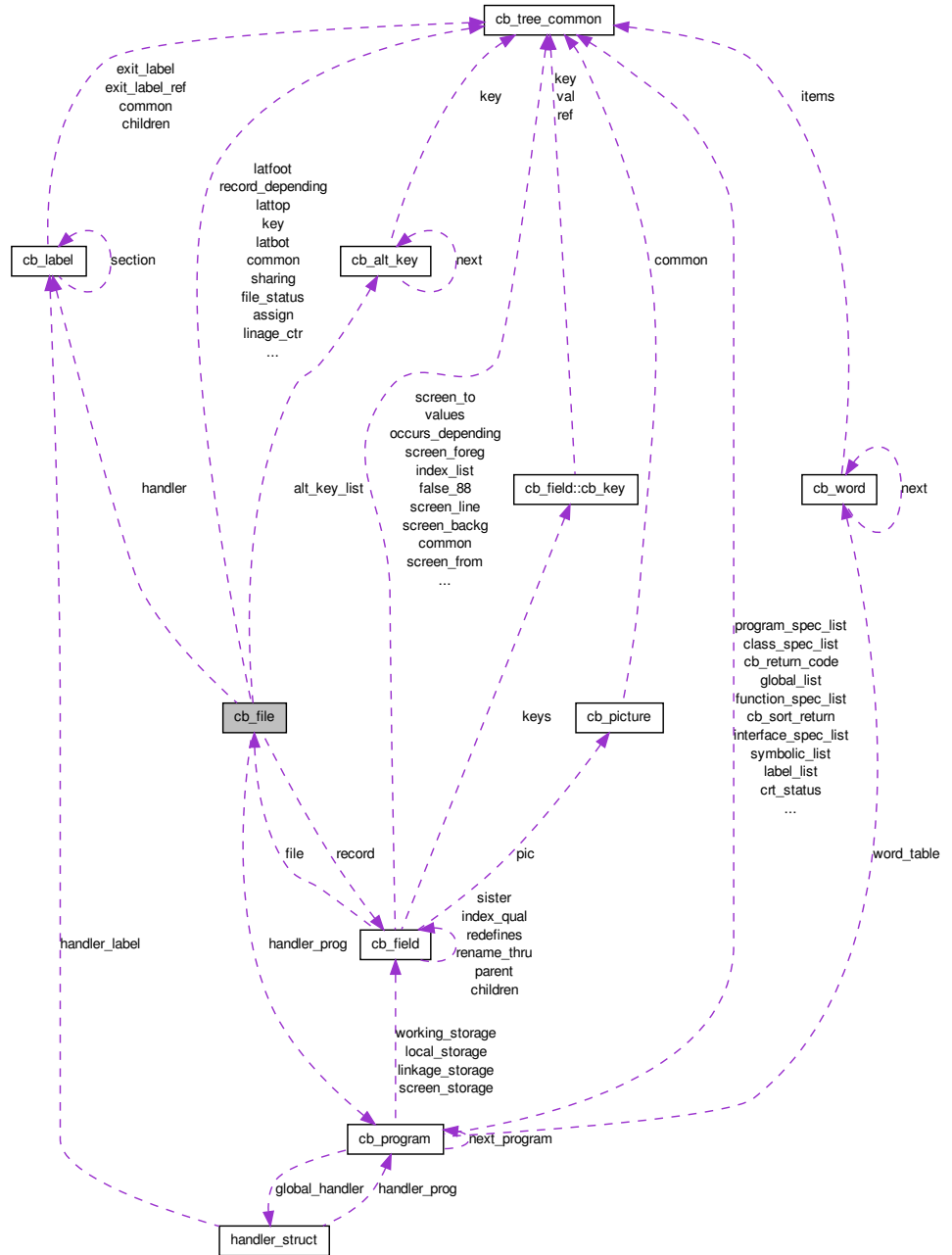
The documentation for this struct was generated from the following file:

- [cobic/tree.h](#)

**6.15 cb\_file Struct Reference**

```
#include <tree.h>
```

Collaboration diagram for `cb_file`:



## Public Attributes

- struct `cb_tree_common` `common`
- const char \* `name`
- char \* `cname`
- `cb_tree` `assign`
- `cb_tree` `file_status`
- `cb_tree` `sharing`
- `cb_tree` `key`
- struct `cb_alt_key` \* `alt_key_list`
- struct `cb_field` \* `record`
- `cb_tree` `recordDepending`
- `cb_tree` `linage`
- `cb_tree` `linage_ctr`
- `cb_tree` `latfoot`
- `cb_tree` `lattop`
- `cb_tree` `latbot`
- struct `cb_label` \* `handler`
- struct `cb_program` \* `handler_prog`
- int `record_min`
- int `record_max`
- int `optional`
- int `organization`
- int `access_mode`
- int `lock_mode`
- int `same_clause`
- int `finalized`
- int `external`
- int `special`
- int `external_assign`
- int `fileid_assign`
- int `global`

### 6.15.1 Detailed Description

Definition at line 599 of file `tree.h`.

### 6.15.2 Member Data Documentation

#### 6.15.2.1 int `cb_file::access_mode`

Definition at line 623 of file `tree.h`.

#### 6.15.2.2 struct `cb_alt_key`\* `cb_file::alt_key_list`

Definition at line 608 of file `tree.h`.

**6.15.2.3   cb\_tree cb\_file::assign**

Definition at line 604 of file tree.h.

**6.15.2.4   char\* cb\_file::cname**

Definition at line 602 of file tree.h.

**6.15.2.5   struct cb\_tree\_common cb\_file::common**

Definition at line 600 of file tree.h.

**6.15.2.6   int cb\_file::external**

Definition at line 627 of file tree.h.

**6.15.2.7   int cb\_file::external\_assign**

Definition at line 629 of file tree.h.

**6.15.2.8   cb\_tree cb\_file::file\_status**

Definition at line 605 of file tree.h.

**6.15.2.9   int cb\_file::fileid\_assign**

Definition at line 630 of file tree.h.

**6.15.2.10   int cb\_file::finalized**

Definition at line 626 of file tree.h.

**6.15.2.11   int cb\_file::global**

Definition at line 631 of file tree.h.

**6.15.2.12   struct cb\_label\* cb\_file::handler**

Definition at line 617 of file tree.h.

**6.15.2.13 struct cb\_program\* cb\_file::handler\_prog**

Definition at line 618 of file tree.h.

**6.15.2.14 cb\_tree cb\_file::key**

Definition at line 607 of file tree.h.

**6.15.2.15 cb\_tree cb\_file::latbot**

Definition at line 616 of file tree.h.

**6.15.2.16 cb\_tree cb\_file::latfoot**

Definition at line 614 of file tree.h.

**6.15.2.17 cb\_tree cb\_file::lattop**

Definition at line 615 of file tree.h.

**6.15.2.18 cb\_tree cb\_file::linage**

Definition at line 612 of file tree.h.

**6.15.2.19 cb\_tree cb\_file::linage\_ctr**

Definition at line 613 of file tree.h.

**6.15.2.20 int cb\_file::lock\_mode**

Definition at line 624 of file tree.h.

**6.15.2.21 const char\* cb\_file::name**

Definition at line 601 of file tree.h.

**6.15.2.22 int cb\_file::optional**

Definition at line 621 of file tree.h.

**6.15.2.23 int cb\_file::organization**

Definition at line 622 of file tree.h.

**6.15.2.24 struct cb\_field\* cb\_file::record**

Definition at line 610 of file tree.h.

**6.15.2.25 cb\_tree cb\_file::recordDepending**

Definition at line 611 of file tree.h.

**6.15.2.26 int cb\_file::record\_max**

Definition at line 620 of file tree.h.

**6.15.2.27 int cb\_file::record\_min**

Definition at line 619 of file tree.h.

**6.15.2.28 int cb\_file::same\_clause**

Definition at line 625 of file tree.h.

**6.15.2.29 cb\_tree cb\_file::sharing**

Definition at line 606 of file tree.h.

**6.15.2.30 int cb\_file::special**

Definition at line 628 of file tree.h.

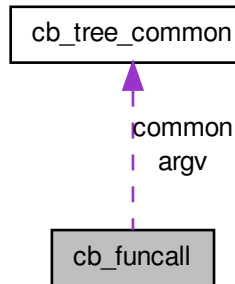
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.16 cb\_funcall Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_funcall`:



### Public Attributes

- struct `cb_tree_common` `common`
- const char \* `name`
- `cb_tree` `argv` [7]
- int `argc`
- int `varcnt`
- size\_t `screenptr`

#### 6.16.1 Detailed Description

Definition at line 725 of file `tree.h`.

#### 6.16.2 Member Data Documentation

##### 6.16.2.1 int `cb_funcall::argc`

Definition at line 729 of file `tree.h`.

##### 6.16.2.2 `cb_tree` `cb_funcall::argv`[7]

Definition at line 728 of file `tree.h`.

##### 6.16.2.3 struct `cb_tree_common` `cb_funcall::common`

Definition at line 726 of file `tree.h`.

#### 6.16.2.4 `const char* cb_funcall::name`

Definition at line 727 of file tree.h.

#### 6.16.2.5 `size_t cb_funcall::screenptr`

Definition at line 731 of file tree.h.

#### 6.16.2.6 `int cb_funcall::varcnt`

Definition at line 730 of file tree.h.

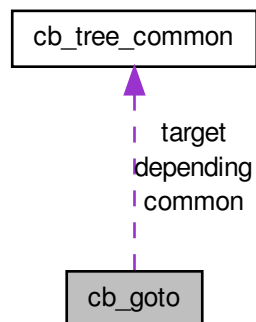
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.17 `cb_goto` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_goto`:



### Public Attributes

- struct [cb\\_tree\\_common](#) common
- [cb\\_tree](#) target
- [cb\\_tree](#) depending



### 6.17.1 Detailed Description

Definition at line 995 of file tree.h.

### 6.17.2 Member Data Documentation

#### 6.17.2.1 struct cb\_tree\_common cb\_goto::common

Definition at line 996 of file tree.h.

#### 6.17.2.2 cb\_tree cb\_goto::depending

Definition at line 998 of file tree.h.

#### 6.17.2.3 cb\_tree cb\_goto::target

Definition at line 997 of file tree.h.

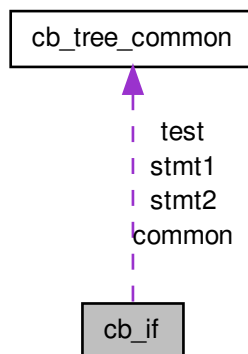
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.18 cb\_if Struct Reference

```
#include <tree.h>
```

Collaboration diagram for cb\_if:



## Public Attributes

- [struct `cb\_tree\_common` `common`](#)
- [cb\\_tree `test`](#)
- [cb\\_tree `stmt1`](#)
- [cb\\_tree `stmt2`](#)

### 6.18.1 Detailed Description

Definition at line 1011 of file `tree.h`.

### 6.18.2 Member Data Documentation

#### 6.18.2.1 `struct cb_tree_common cb_if::common`

Definition at line 1012 of file `tree.h`.

#### 6.18.2.2 `cb_tree cb_if::stmt1`

Definition at line 1014 of file `tree.h`.

#### 6.18.2.3 `cb_tree cb_if::stmt2`

Definition at line 1015 of file `tree.h`.

#### 6.18.2.4 `cb_tree cb_if::test`

Definition at line 1013 of file `tree.h`.

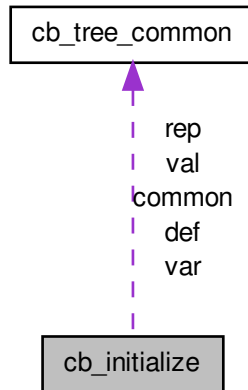
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.19 `cb_initialize` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_initialize`:



### Public Attributes

- struct `cb_tree_common` `common`
- `cb_tree` `var`
- `cb_tree` `val`
- `cb_tree` `rep`
- `cb_tree` `def`
- int `flag_statement`

### 6.19.1 Detailed Description

Definition at line 932 of file `tree.h`.

### 6.19.2 Member Data Documentation

#### 6.19.2.1 struct `cb_tree_common` `cb_initialize::common`

Definition at line 933 of file `tree.h`.

#### 6.19.2.2 `cb_tree` `cb_initialize::def`

Definition at line 937 of file `tree.h`.

### 6.19.2.3 int `cb_initialize::flag_statement`

Definition at line 938 of file `tree.h`.

### 6.19.2.4 `cb_tree` `cb_initialize::rep`

Definition at line 936 of file `tree.h`.

### 6.19.2.5 `cb_tree` `cb_initialize::val`

Definition at line 935 of file `tree.h`.

### 6.19.2.6 `cb_tree` `cb_initialize::var`

Definition at line 934 of file `tree.h`.

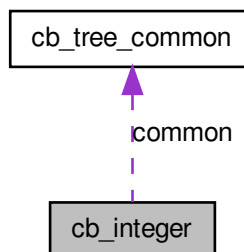
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.20 `cb_integer` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_integer`:



### Public Attributes

- struct `cb_tree_common` `common`
- int `val`

### 6.20.1 Detailed Description

Definition at line 295 of file `tree.h`.

### 6.20.2 Member Data Documentation

#### 6.20.2.1 `struct cb_tree_common cb_integer::common`

Definition at line 296 of file `tree.h`.

#### 6.20.2.2 `int cb_integer::val`

Definition at line 297 of file `tree.h`.

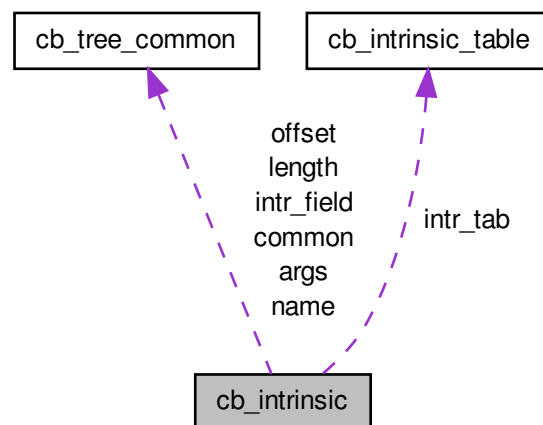
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.21 `cb_intrinsic` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_intrinsic`:



## Public Attributes

- struct [cb\\_tree\\_common](#) `common`
- [cb\\_tree](#) `name`
- [cb\\_tree](#) `args`
- [cb\\_tree](#) `intr_field`
- struct [cb\\_intrinsic\\_table](#) \* `intr_tab`
- [cb\\_tree](#) `offset`
- [cb\\_tree](#) `length`

### 6.21.1 Detailed Description

Definition at line 907 of file `tree.h`.

### 6.21.2 Member Data Documentation

#### 6.21.2.1 `cb_tree` `cb_intrinsic::args`

Definition at line 910 of file `tree.h`.

#### 6.21.2.2 struct `cb_tree_common` `cb_intrinsic::common`

Definition at line 908 of file `tree.h`.

#### 6.21.2.3 `cb_tree` `cb_intrinsic::intr_field`

Definition at line 911 of file `tree.h`.

#### 6.21.2.4 struct `cb_intrinsic_table*` `cb_intrinsic::intr_tab`

Definition at line 912 of file `tree.h`.

#### 6.21.2.5 `cb_tree` `cb_intrinsic::length`

Definition at line 914 of file `tree.h`.

#### 6.21.2.6 `cb_tree` `cb_intrinsic::name`

Definition at line 909 of file `tree.h`.

### 6.21.2.7 `cb_tree` `cb_intrinsic::offset`

Definition at line 913 of file `tree.h`.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.22 `cb_intrinsic_table` Struct Reference

```
#include <tree.h>
```

### Public Attributes

- const char \* [name](#)
- const int [args](#)
- const int [implemented](#)
- enum [cb\\_intr\\_enum](#) [intr\\_enum](#)
- const char \* [intr\\_routine](#)
- enum [cb\\_category](#) [category](#)
- const unsigned int [refmod](#)

### 6.22.1 Detailed Description

Definition at line 897 of file `tree.h`.

### 6.22.2 Member Data Documentation

#### 6.22.2.1 const int `cb_intrinsic_table::args`

Definition at line 899 of file `tree.h`.

#### 6.22.2.2 enum `cb_category` `cb_intrinsic_table::category`

Definition at line 903 of file `tree.h`.

#### 6.22.2.3 const int `cb_intrinsic_table::implemented`

Definition at line 900 of file `tree.h`.

#### 6.22.2.4 enum `cb_intr_enum` `cb_intrinsic_table::intr_enum`

Definition at line 901 of file `tree.h`.

#### 6.22.2.5 `const char* cb_intrinsic_table::intr_routine`

Definition at line 902 of file tree.h.

#### 6.22.2.6 `const char* cb_intrinsic_table::name`

Definition at line 898 of file tree.h.

#### 6.22.2.7 `const unsigned int cb_intrinsic_table::refmod`

Definition at line 904 of file tree.h.

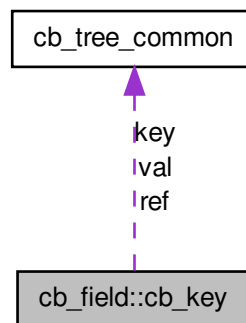
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

### 6.23 `cb_field::cb_key` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_field::cb_key`:



#### Public Attributes

- `int dir`
- `cb_tree key`
- `cb_tree ref`
- `cb_tree val`



### 6.23.1 Detailed Description

Definition at line 486 of file `tree.h`.

### 6.23.2 Member Data Documentation

#### 6.23.2.1 `int cb_field::cb_key::dir`

Definition at line 487 of file `tree.h`.

#### 6.23.2.2 `cb_tree cb_field::cb_key::key`

Definition at line 488 of file `tree.h`.

#### 6.23.2.3 `cb_tree cb_field::cb_key::ref`

Definition at line 489 of file `tree.h`.

#### 6.23.2.4 `cb_tree cb_field::cb_key::val`

Definition at line 490 of file `tree.h`.

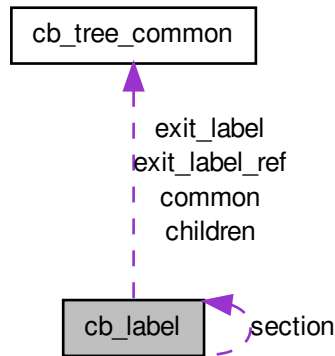
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.24 `cb_label` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_label`:



### Public Attributes

- struct `cb_tree_common` `common`
- const unsigned char \* `name`
- struct `cb_label` \* `section`
- `cb_tree` `exit_label`
- `cb_tree` `exit_label_ref`
- `cb_tree` `children`
- const unsigned char \* `orig_name`
- int `id`
- int `is_section`
- int `is_entry`
- unsigned char `need_begin`
- unsigned char `need_return`
- unsigned char `is_global`
- unsigned char `spare`

### 6.24.1 Detailed Description

Definition at line 561 of file `tree.h`.

### 6.24.2 Member Data Documentation

**6.24.2.1   cb\_tree cb\_label::children**

Definition at line 567 of file tree.h.

**6.24.2.2   struct cb\_tree\_common cb\_label::common**

Definition at line 562 of file tree.h.

**6.24.2.3   cb\_tree cb\_label::exit\_label**

Definition at line 565 of file tree.h.

**6.24.2.4   cb\_tree cb\_label::exit\_label\_ref**

Definition at line 566 of file tree.h.

**6.24.2.5   int cb\_label::id**

Definition at line 569 of file tree.h.

**6.24.2.6   int cb\_label::is\_entry**

Definition at line 571 of file tree.h.

**6.24.2.7   unsigned char cb\_label::is\_global**

Definition at line 574 of file tree.h.

**6.24.2.8   int cb\_label::is\_section**

Definition at line 570 of file tree.h.

**6.24.2.9   const unsigned char\* cb\_label::name**

Definition at line 563 of file tree.h.

**6.24.2.10   unsigned char cb\_label::need\_begin**

Definition at line 572 of file tree.h.

**6.24.2.11 unsigned char `cb_label::need_return`**

Definition at line 573 of file `tree.h`.

**6.24.2.12 const unsigned char\* `cb_label::orig_name`**

Definition at line 568 of file `tree.h`.

**6.24.2.13 struct `cb_label*` `cb_label::section`**

Definition at line 564 of file `tree.h`.

**6.24.2.14 unsigned char `cb_label::spare`**

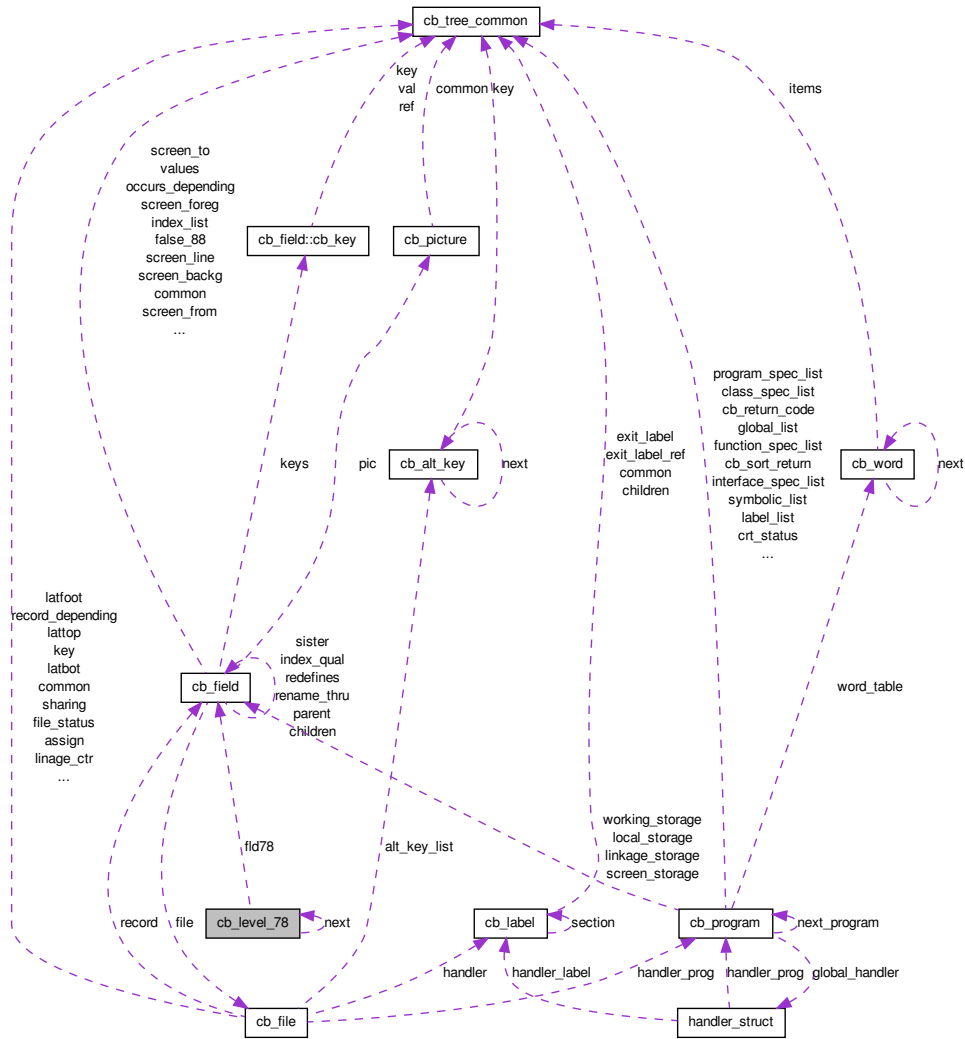
Definition at line 575 of file `tree.h`.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.25 cb\_level\_78 Struct Reference

Collaboration diagram for cb\_level\_78:



### Public Attributes

- struct `cb_level_78` \* `next`
- struct `cb_field` \* `fld78`

### 6.25.1 Detailed Description

Definition at line 901 of file scanner.c.

### 6.25.2 Member Data Documentation

#### 6.25.2.1 struct cb\_field\* cb\_level\_78::fld78

Definition at line 903 of file scanner.c.

#### 6.25.2.2 struct cb\_level\_78\* cb\_level\_78::next

Definition at line 902 of file scanner.c.

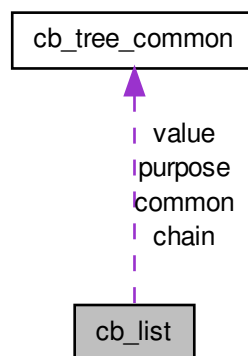
The documentation for this struct was generated from the following file:

- [cobc/scanner.c](#)

## 6.26 cb\_list Struct Reference

```
#include <tree.h>
```

Collaboration diagram for cb\_list:



### Public Attributes

- struct [cb\\_tree\\_common](#) common

- [cb\\_tree purpose](#)
- [cb\\_tree value](#)
- [cb\\_tree chain](#)
- [int sizes](#)

### 6.26.1 Detailed Description

Definition at line 1104 of file `tree.h`.

### 6.26.2 Member Data Documentation

#### 6.26.2.1 `cb_tree cb_list::chain`

Definition at line 1108 of file `tree.h`.

#### 6.26.2.2 `struct cb_tree_common cb_list::common`

Definition at line 1105 of file `tree.h`.

#### 6.26.2.3 `cb_tree cb_list::purpose`

Definition at line 1106 of file `tree.h`.

#### 6.26.2.4 `int cb_list::sizes`

Definition at line 1109 of file `tree.h`.

#### 6.26.2.5 `cb_tree cb_list::value`

Definition at line 1107 of file `tree.h`.

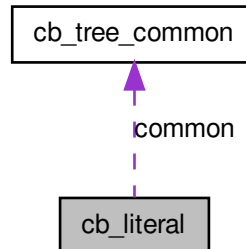
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.27 `cb_literal` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_literal`:



### Public Attributes

- struct `cb_tree_common` `common`
- `size_t` `size`
- unsigned char \* `data`
- signed char `all`
- signed char `sign`
- signed char `scale`
- signed char `spare`

#### 6.27.1 Detailed Description

Definition at line 397 of file `tree.h`.

#### 6.27.2 Member Data Documentation

##### 6.27.2.1 signed char `cb_literal::all`

Definition at line 401 of file `tree.h`.

##### 6.27.2.2 struct `cb_tree_common` `cb_literal::common`

Definition at line 398 of file `tree.h`.

##### 6.27.2.3 unsigned char\* `cb_literal::data`

Definition at line 400 of file `tree.h`.



6.27.2.4 signed char `cb_literal::scale`

Definition at line 403 of file `tree.h`.

6.27.2.5 signed char `cb_literal::sign`

Definition at line 402 of file `tree.h`.

6.27.2.6 `size_t` `cb_literal::size`

Definition at line 399 of file `tree.h`.

6.27.2.7 signed char `cb_literal::spare`

Definition at line 404 of file `tree.h`.

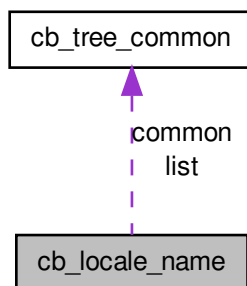
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.28 `cb_locale_name` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_locale_name`:



### Public Attributes

- struct [cb\\_tree\\_common](#) `common`

- `const char * name`
- `char * cname`
- `cb_tree list`

### 6.28.1 Detailed Description

Definition at line 365 of file tree.h.

### 6.28.2 Member Data Documentation

#### 6.28.2.1 `char* cb_locale_name::cname`

Definition at line 368 of file tree.h.

#### 6.28.2.2 `struct cb_tree_common cb_locale_name::common`

Definition at line 366 of file tree.h.

#### 6.28.2.3 `cb_tree cb_locale_name::list`

Definition at line 369 of file tree.h.

#### 6.28.2.4 `const char* cb_locale_name::name`

Definition at line 367 of file tree.h.

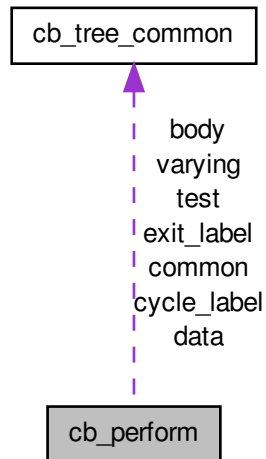
The documentation for this struct was generated from the following file:

- `cobc/tree.h`

## 6.29 `cb_perform` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_perform`:



### Public Attributes

- struct `cb_tree_common` `common`
- enum `cb_perform_type` `type`
- `cb_tree` `test`
- `cb_tree` `body`
- `cb_tree` `data`
- `cb_tree` `varying`
- `cb_tree` `exit_label`
- `cb_tree` `cycle_label`

### 6.29.1 Detailed Description

Definition at line 1044 of file `tree.h`.

### 6.29.2 Member Data Documentation

#### 6.29.2.1 `cb_tree` `cb_perform::body`

Definition at line 1048 of file `tree.h`.

### 6.29.2.2 struct `cb_tree_common` `cb_perform::common`

Definition at line 1045 of file `tree.h`.

### 6.29.2.3 `cb_tree` `cb_perform::cycle_label`

Definition at line 1052 of file `tree.h`.

### 6.29.2.4 `cb_tree` `cb_perform::data`

Definition at line 1049 of file `tree.h`.

### 6.29.2.5 `cb_tree` `cb_perform::exit_label`

Definition at line 1051 of file `tree.h`.

### 6.29.2.6 `cb_tree` `cb_perform::test`

Definition at line 1047 of file `tree.h`.

### 6.29.2.7 enum `cb_perform_type` `cb_perform::type`

Definition at line 1046 of file `tree.h`.

### 6.29.2.8 `cb_tree` `cb_perform::varying`

Definition at line 1050 of file `tree.h`.

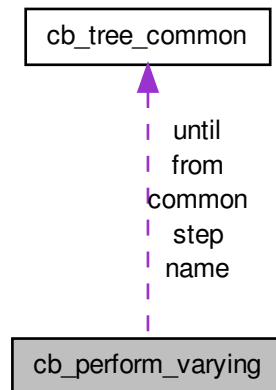
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.30 `cb_perform_varying` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_perform_varying`:



### Public Attributes

- struct `cb_tree_common` `common`
- `cb_tree` `name`
- `cb_tree` `from`
- `cb_tree` `step`
- `cb_tree` `until`

### 6.30.1 Detailed Description

Definition at line 1036 of file `tree.h`.

### 6.30.2 Member Data Documentation

#### 6.30.2.1 struct `cb_tree_common` `cb_perform_varying::common`

Definition at line 1037 of file `tree.h`.

#### 6.30.2.2 `cb_tree` `cb_perform_varying::from`

Definition at line 1039 of file `tree.h`.

### 6.30.2.3 `cb_tree cb_perform_varying::name`

Definition at line 1038 of file tree.h.

### 6.30.2.4 `cb_tree cb_perform_varying::step`

Definition at line 1040 of file tree.h.

### 6.30.2.5 `cb_tree cb_perform_varying::until`

Definition at line 1041 of file tree.h.

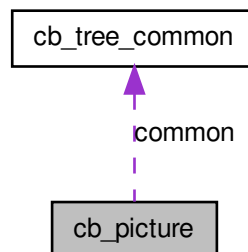
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.31 `cb_picture` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_picture`:



### Public Attributes

- struct [cb\\_tree\\_common](#) `common`
- int `size`
- int `lenstr`
- char \* `orig`
- char \* `str`

- enum `cb_category` `category`
- unsigned char `digits`
- signed char `scale`
- unsigned char `have_sign`
- unsigned char `spare`

### 6.31.1 Detailed Description

Definition at line 436 of file `tree.h`.

### 6.31.2 Member Data Documentation

#### 6.31.2.1 enum `cb_category` `cb_picture::category`

Definition at line 442 of file `tree.h`.

#### 6.31.2.2 struct `cb_tree_common` `cb_picture::common`

Definition at line 437 of file `tree.h`.

#### 6.31.2.3 unsigned char `cb_picture::digits`

Definition at line 443 of file `tree.h`.

#### 6.31.2.4 unsigned char `cb_picture::have_sign`

Definition at line 445 of file `tree.h`.

#### 6.31.2.5 int `cb_picture::lenstr`

Definition at line 439 of file `tree.h`.

#### 6.31.2.6 char\* `cb_picture::orig`

Definition at line 440 of file `tree.h`.

#### 6.31.2.7 signed char `cb_picture::scale`

Definition at line 444 of file `tree.h`.

#### 6.31.2.8 int cb\_picture::size

Definition at line 438 of file tree.h.

#### 6.31.2.9 unsigned char cb\_picture::spare

Definition at line 446 of file tree.h.

#### 6.31.2.10 char\* cb\_picture::str

Definition at line 441 of file tree.h.

The documentation for this struct was generated from the following file:

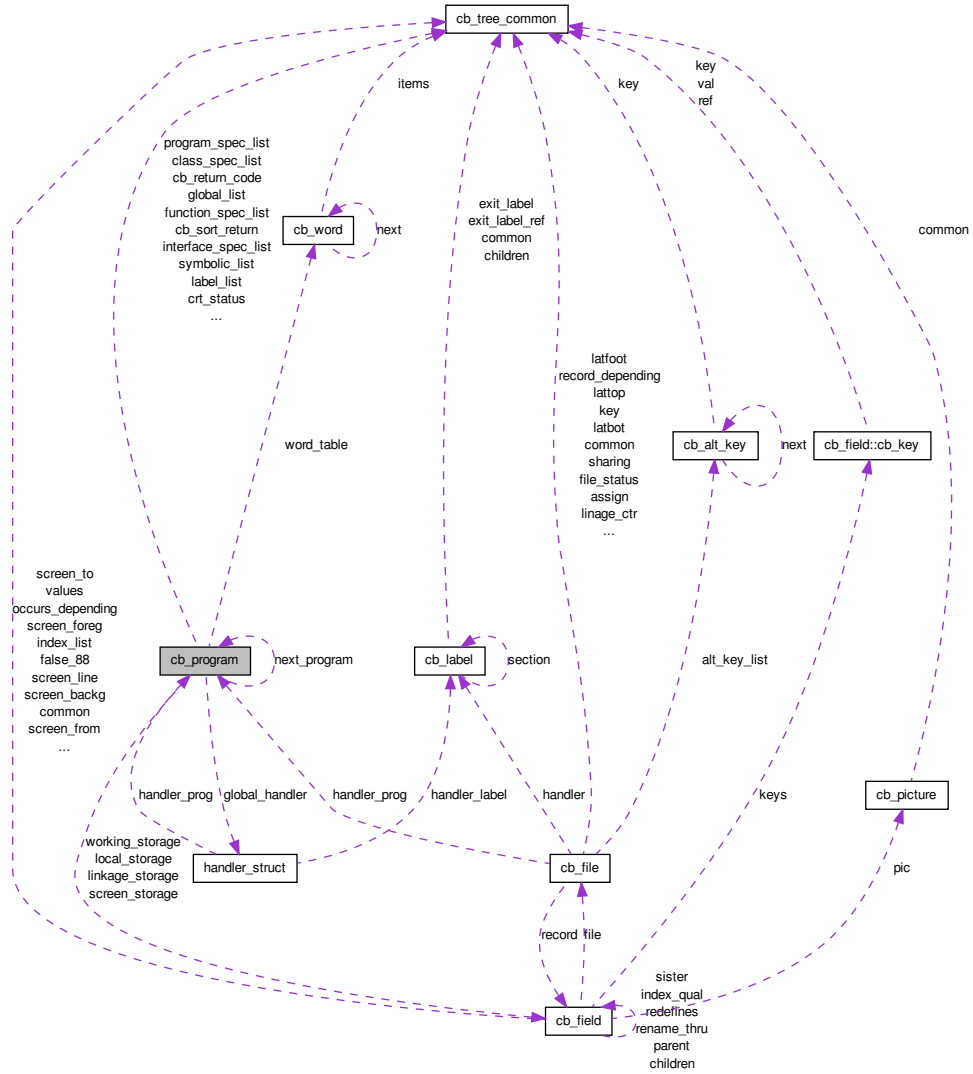
- [cobc/tree.h](#)

## 6.32 cb\_program Struct Reference

```
#include <tree.h>
```



Collaboration diagram for `cb_program`:



## Public Attributes

- struct `cb_program` \* `next_program`
- const char \* `program_id`
- char \* `source_name`
- char \* `orig_source_name`
- FILE \* `local_storage_file`
- char \* `local_storage_name`

- [cb\\_tree entry\\_list](#)
- [cb\\_tree file\\_list](#)
- [cb\\_tree exec\\_list](#)
- [cb\\_tree label\\_list](#)
- [cb\\_tree reference\\_list](#)
- [cb\\_tree alphabet\\_name\\_list](#)
- [cb\\_tree class\\_name\\_list](#)
- [cb\\_tree parameter\\_list](#)
- [cb\\_tree locale\\_list](#)
- [cb\\_tree symbolic\\_list](#)
- [cb\\_tree global\\_list](#)
- [cb\\_tree cb\\_return\\_code](#)
- [cb\\_tree cb\\_sort\\_return](#)
- [cb\\_tree cb\\_call\\_params](#)
- [cb\\_tree class\\_spec\\_list](#)
- [cb\\_tree interface\\_spec\\_list](#)
- [cb\\_tree function\\_spec\\_list](#)
- [cb\\_tree program\\_spec\\_list](#)
- [cb\\_tree property\\_spec\\_list](#)
- [struct cb\\_field \\* working\\_storage](#)
- [struct cb\\_field \\* local\\_storage](#)
- [struct cb\\_field \\* linkage\\_storage](#)
- [struct cb\\_field \\* screen\\_storage](#)
- [cb\\_tree local\\_file\\_list](#)
- [cb\\_tree global\\_file\\_list](#)
- [struct handler\\_struct global\\_handler \[5\]](#)
- [cb\\_tree collating\\_sequence](#)
- [cb\\_tree cursor\\_pos](#)
- [cb\\_tree crt\\_status](#)
- [cb\\_tree returning](#)
- [struct cb\\_word \\* word\\_table \[CB\\_WORD\\_HASH\\_SIZE\]](#)
- [int loop\\_counter](#)
- [int decimal\\_index](#)
- [int decimal\\_index\\_max](#)
- [unsigned char decimal\\_point](#)
- [unsigned char currency\\_symbol](#)
- [unsigned char numeric\\_separator](#)
- [unsigned char nested\\_level](#)
- [unsigned char flag\\_main](#)
- [unsigned char flag\\_common](#)
- [unsigned char flag\\_initial](#)
- [unsigned char flag\\_recursive](#)
- [unsigned char flag\\_screen](#)
- [unsigned char flag\\_validated](#)
- [unsigned char flag\\_chained](#)
- [unsigned char flag\\_global\\_use](#)

- unsigned char `gen_decset`
- unsigned char `gen_udecset`
- unsigned char `gen_ptrmanip`
- unsigned char `gen_file_error`
- unsigned char `prog_type`
- unsigned char `spare` [3]

### 6.32.1 Detailed Description

Definition at line 1154 of file `tree.h`.

### 6.32.2 Member Data Documentation

#### 6.32.2.1 `cb_tree cb_program::alphabet_name_list`

Definition at line 1167 of file `tree.h`.

#### 6.32.2.2 `cb_tree cb_program::cb_call_params`

Definition at line 1175 of file `tree.h`.

#### 6.32.2.3 `cb_tree cb_program::cb_return_code`

Definition at line 1173 of file `tree.h`.

#### 6.32.2.4 `cb_tree cb_program::cb_sort_return`

Definition at line 1174 of file `tree.h`.

#### 6.32.2.5 `cb_tree cb_program::class_name_list`

Definition at line 1168 of file `tree.h`.

#### 6.32.2.6 `cb_tree cb_program::class_spec_list`

Definition at line 1176 of file `tree.h`.

#### 6.32.2.7 `cb_tree cb_program::collating_sequence`

Definition at line 1188 of file `tree.h`.

**6.32.2.8   cb\_tree   cb\_program::crt\_status**

Definition at line 1190 of file tree.h.

**6.32.2.9   unsigned char   cb\_program::currency\_symbol**

Definition at line 1198 of file tree.h.

**6.32.2.10   cb\_tree   cb\_program::cursor\_pos**

Definition at line 1189 of file tree.h.

**6.32.2.11   int   cb\_program::decimal\_index**

Definition at line 1195 of file tree.h.

**6.32.2.12   int   cb\_program::decimal\_index\_max**

Definition at line 1196 of file tree.h.

**6.32.2.13   unsigned char   cb\_program::decimal\_point**

Definition at line 1197 of file tree.h.

**6.32.2.14   cb\_tree   cb\_program::entry\_list**

Definition at line 1162 of file tree.h.

**6.32.2.15   cb\_tree   cb\_program::exec\_list**

Definition at line 1164 of file tree.h.

**6.32.2.16   cb\_tree   cb\_program::file\_list**

Definition at line 1163 of file tree.h.

**6.32.2.17   unsigned char   cb\_program::flag\_chained**

Definition at line 1207 of file tree.h.

**6.32.2.18 unsigned char cb\_program::flag\_common**

Definition at line 1202 of file tree.h.

**6.32.2.19 unsigned char cb\_program::flag\_global\_use**

Definition at line 1208 of file tree.h.

**6.32.2.20 unsigned char cb\_program::flag\_initial**

Definition at line 1203 of file tree.h.

**6.32.2.21 unsigned char cb\_program::flag\_main**

Definition at line 1201 of file tree.h.

**6.32.2.22 unsigned char cb\_program::flag\_recursive**

Definition at line 1204 of file tree.h.

**6.32.2.23 unsigned char cb\_program::flag\_screen**

Definition at line 1205 of file tree.h.

**6.32.2.24 unsigned char cb\_program::flag\_validated**

Definition at line 1206 of file tree.h.

**6.32.2.25 cb\_tree cb\_program::function\_spec\_list**

Definition at line 1178 of file tree.h.

**6.32.2.26 unsigned char cb\_program::gen\_decset**

Definition at line 1209 of file tree.h.

**6.32.2.27 unsigned char cb\_program::gen\_file\_error**

Definition at line 1212 of file tree.h.

**6.32.2.28 unsigned char `cb_program::gen_ptrmanip`**

Definition at line 1211 of file tree.h.

**6.32.2.29 unsigned char `cb_program::gen_udecset`**

Definition at line 1210 of file tree.h.

**6.32.2.30 `cb_tree` `cb_program::global_file_list`**

Definition at line 1186 of file tree.h.

**6.32.2.31 `struct handler_struct` `cb_program::global_handler[5]`**

Definition at line 1187 of file tree.h.

**6.32.2.32 `cb_tree` `cb_program::global_list`**

Definition at line 1172 of file tree.h.

**6.32.2.33 `cb_tree` `cb_program::interface_spec_list`**

Definition at line 1177 of file tree.h.

**6.32.2.34 `cb_tree` `cb_program::label_list`**

Definition at line 1165 of file tree.h.

**6.32.2.35 `struct cb_field*` `cb_program::linkage_storage`**

Definition at line 1183 of file tree.h.

**6.32.2.36 `cb_tree` `cb_program::local_file_list`**

Definition at line 1185 of file tree.h.

**6.32.2.37 `struct cb_field*` `cb_program::local_storage`**

Definition at line 1182 of file tree.h.

**6.32.2.38 FILE\* cb\_program::local\_storage\_file**

Definition at line 1160 of file tree.h.

**6.32.2.39 char\* cb\_program::local\_storage\_name**

Definition at line 1161 of file tree.h.

**6.32.2.40 cb\_tree cb\_program::locale\_list**

Definition at line 1170 of file tree.h.

**6.32.2.41 int cb\_program::loop\_counter**

Definition at line 1194 of file tree.h.

**6.32.2.42 unsigned char cb\_program::nested\_level**

Definition at line 1200 of file tree.h.

**6.32.2.43 struct cb\_program\* cb\_program::next\_program**

Definition at line 1156 of file tree.h.

**6.32.2.44 unsigned char cb\_program::numeric\_separator**

Definition at line 1199 of file tree.h.

**6.32.2.45 char\* cb\_program::orig\_source\_name**

Definition at line 1159 of file tree.h.

**6.32.2.46 cb\_tree cb\_program::parameter\_list**

Definition at line 1169 of file tree.h.

**6.32.2.47 unsigned char cb\_program::prog\_type**

Definition at line 1213 of file tree.h.

**6.32.2.48 const char\* cb\_program::program\_id**

Definition at line 1157 of file tree.h.

**6.32.2.49 cb\_tree cb\_program::program\_spec\_list**

Definition at line 1179 of file tree.h.

**6.32.2.50 cb\_tree cb\_program::property\_spec\_list**

Definition at line 1180 of file tree.h.

**6.32.2.51 cb\_tree cb\_program::reference\_list**

Definition at line 1166 of file tree.h.

**6.32.2.52 cb\_tree cb\_program::returning**

Definition at line 1191 of file tree.h.

**6.32.2.53 struct cb\_field\* cb\_program::screen\_storage**

Definition at line 1184 of file tree.h.

**6.32.2.54 char\* cb\_program::source\_name**

Definition at line 1158 of file tree.h.

**6.32.2.55 unsigned char cb\_program::spare[3]**

Definition at line 1214 of file tree.h.

**6.32.2.56 cb\_tree cb\_program::symbolic\_list**

Definition at line 1171 of file tree.h.

**6.32.2.57 struct cb\_word\* cb\_program::word\_table[CB\_WORD\_HASH\_SIZE]**

Definition at line 1192 of file tree.h.



6.32.2.58 `struct cb_field* cb_program::working_storage`

Definition at line 1181 of file `tree.h`.

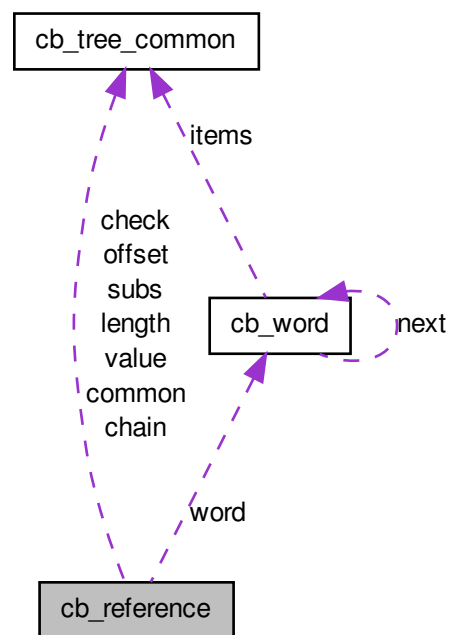
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

6.33 `cb_reference` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_reference`:



## Public Attributes

- struct `cb_tree_common` `common`
- struct `cb_word` \* `word`
- enum `cb_operand_type` `type`

- [cb\\_tree value](#)
- [cb\\_tree subs](#)
- [cb\\_tree offset](#)
- [cb\\_tree length](#)
- [cb\\_tree check](#)
- [cb\\_tree chain](#)
- [int all](#)

### 6.33.1 Detailed Description

Definition at line 656 of file tree.h.

### 6.33.2 Member Data Documentation

#### 6.33.2.1 `int cb_reference::all`

Definition at line 666 of file tree.h.

#### 6.33.2.2 `cb_tree cb_reference::chain`

Definition at line 665 of file tree.h.

#### 6.33.2.3 `cb_tree cb_reference::check`

Definition at line 664 of file tree.h.

#### 6.33.2.4 `struct cb_tree_common cb_reference::common`

Definition at line 657 of file tree.h.

#### 6.33.2.5 `cb_tree cb_reference::length`

Definition at line 663 of file tree.h.

#### 6.33.2.6 `cb_tree cb_reference::offset`

Definition at line 662 of file tree.h.

#### 6.33.2.7 `cb_tree cb_reference::subs`

Definition at line 661 of file tree.h.

6.33.2.8 `enum cb_operand_type cb_reference::type`

Definition at line 659 of file `tree.h`.

6.33.2.9 `cb_tree cb_reference::value`

Definition at line 660 of file `tree.h`.

6.33.2.10 `struct cb_word* cb_reference::word`

Definition at line 658 of file `tree.h`.

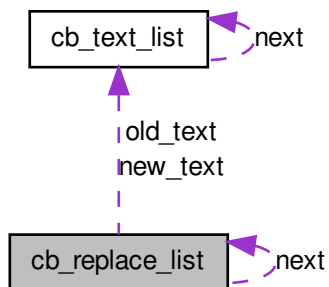
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

6.34 `cb_replace_list` Struct Reference

```
#include <cobc.h>
```

Collaboration diagram for `cb_replace_list`:



## Public Attributes

- struct `cb_text_list` \* `old_text`
- struct `cb_text_list` \* `new_text`
- struct `cb_replace_list` \* `next`

### 6.34.1 Detailed Description

Definition at line 71 of file cobic.h.

### 6.34.2 Member Data Documentation

#### 6.34.2.1 struct `cb_text_list*` `cb_replace_list::new_text`

Definition at line 73 of file cobic.h.

#### 6.34.2.2 struct `cb_replace_list*` `cb_replace_list::next`

Definition at line 74 of file cobic.h.

#### 6.34.2.3 struct `cb_text_list*` `cb_replace_list::old_text`

Definition at line 72 of file cobic.h.

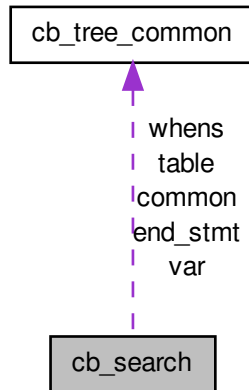
The documentation for this struct was generated from the following file:

- [cobic/cobic.h](#)

## 6.35 `cb_search` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_search`:



### Public Attributes

- struct `cb_tree_common` `common`
- int `flag_all`
- `cb_tree` `table`
- `cb_tree` `var`
- `cb_tree` `end_stmt`
- `cb_tree` `whens`

#### 6.35.1 Detailed Description

Definition at line 951 of file `tree.h`.

#### 6.35.2 Member Data Documentation

##### 6.35.2.1 struct `cb_tree_common` `cb_search::common`

Definition at line 952 of file `tree.h`.

##### 6.35.2.2 `cb_tree` `cb_search::end_stmt`

Definition at line 956 of file `tree.h`.

### 6.35.2.3 int cb\_search::flag\_all

Definition at line 953 of file tree.h.

### 6.35.2.4 cb\_tree cb\_search::table

Definition at line 954 of file tree.h.

### 6.35.2.5 cb\_tree cb\_search::var

Definition at line 955 of file tree.h.

### 6.35.2.6 cb\_tree cb\_search::whens

Definition at line 957 of file tree.h.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.36 cb\_statement Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_statement`:



### Public Attributes

- struct `cb_tree_common` `common`
- const char \* `name`
- `cb_tree` `body`
- `cb_tree` `file`
- `cb_tree` `handler1`
- `cb_tree` `handler2`
- `cb_tree` `handler3`
- `cb_tree` `null_check`
- int `handler_id`
- int `need_terminator`

### 6.36.1 Detailed Description

Definition at line 1067 of file `tree.h`.

### 6.36.2 Member Data Documentation

#### 6.36.2.1 `cb_tree` `cb_statement::body`

Definition at line 1070 of file `tree.h`.

**6.36.2.2 struct `cb_tree_common` `cb_statement::common`**

Definition at line 1068 of file `tree.h`.

**6.36.2.3 `cb_tree` `cb_statement::file`**

Definition at line 1071 of file `tree.h`.

**6.36.2.4 `cb_tree` `cb_statement::handler1`**

Definition at line 1072 of file `tree.h`.

**6.36.2.5 `cb_tree` `cb_statement::handler2`**

Definition at line 1073 of file `tree.h`.

**6.36.2.6 `cb_tree` `cb_statement::handler3`**

Definition at line 1074 of file `tree.h`.

**6.36.2.7 int `cb_statement::handler_id`**

Definition at line 1076 of file `tree.h`.

**6.36.2.8 const char\* `cb_statement::name`**

Definition at line 1069 of file `tree.h`.

**6.36.2.9 int `cb_statement::need_terminator`**

Definition at line 1077 of file `tree.h`.

**6.36.2.10 `cb_tree` `cb_statement::null_check`**

Definition at line 1075 of file `tree.h`.

The documentation for this struct was generated from the following file:

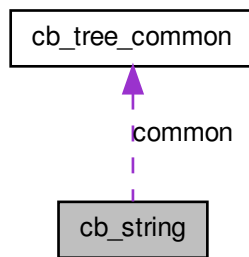
- [cobc/tree.h](#)



## 6.37 `cb_string` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_string`:



### Public Attributes

- struct `cb_tree_common` `common`
- `size_t` `size`
- `const unsigned char *` `data`

#### 6.37.1 Detailed Description

Definition at line 310 of file `tree.h`.

#### 6.37.2 Member Data Documentation

##### 6.37.2.1 `struct cb_tree_common cb_string::common`

Definition at line 311 of file `tree.h`.

##### 6.37.2.2 `const unsigned char* cb_string::data`

Definition at line 313 of file `tree.h`.

##### 6.37.2.3 `size_t cb_string::size`

Definition at line 312 of file `tree.h`.

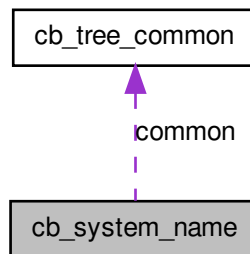
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.38 `cb_system_name` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_system_name`:



### Public Attributes

- struct [cb\\_tree\\_common](#) `common`
- enum [cb\\_system\\_name\\_category](#) `category`
- int `token`

#### 6.38.1 Detailed Description

Definition at line 381 of file `tree.h`.

#### 6.38.2 Member Data Documentation

##### 6.38.2.1 enum `cb_system_name_category` `cb_system_name::category`

Definition at line 383 of file `tree.h`.

##### 6.38.2.2 struct `cb_tree_common` `cb_system_name::common`

Definition at line 382 of file `tree.h`.

### 6.38.2.3 `int cb_system_name::token`

Definition at line 384 of file `tree.h`.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.39 `cb_text_list` Struct Reference

```
#include <cobc.h>
```

Collaboration diagram for `cb_text_list`:



### Public Attributes

- `const char * text`
- `struct cb_text_list * next`

### 6.39.1 Detailed Description

Definition at line 66 of file `cobc.h`.

### 6.39.2 Member Data Documentation

#### 6.39.2.1 `struct cb_text_list* cb_text_list::next`

Definition at line 68 of file `cobc.h`.

#### 6.39.2.2 `const char* cb_text_list::text`

Definition at line 67 of file `cobc.h`.

The documentation for this struct was generated from the following file:

- [cobc/cobc.h](#)

## 6.40 `cb_tree_common` Struct Reference

```
#include <tree.h>
```

### Public Attributes

- enum [cb\\_tag](#) tag
- enum [cb\\_category](#) category
- unsigned char \* [source\\_file](#)
- int [source\\_line](#)

### 6.40.1 Detailed Description

Definition at line 210 of file tree.h.

### 6.40.2 Member Data Documentation

#### 6.40.2.1 enum `cb_category` `cb_tree_common::category`

Definition at line 212 of file tree.h.

#### 6.40.2.2 unsigned char\* `cb_tree_common::source_file`

Definition at line 213 of file tree.h.

#### 6.40.2.3 int `cb_tree_common::source_line`

Definition at line 214 of file tree.h.

#### 6.40.2.4 enum `cb_tag` `cb_tree_common::tag`

Definition at line 211 of file tree.h.

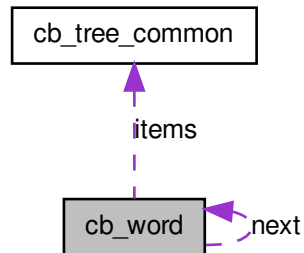
The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.41 `cb_word` Struct Reference

```
#include <tree.h>
```

Collaboration diagram for `cb_word`:



### Public Attributes

- struct `cb_word` \* `next`
- const char \* `name`
- `cb_tree` `items`
- int `count`
- int `error`

#### 6.41.1 Detailed Description

Definition at line 648 of file `tree.h`.

#### 6.41.2 Member Data Documentation

##### 6.41.2.1 int `cb_word::count`

Definition at line 652 of file `tree.h`.

##### 6.41.2.2 int `cb_word::error`

Definition at line 653 of file `tree.h`.

##### 6.41.2.3 `cb_tree` `cb_word::items`

Definition at line 651 of file `tree.h`.

#### 6.41.2.4 `const char* cb_word::name`

Definition at line 650 of file tree.h.

#### 6.41.2.5 `struct cb_word* cb_word::next`

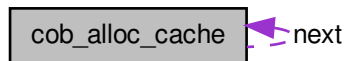
Definition at line 649 of file tree.h.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.42 `cob_alloc_cache` Struct Reference

Collaboration diagram for `cob_alloc_cache`:



### Public Attributes

- struct [cob\\_alloc\\_cache](#) \* [next](#)
- void \* [cob\\_pointer](#)
- [size\\_t](#) [size](#)

#### 6.42.1 Detailed Description

Definition at line 69 of file common.c.

#### 6.42.2 Member Data Documentation

##### 6.42.2.1 `void* cob_alloc_cache::cob_pointer`

Definition at line 71 of file common.c.

#### 6.42.2.2 `struct cob_alloc_cache*` `cob_alloc_cache::next`

Definition at line 70 of file `common.c`.

#### 6.42.2.3 `size_t` `cob_alloc_cache::size`

Definition at line 72 of file `common.c`.

The documentation for this struct was generated from the following file:

- [libcob/common.c](#)

## 6.43 `cob_decimal` Struct Reference

```
#include <numeric.h>
```

### Public Attributes

- `mpz_t` [value](#)
- `int` [scale](#)

#### 6.43.1 Detailed Description

Definition at line 36 of file `numeric.h`.

#### 6.43.2 Member Data Documentation

##### 6.43.2.1 `int` `cob_decimal::scale`

Definition at line 38 of file `numeric.h`.

##### 6.43.2.2 `mpz_t` `cob_decimal::value`

Definition at line 37 of file `numeric.h`.

The documentation for this struct was generated from the following file:

- [libcob/numeric.h](#)

## 6.44 `cob_exception` Struct Reference

## Public Attributes

- const char \* [name](#)
- const int [code](#)
- const int [critical](#)

### 6.44.1 Detailed Description

Definition at line 63 of file common.c.

### 6.44.2 Member Data Documentation

#### 6.44.2.1 const int `cob_exception::code`

Definition at line 65 of file common.c.

#### 6.44.2.2 const int `cob_exception::critical`

Definition at line 66 of file common.c.

#### 6.44.2.3 const char\* `cob_exception::name`

Definition at line 64 of file common.c.

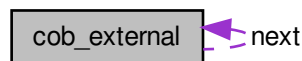
The documentation for this struct was generated from the following file:

- [libcob/common.c](#)

## 6.45 `cob_external` Struct Reference

```
#include <common.h>
```

Collaboration diagram for `cob_external`:





## Public Attributes

- struct `cob_external` \* `next`
- char \* `ext_alloc`
- char \* `ename`
- int `esize`

### 6.45.1 Detailed Description

Definition at line 98 of file `common.h`.

### 6.45.2 Member Data Documentation

#### 6.45.2.1 char\* `cob_external::ename`

Definition at line 101 of file `common.h`.

#### 6.45.2.2 int `cob_external::esize`

Definition at line 102 of file `common.h`.

#### 6.45.2.3 char\* `cob_external::ext_alloc`

Definition at line 100 of file `common.h`.

#### 6.45.2.4 struct `cob_external`\* `cob_external::next`

Definition at line 99 of file `common.h`.

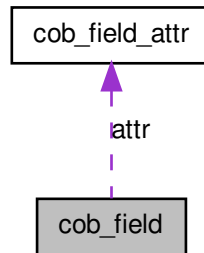
The documentation for this struct was generated from the following file:

- `libcob/common.h`

## 6.46 `cob_field` Struct Reference

```
#include <common.h>
```

Collaboration diagram for `cob_field`:



### Public Attributes

- `size_t` `size`
- `unsigned char *` `data`
- `const cob_field_attr *` `attr`

#### 6.46.1 Detailed Description

Definition at line 165 of file `common.h`.

#### 6.46.2 Member Data Documentation

##### 6.46.2.1 `const cob_field_attr* cob_field::attr`

Definition at line 168 of file `common.h`.

##### 6.46.2.2 `unsigned char* cob_field::data`

Definition at line 167 of file `common.h`.

##### 6.46.2.3 `size_t cob_field::size`

Definition at line 166 of file `common.h`.

The documentation for this struct was generated from the following file:

- `libcob/common.h`

## 6.47 `cob_field_attr` Struct Reference

```
#include <common.h>
```

### Public Attributes

- unsigned char [type](#)
- unsigned char [digits](#)
- signed char [scale](#)
- unsigned char [flags](#)
- const char \* [pic](#)

### 6.47.1 Detailed Description

Definition at line 155 of file `common.h`.

### 6.47.2 Member Data Documentation

#### 6.47.2.1 unsigned char `cob_field_attr::digits`

Definition at line 157 of file `common.h`.

#### 6.47.2.2 unsigned char `cob_field_attr::flags`

Definition at line 159 of file `common.h`.

#### 6.47.2.3 const char\* `cob_field_attr::pic`

Definition at line 160 of file `common.h`.

#### 6.47.2.4 signed char `cob_field_attr::scale`

Definition at line 158 of file `common.h`.

#### 6.47.2.5 unsigned char `cob_field_attr::type`

Definition at line 156 of file `common.h`.

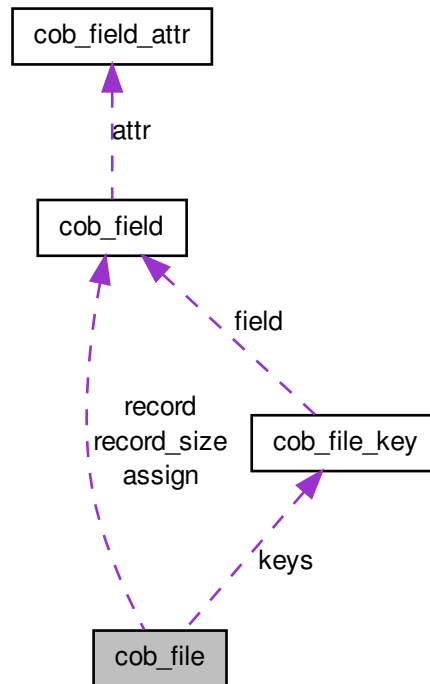
The documentation for this struct was generated from the following file:

- [libcob/common.h](#)

## 6.48 cob\_file Struct Reference

```
#include <fileio.h>
```

Collaboration diagram for cob\_file:



### Public Attributes

- `const char * select_name`
- `unsigned char * file_status`
- `cob_field * assign`
- `cob_field * record`
- `cob_field * record_size`
- `struct cob_file_key * keys`
- `void * file`
- `void * linorkeyptr`
- `const unsigned char * sort_collating`
- `void * extfh_ptr`

- [size\\_t record\\_min](#)
- [size\\_t record\\_max](#)
- [size\\_t nkeys](#)
- [char organization](#)
- [char access\\_mode](#)
- [char lock\\_mode](#)
- [char open\\_mode](#)
- [char flag\\_optional](#)
- [char last\\_open\\_mode](#)
- [char special](#)
- [char flag\\_nonexistent](#)
- [char flag\\_end\\_of\\_file](#)
- [char flag\\_begin\\_of\\_file](#)
- [char flag\\_first\\_read](#)
- [char flag\\_read\\_done](#)
- [char flag\\_select\\_features](#)
- [char flag\\_needs\\_nl](#)
- [char flag\\_needs\\_top](#)
- [char file\\_version](#)

### 6.48.1 Detailed Description

Definition at line 171 of file fileio.h.

### 6.48.2 Member Data Documentation

#### 6.48.2.1 `char cob_file::access_mode`

Definition at line 186 of file fileio.h.

#### 6.48.2.2 `cob_field* cob_file::assign`

Definition at line 174 of file fileio.h.

#### 6.48.2.3 `void* cob_file::extfh_ptr`

Definition at line 181 of file fileio.h.

#### 6.48.2.4 `void* cob_file::file`

Definition at line 178 of file fileio.h.

**6.48.2.5 unsigned char\* cob\_file::file\_status**

Definition at line 173 of file fileio.h.

**6.48.2.6 char cob\_file::file\_version**

Definition at line 200 of file fileio.h.

**6.48.2.7 char cob\_file::flag\_begin\_of\_file**

Definition at line 194 of file fileio.h.

**6.48.2.8 char cob\_file::flag\_end\_of\_file**

Definition at line 193 of file fileio.h.

**6.48.2.9 char cob\_file::flag\_first\_read**

Definition at line 195 of file fileio.h.

**6.48.2.10 char cob\_file::flag\_needs\_nl**

Definition at line 198 of file fileio.h.

**6.48.2.11 char cob\_file::flag\_needs\_top**

Definition at line 199 of file fileio.h.

**6.48.2.12 char cob\_file::flag\_nonexistent**

Definition at line 192 of file fileio.h.

**6.48.2.13 char cob\_file::flag\_optional**

Definition at line 189 of file fileio.h.

**6.48.2.14 char cob\_file::flag\_read\_done**

Definition at line 196 of file fileio.h.

**6.48.2.15 char cob\_file::flag\_select\_features**

Definition at line 197 of file fileio.h.

**6.48.2.16 struct cob\_file\_key\* cob\_file::keys**

Definition at line 177 of file fileio.h.

**6.48.2.17 char cob\_file::last\_open\_mode**

Definition at line 190 of file fileio.h.

**6.48.2.18 void\* cob\_file::linorkeypnr**

Definition at line 179 of file fileio.h.

**6.48.2.19 char cob\_file::lock\_mode**

Definition at line 187 of file fileio.h.

**6.48.2.20 size\_t cob\_file::nkeys**

Definition at line 184 of file fileio.h.

**6.48.2.21 char cob\_file::open\_mode**

Definition at line 188 of file fileio.h.

**6.48.2.22 char cob\_file::organization**

Definition at line 185 of file fileio.h.

**6.48.2.23 cob\_field\* cob\_file::record**

Definition at line 175 of file fileio.h.

**6.48.2.24 size\_t cob\_file::record\_max**

Definition at line 183 of file fileio.h.

**6.48.2.25** `size_t cob_file::record_min`

Definition at line 182 of file fileio.h.

**6.48.2.26** `cob_field* cob_file::record_size`

Definition at line 176 of file fileio.h.

**6.48.2.27** `const char* cob_file::select_name`

Definition at line 172 of file fileio.h.

**6.48.2.28** `const unsigned char* cob_file::sort_collating`

Definition at line 180 of file fileio.h.

**6.48.2.29** `char cob_file::special`

Definition at line 191 of file fileio.h.

The documentation for this struct was generated from the following file:

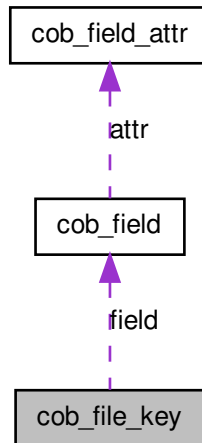
- [libcob/fileio.h](#)

## 6.49 `cob_file_key` Struct Reference

```
#include <fileio.h>
```



Collaboration diagram for cob\_file\_key:



### Public Attributes

- `cob_field * field`
- `int flag`
- `size_t offset`

#### 6.49.1 Detailed Description

Definition at line 152 of file fileio.h.

#### 6.49.2 Member Data Documentation

##### 6.49.2.1 `cob_field* cob_file_key::field`

Definition at line 153 of file fileio.h.

##### 6.49.2.2 `int cob_file_key::flag`

Definition at line 154 of file fileio.h.

### 6.49.2.3 `size_t cob_file_key::offset`

Definition at line 156 of file `fileio.h`.

The documentation for this struct was generated from the following file:

- [libcob/fileio.h](#)

## 6.50 `cob_fileio_funcs` Struct Reference

```
#include <fileio.h>
```

### Public Attributes

- `int(* open)(cob_file *, char *, const int, const int)`
- `int(* close)(cob_file *, const int)`
- `int(* start)(cob_file *, const int, cob_field *)`
- `int(* read)(cob_file *, cob_field *, int)`
- `int(* read_next)(cob_file *, int)`
- `int(* write)(cob_file *, const int)`
- `int(* rewrite)(cob_file *, const int)`
- `int(* fdelete)(cob_file *)`

### 6.50.1 Detailed Description

Definition at line 216 of file `fileio.h`.

### 6.50.2 Member Data Documentation

#### 6.50.2.1 `int(* cob_fileio_funcs::close)(cob_file *, const int)`

Definition at line 218 of file `fileio.h`.

#### 6.50.2.2 `int(* cob_fileio_funcs::fdelete)(cob_file *)`

Definition at line 224 of file `fileio.h`.

#### 6.50.2.3 `int(* cob_fileio_funcs::open)(cob_file *, char *, const int, const int)`

Definition at line 217 of file `fileio.h`.

**6.50.2.4** `int(* cob_fileio_funcs::read)(cob_file *, cob_field *, int)`

Definition at line 220 of file fileio.h.

**6.50.2.5** `int(* cob_fileio_funcs::read_next)(cob_file *, int)`

Definition at line 221 of file fileio.h.

**6.50.2.6** `int(* cob_fileio_funcs::rewrite)(cob_file *, const int)`

Definition at line 223 of file fileio.h.

**6.50.2.7** `int(* cob_fileio_funcs::start)(cob_file *, const int, cob_field *)`

Definition at line 219 of file fileio.h.

**6.50.2.8** `int(* cob_fileio_funcs::write)(cob_file *, const int)`

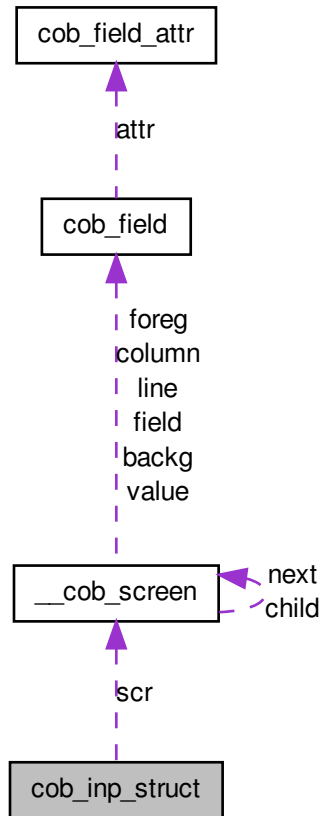
Definition at line 222 of file fileio.h.

The documentation for this struct was generated from the following file:

- [libcob/fileio.h](#)

## 6.51 cob\_inp\_struct Struct Reference

Collaboration diagram for cob\_inp\_struct:



### Public Attributes

- `cob_screen * scr`
- `size_t up_index`
- `size_t down_index`
- `int this_y`
- `int this_x`

### 6.51.1 Detailed Description

Definition at line 56 of file screenio.c.

### 6.51.2 Member Data Documentation

#### 6.51.2.1 `size_t cob_inp_struct::down_index`

Definition at line 59 of file screenio.c.

#### 6.51.2.2 `cob_screen* cob_inp_struct::scr`

Definition at line 57 of file screenio.c.

#### 6.51.2.3 `int cob_inp_struct::this_x`

Definition at line 61 of file screenio.c.

#### 6.51.2.4 `int cob_inp_struct::this_y`

Definition at line 60 of file screenio.c.

#### 6.51.2.5 `size_t cob_inp_struct::up_index`

Definition at line 58 of file screenio.c.

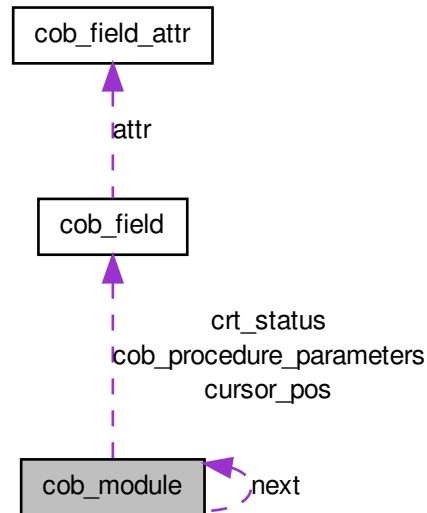
The documentation for this struct was generated from the following file:

- [libcob/screenio.c](#)

## 6.52 cob\_module Struct Reference

```
#include <common.h>
```

Collaboration diagram for `cob_module`:



### Public Attributes

- struct `cob_module` \* `next`
- const unsigned char \* `collating_sequence`
- `cob_field` \* `crt_status`
- `cob_field` \* `cursor_pos`
- `cob_field` \*\* `cob_procedure_parameters`
- const unsigned char `display_sign`
- const unsigned char `decimal_point`
- const unsigned char `currency_symbol`
- const unsigned char `numeric_separator`
- const unsigned char `flag_filename_mapping`
- const unsigned char `flag_binary_truncate`
- const unsigned char `flag_pretty_display`
- const unsigned char `spare8`

### 6.52.1 Detailed Description

Definition at line 201 of file `common.h`.

**6.52.2 Member Data Documentation****6.52.2.1 cob\_field\*\* cob\_module::cob\_procedure\_parameters**

Definition at line 206 of file common.h.

**6.52.2.2 const unsigned char\* cob\_module::collating\_sequence**

Definition at line 203 of file common.h.

**6.52.2.3 cob\_field\* cob\_module::crt\_status**

Definition at line 204 of file common.h.

**6.52.2.4 const unsigned char cob\_module::currency\_symbol**

Definition at line 209 of file common.h.

**6.52.2.5 cob\_field\* cob\_module::cursor\_pos**

Definition at line 205 of file common.h.

**6.52.2.6 const unsigned char cob\_module::decimal\_point**

Definition at line 208 of file common.h.

**6.52.2.7 const unsigned char cob\_module::display\_sign**

Definition at line 207 of file common.h.

**6.52.2.8 const unsigned char cob\_module::flag\_binary\_truncate**

Definition at line 212 of file common.h.

**6.52.2.9 const unsigned char cob\_module::flag\_filename\_mapping**

Definition at line 211 of file common.h.

**6.52.2.10 const unsigned char cob\_module::flag\_pretty\_display**

Definition at line 213 of file common.h.

#### 6.52.2.11 struct `cob_module*` `cob_module::next`

Definition at line 202 of file `common.h`.

#### 6.52.2.12 const unsigned char `cob_module::numeric_separator`

Definition at line 210 of file `common.h`.

#### 6.52.2.13 const unsigned char `cob_module::spare8`

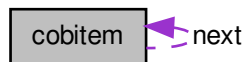
Definition at line 214 of file `common.h`.

The documentation for this struct was generated from the following file:

- [libcob/common.h](#)

### 6.53 cobitem Struct Reference

Collaboration diagram for `cobitem`:



#### Public Attributes

- struct `cobitem` \* `next`
- `size_t` `end_of_block`
- unsigned char `block_byte`
- unsigned char `unique` [`sizeof(size_t)`]
- unsigned char `item` [1]

#### 6.53.1 Detailed Description

Definition at line 158 of file `fileio.c`.



### 6.53.2 Member Data Documentation

#### 6.53.2.1 unsigned char cobitem::block\_byte

Definition at line 161 of file fileio.c.

#### 6.53.2.2 size\_t cobitem::end\_of\_block

Definition at line 160 of file fileio.c.

#### 6.53.2.3 unsigned char cobitem::item[1]

Definition at line 163 of file fileio.c.

#### 6.53.2.4 struct cobitem\* cobitem::next

Definition at line 159 of file fileio.c.

#### 6.53.2.5 unsigned char cobitem::unique[sizeof(size\_t)]

Definition at line 162 of file fileio.c.

The documentation for this struct was generated from the following file:

- [libcob/fileio.c](#)

## 6.54 cobjmp\_buf Struct Reference

```
#include <call.h>
```

### Public Attributes

- int [cbj\\_int](#) [4]
- void \* [cbj\\_ptr](#) [4]
- jmp\_buf [cbj\\_jmp\\_buf](#)
- void \* [cbj\\_ptr\\_rest](#) [2]

### 6.54.1 Detailed Description

Definition at line 27 of file call.h.

## 6.54.2 Member Data Documentation

### 6.54.2.1 `int cobjmp_buf::cbj_int[4]`

Definition at line 28 of file call.h.

### 6.54.2.2 `jmp_buf cobjmp_buf::cbj_jmp_buf`

Definition at line 30 of file call.h.

### 6.54.2.3 `void* cobjmp_buf::cbj_ptr[4]`

Definition at line 29 of file call.h.

### 6.54.2.4 `void* cobjmp_buf::cbj_ptr_rest[2]`

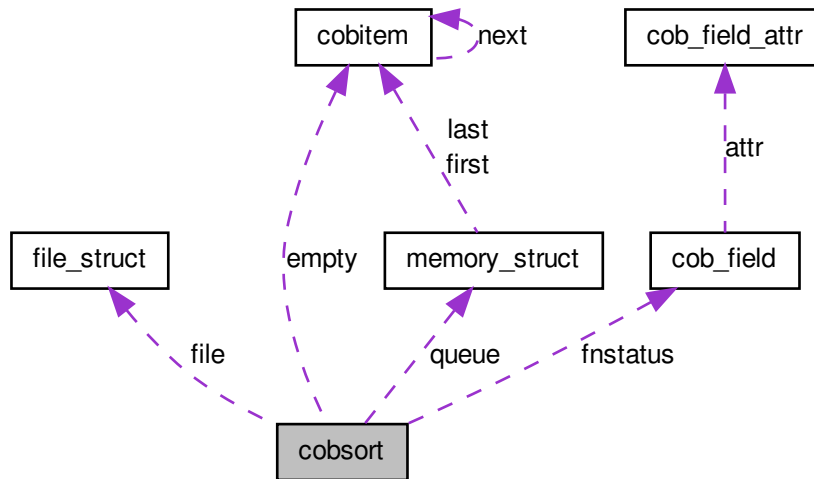
Definition at line 31 of file call.h.

The documentation for this struct was generated from the following file:

- [libcob/call.h](#)

## 6.55 cobsort Struct Reference

Collaboration diagram for cobsort:



### Public Attributes

- void \* `pointer`
- struct `cobitem` \* `empty`
- void \* `sort_return`
- `cob_field` \* `fnstatus`
- size\_t `unique`
- size\_t `retrieving`
- size\_t `files_used`
- size\_t `size`
- size\_t `r_size`
- size\_t `w_size`
- size\_t `memory`
- int `destination_file`
- int `retrieval_queue`
- struct `memory_struct` `queue` [4]
- struct `file_struct` `file` [4]

### 6.55.1 Detailed Description

Definition at line 177 of file fileio.c.

### 6.55.2 Member Data Documentation

#### 6.55.2.1 `int cobsort::destination_file`

Definition at line 189 of file fileio.c.

#### 6.55.2.2 `struct cobitem* cobsort::empty`

Definition at line 179 of file fileio.c.

#### 6.55.2.3 `struct file_struct cobsort::file[4]`

Definition at line 192 of file fileio.c.

#### 6.55.2.4 `size_t cobsort::files_used`

Definition at line 184 of file fileio.c.

#### 6.55.2.5 `cob_field* cobsort::fnstatus`

Definition at line 181 of file fileio.c.

#### 6.55.2.6 `size_t cobsort::memory`

Definition at line 188 of file fileio.c.

#### 6.55.2.7 `void* cobsort::pointer`

Definition at line 178 of file fileio.c.

#### 6.55.2.8 `struct memory_struct cobsort::queue[4]`

Definition at line 191 of file fileio.c.

#### 6.55.2.9 `size_t cobsort::r_size`

Definition at line 186 of file fileio.c.

**6.55.2.10 int cobsort::retrieval\_queue**

Definition at line 190 of file fileio.c.

**6.55.2.11 size\_t cobsort::retrieving**

Definition at line 183 of file fileio.c.

**6.55.2.12 size\_t cobsort::size**

Definition at line 185 of file fileio.c.

**6.55.2.13 void\* cobsort::sort\_return**

Definition at line 180 of file fileio.c.

**6.55.2.14 size\_t cobsort::unique**

Definition at line 182 of file fileio.c.

**6.55.2.15 size\_t cobsort::w\_size**

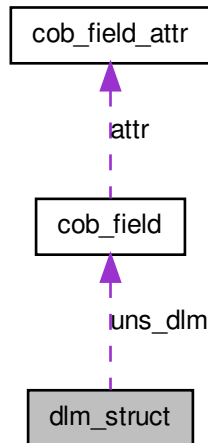
Definition at line 187 of file fileio.c.

The documentation for this struct was generated from the following file:

- [libcob/fileio.c](#)

## 6.56 dlm\_struct Struct Reference

Collaboration diagram for dlm\_struct:



### Public Attributes

- `cob_field * uns_dlm`
- `int uns_all`

### 6.56.1 Detailed Description

Definition at line 41 of file strings.c.

### 6.56.2 Member Data Documentation

#### 6.56.2.1 `int dlm_struct::uns_all`

Definition at line 43 of file strings.c.

#### 6.56.2.2 `cob_field* dlm_struct::uns_dlm`

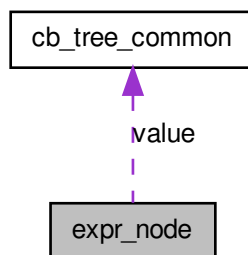
Definition at line 42 of file strings.c.

The documentation for this struct was generated from the following file:

- [libcob/strings.c](#)

## 6.57 `expr_node` Struct Reference

Collaboration diagram for `expr_node`:



### Public Attributes

- `int token`
- `cb_tree value`

#### 6.57.1 Detailed Description

Definition at line 48 of file `typeck.c`.

#### 6.57.2 Member Data Documentation

##### 6.57.2.1 `int expr_node::token`

Definition at line 56 of file `typeck.c`.

##### 6.57.2.2 `cb_tree expr_node::value`

Definition at line 58 of file `typeck.c`.

The documentation for this struct was generated from the following file:

- [cobc/typeck.c](#)

## 6.58 file\_struct Struct Reference

### Public Attributes

- FILE \* [fp](#)
- size\_t [count](#)

### 6.58.1 Detailed Description

Definition at line 172 of file fileio.c.

### 6.58.2 Member Data Documentation

#### 6.58.2.1 size\_t file\_struct::count

Definition at line 174 of file fileio.c.

#### 6.58.2.2 FILE\* file\_struct::fp

Definition at line 173 of file fileio.c.

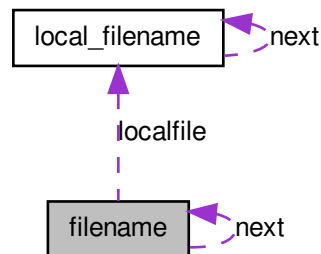
The documentation for this struct was generated from the following file:

- [libcob/fileio.c](#)

## 6.59 filename Struct Reference

```
#include <cobc.h>
```

Collaboration diagram for filename:





## Public Attributes

- struct [filename](#) \* [next](#)
- char \* [source](#)
- char \* [preprocess](#)
- char \* [translate](#)
- char \* [trstorage](#)
- char \* [object](#)
- char \* [demangle\\_source](#)
- struct [local\\_filename](#) \* [localfile](#)
- int [need\\_preprocess](#)
- int [need\\_translate](#)
- int [need\\_assemble](#)

### 6.59.1 Detailed Description

Definition at line 83 of file `cobc.h`.

### 6.59.2 Member Data Documentation

#### 6.59.2.1 char\* filename::demangle\_source

Definition at line 90 of file `cobc.h`.

#### 6.59.2.2 struct local\_filename\* filename::localfile

Definition at line 91 of file `cobc.h`.

#### 6.59.2.3 int filename::need\_assemble

Definition at line 94 of file `cobc.h`.

#### 6.59.2.4 int filename::need\_preprocess

Definition at line 92 of file `cobc.h`.

#### 6.59.2.5 int filename::need\_translate

Definition at line 93 of file `cobc.h`.

#### 6.59.2.6 struct filename\* filename::next

Definition at line 84 of file `cobc.h`.

**6.59.2.7 char\* filename::object**

Definition at line 89 of file cobc.h.

**6.59.2.8 char\* filename::preprocess**

Definition at line 86 of file cobc.h.

**6.59.2.9 char\* filename::source**

Definition at line 85 of file cobc.h.

**6.59.2.10 char\* filename::translate**

Definition at line 87 of file cobc.h.

**6.59.2.11 char\* filename::trstorage**

Definition at line 88 of file cobc.h.

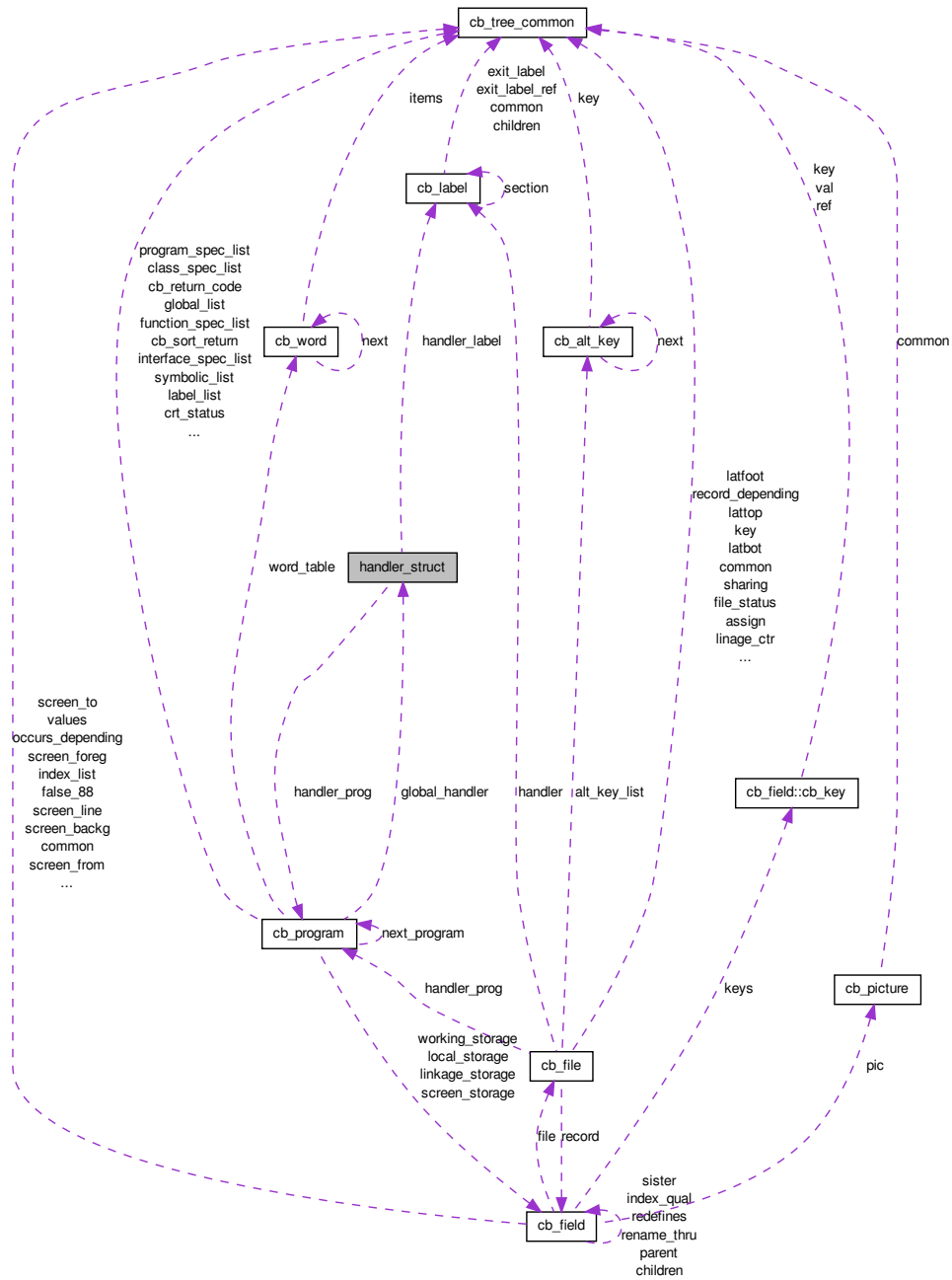
The documentation for this struct was generated from the following file:

- [cobc/cobc.h](#)

## 6.60 handler\_struct Struct Reference

```
#include <tree.h>
```

Collaboration diagram for handler\_struct:



## Public Attributes

- struct [cb\\_label](#) \* [handler\\_label](#)
- struct [cb\\_program](#) \* [handler\\_prog](#)

### 6.60.1 Detailed Description

Definition at line 583 of file tree.h.

### 6.60.2 Member Data Documentation

#### 6.60.2.1 struct [cb\\_label](#)\* [handler\\_struct::handler\\_label](#)

Definition at line 584 of file tree.h.

#### 6.60.2.2 struct [cb\\_program](#)\* [handler\\_struct::handler\\_prog](#)

Definition at line 585 of file tree.h.

The documentation for this struct was generated from the following file:

- [cobc/tree.h](#)

## 6.61 indexed\_file Struct Reference

### Public Attributes

- [size\\_t](#) [key\\_index](#)
- unsigned char \* [last\\_key](#)
- unsigned char \* [temp\\_key](#)
- DB \*\* [db](#)
- DBT [key](#)
- DBT [data](#)
- unsigned char \*\* [last\\_readkey](#)
- unsigned int \* [last\\_dupno](#)
- int \* [rewrite\\_sec\\_key](#)
- DBC \*\* [cursor](#)
- DB\_LOCK [bdb\\_file\\_lock](#)
- char \* [filename](#)
- DB\_LOCK [bdb\\_record\\_lock](#)
- int [write\\_cursor\\_open](#)
- unsigned int [bdb\\_lock\\_id](#)
- int [record\\_locked](#)
- int [filenamelen](#)

**6.61.1 Detailed Description**

Definition at line 307 of file fileio.c.

**6.61.2 Member Data Documentation****6.61.2.1 DB\_LOCK indexed\_file::bdb\_file\_lock**

Definition at line 319 of file fileio.c.

**6.61.2.2 unsigned int indexed\_file::bdb\_lock\_id**

Definition at line 323 of file fileio.c.

**6.61.2.3 DB\_LOCK indexed\_file::bdb\_record\_lock**

Definition at line 321 of file fileio.c.

**6.61.2.4 DBC\*\* indexed\_file::cursor**

Definition at line 318 of file fileio.c.

**6.61.2.5 DBT indexed\_file::data**

Definition at line 313 of file fileio.c.

**6.61.2.6 DB\*\* indexed\_file::db**

Definition at line 311 of file fileio.c.

**6.61.2.7 char\* indexed\_file::filename**

Definition at line 320 of file fileio.c.

**6.61.2.8 int indexed\_file::filenamelen**

Definition at line 325 of file fileio.c.

**6.61.2.9 DBT indexed\_file::key**

Definition at line 312 of file fileio.c.

**6.61.2.10** `size_t indexed_file::key_index`

Definition at line 308 of file fileio.c.

**6.61.2.11** `unsigned int* indexed_file::last_dupno`

Definition at line 315 of file fileio.c.

**6.61.2.12** `unsigned char* indexed_file::last_key`

Definition at line 309 of file fileio.c.

**6.61.2.13** `unsigned char** indexed_file::last_readkey`

Definition at line 314 of file fileio.c.

**6.61.2.14** `int indexed_file::record_locked`

Definition at line 324 of file fileio.c.

**6.61.2.15** `int* indexed_file::rewrite_sec_key`

Definition at line 316 of file fileio.c.

**6.61.2.16** `unsigned char* indexed_file::temp_key`

Definition at line 310 of file fileio.c.

**6.61.2.17** `int indexed_file::write_cursor_open`

Definition at line 322 of file fileio.c.

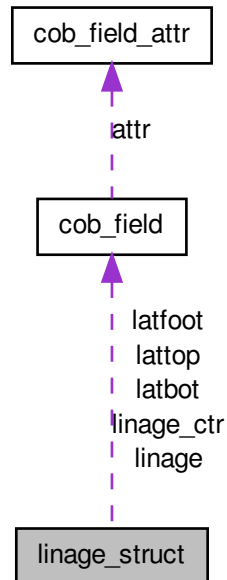
The documentation for this struct was generated from the following file:

- [libcob/fileio.c](#)

## 6.62 `linage_struct` Struct Reference

```
#include <fileio.h>
```

Collaboration diagram for lineage\_struct:



### Public Attributes

- `cob_field` \* `linage`
- `cob_field` \* `lineage_ctr`
- `cob_field` \* `latfoot`
- `cob_field` \* `lattop`
- `cob_field` \* `latbot`
- `int` `lin_lines`
- `int` `lin_foot`
- `int` `lin_top`
- `int` `lin_bot`

### 6.62.1 Detailed Description

Definition at line 159 of file `fileio.h`.

## 6.62.2 Member Data Documentation

### 6.62.2.1 `cob_field* lineage_struct::latbot`

Definition at line 164 of file fileio.h.

### 6.62.2.2 `cob_field* lineage_struct::latfoot`

Definition at line 162 of file fileio.h.

### 6.62.2.3 `cob_field* lineage_struct::lattop`

Definition at line 163 of file fileio.h.

### 6.62.2.4 `int lineage_struct::lin_bot`

Definition at line 168 of file fileio.h.

### 6.62.2.5 `int lineage_struct::lin_foot`

Definition at line 166 of file fileio.h.

### 6.62.2.6 `int lineage_struct::lin_lines`

Definition at line 165 of file fileio.h.

### 6.62.2.7 `int lineage_struct::lin_top`

Definition at line 167 of file fileio.h.

### 6.62.2.8 `cob_field* lineage_struct::linage`

Definition at line 160 of file fileio.h.

### 6.62.2.9 `cob_field* lineage_struct::linage_ctr`

Definition at line 161 of file fileio.h.

The documentation for this struct was generated from the following file:

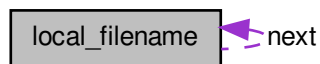
- [libcob/fileio.h](#)



## 6.63 local\_filename Struct Reference

```
#include <cobc.h>
```

Collaboration diagram for local\_filename:



### Public Attributes

- struct [local\\_filename](#) \* [next](#)
- char \* [local\\_name](#)
- FILE \* [local\\_fp](#)

### 6.63.1 Detailed Description

Definition at line 77 of file [cobc.h](#).

### 6.63.2 Member Data Documentation

#### 6.63.2.1 FILE\* [local\\_filename::local\\_fp](#)

Definition at line 80 of file [cobc.h](#).

#### 6.63.2.2 char\* [local\\_filename::local\\_name](#)

Definition at line 79 of file [cobc.h](#).

#### 6.63.2.3 struct [local\\_filename](#)\* [local\\_filename::next](#)

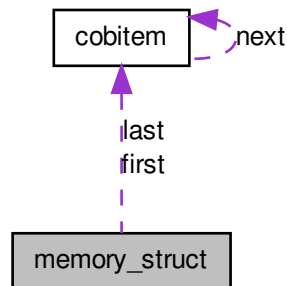
Definition at line 78 of file [cobc.h](#).

The documentation for this struct was generated from the following file:

- [cobc/cobc.h](#)

## 6.64 memory\_struct Struct Reference

Collaboration diagram for memory\_struct:



### Public Attributes

- struct `cobitem` \* `first`
- struct `cobitem` \* `last`
- size\_t `count`

### 6.64.1 Detailed Description

Definition at line 166 of file `fileio.c`.

### 6.64.2 Member Data Documentation

#### 6.64.2.1 size\_t memory\_struct::count

Definition at line 169 of file `fileio.c`.

#### 6.64.2.2 struct cobitem\* memory\_struct::first

Definition at line 167 of file `fileio.c`.

#### 6.64.2.3 struct cobitem\* memory\_struct::last

Definition at line 168 of file `fileio.c`.

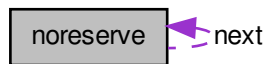
The documentation for this struct was generated from the following file:

- [libcob/fileio.c](#)

## 6.65 noreserve Struct Reference

```
#include <cobc.h>
```

Collaboration diagram for noreserve:



### Public Attributes

- struct [noreserve](#) \* [next](#)
- char \* [noresword](#)

### 6.65.1 Detailed Description

Definition at line 151 of file [cobc.h](#).

### 6.65.2 Member Data Documentation

#### 6.65.2.1 struct [noreserve](#)\* [noreserve::next](#)

Definition at line 152 of file [cobc.h](#).

#### 6.65.2.2 char\* [noreserve::noresword](#)

Definition at line 153 of file [cobc.h](#).

The documentation for this struct was generated from the following file:

- [cobc/cobc.h](#)

## 6.66 option Struct Reference

```
#include <getopt.h>
```

### Public Attributes

- char \* [name](#)
- int [has\\_arg](#)
- int \* [flag](#)
- int [val](#)

### 6.66.1 Detailed Description

Definition at line 94 of file getopt.h.

### 6.66.2 Member Data Documentation

#### 6.66.2.1 int\* option::flag

Definition at line 104 of file getopt.h.

#### 6.66.2.2 int option::has\_arg

Definition at line 103 of file getopt.h.

#### 6.66.2.3 char\* option::name

Definition at line 99 of file getopt.h.

#### 6.66.2.4 int option::val

Definition at line 105 of file getopt.h.

The documentation for this struct was generated from the following file:

- [lib/getopt.h](#)

## 6.67 reserved Struct Reference

### Public Attributes

- const char \* [name](#)
- const int [token](#)

### 6.67.1 Detailed Description

Definition at line 69 of file reserved.c.

### 6.67.2 Member Data Documentation

#### 6.67.2.1 `const char* reserved::name`

Definition at line 70 of file reserved.c.

#### 6.67.2.2 `const int reserved::token`

Definition at line 71 of file reserved.c.

The documentation for this struct was generated from the following file:

- [cobic/reserved.c](#)

## 6.68 sort\_list Struct Reference

Collaboration diagram for sort\_list:



### Public Attributes

- struct [sort\\_list](#) \* [next](#)

### 6.68.1 Detailed Description

Definition at line 121 of file codegen.c.

### 6.68.2 Member Data Documentation

### 6.68.2.1 struct sort\_list\* sort\_list::next

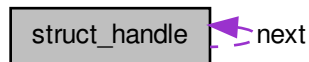
Definition at line 122 of file codegen.c.

The documentation for this struct was generated from the following file:

- [cobc/codegen.c](#)

## 6.69 struct\_handle Struct Reference

Collaboration diagram for struct\_handle:



### Public Attributes

- struct [struct\\_handle](#) \* [next](#)
- [lt\\_dlhandle](#) [preload\\_handle](#)

### 6.69.1 Detailed Description

Definition at line 119 of file call.c.

### 6.69.2 Member Data Documentation

#### 6.69.2.1 struct struct\_handle\* struct\_handle::next

Definition at line 120 of file call.c.

#### 6.69.2.2 lt\_dlhandle struct\_handle::preload\_handle

Definition at line 121 of file call.c.

The documentation for this struct was generated from the following file:

- [libcob/call.c](#)

## 6.70 `system_table` Struct Reference

### Public Attributes

- `const char * syst_name`
- `const char * syst_call`
- `const int syst_params`
- `void * syst_call`

### 6.70.1 Detailed Description

Definition at line 125 of file `codegen.c`.

### 6.70.2 Member Data Documentation

#### 6.70.2.1 `const char* system_table::syst_call`

Definition at line 127 of file `codegen.c`.

#### 6.70.2.2 `void* system_table::syst_call`

Definition at line 148 of file `call.c`.

#### 6.70.2.3 `const char * system_table::syst_name`

Definition at line 126 of file `codegen.c`.

#### 6.70.2.4 `const int system_table::syst_params`

Definition at line 45 of file `typeck.c`.

The documentation for this struct was generated from the following files:

- `cobc/codegen.c`
- `cobc/typeck.c`
- `libcob/call.c`

## 6.71 `yy_buffer_state` Struct Reference

### Public Attributes

- `FILE * yy_input_file`
- `char * yy_ch_buf`

- [char \\* yy\\_buf\\_pos](#)
- [yy\\_size\\_t yy\\_buf\\_size](#)
- [int yy\\_n\\_chars](#)
- [int yy\\_is\\_our\\_buffer](#)
- [int yy\\_is\\_interactive](#)
- [int yy\\_at\\_bol](#)
- [int yy\\_fill\\_buffer](#)
- [int yy\\_buffer\\_status](#)

### 6.71.1 Detailed Description

Definition at line 162 of file pplex.c.

### 6.71.2 Member Data Documentation

#### 6.71.2.1 `int yy_buffer_state::yy_at_bol`

Definition at line 196 of file pplex.c.

#### 6.71.2.2 `char * yy_buffer_state::yy_buf_pos`

Definition at line 167 of file pplex.c.

#### 6.71.2.3 `yy_size_t yy_buffer_state::yy_buf_size`

Definition at line 172 of file pplex.c.

#### 6.71.2.4 `int yy_buffer_state::yy_buffer_status`

Definition at line 203 of file pplex.c.

#### 6.71.2.5 `char * yy_buffer_state::yy_ch_buf`

Definition at line 166 of file pplex.c.

#### 6.71.2.6 `int yy_buffer_state::yy_fill_buffer`

Definition at line 201 of file pplex.c.

#### 6.71.2.7 `FILE * yy_buffer_state::yy_input_file`

Definition at line 164 of file pplex.c.



### 6.71.2.8 int yy\_buffer\_state::yy\_is\_interactive

Definition at line 190 of file pplex.c.

### 6.71.2.9 int yy\_buffer\_state::yy\_is\_our\_buffer

Definition at line 183 of file pplex.c.

### 6.71.2.10 int yy\_buffer\_state::yy\_n\_chars

Definition at line 177 of file pplex.c.

The documentation for this struct was generated from the following files:

- [cobic/pplex.c](#)
- [cobic/scanner.c](#)

## 6.72 yyalloC Union Reference

### Public Attributes

- short [yyss](#)
- [YYSTYPE yyvs](#)

### 6.72.1 Detailed Description

Definition at line 1283 of file parser.c.

### 6.72.2 Member Data Documentation

#### 6.72.2.1 short yyalloC::yyss

Definition at line 1285 of file parser.c.

#### 6.72.2.2 YYSTYPE yyalloC::yyvs

Definition at line 1286 of file parser.c.

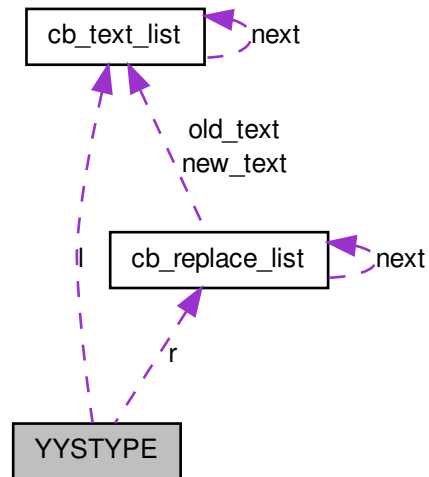
The documentation for this union was generated from the following files:

- [cobic/parser.c](#)
- [cobic/ppparse.c](#)

## 6.73 YYSTYPE Union Reference

```
#include <ppparse.h>
```

Collaboration diagram for YYSTYPE:



### Public Attributes

- char \* `s`
- struct `cb_text_list` \* `l`
- struct `cb_replace_list` \* `r`

#### 6.73.1 Detailed Description

Definition at line 134 of file `ppparse.c`.

#### 6.73.2 Member Data Documentation

##### 6.73.2.1 struct `cb_text_list` \* YYSTYPE::l

Definition at line 136 of file `ppparse.c`.

**6.73.2.2 struct cb\_replace\_list \* YYSTYPE::r**

Definition at line 137 of file ppparse.c.

**6.73.2.3 char \* YYSTYPE::s**

Definition at line 135 of file ppparse.c.

The documentation for this union was generated from the following files:

- [cobic/ppparse.c](#)
- [cobic/ppparse.h](#)



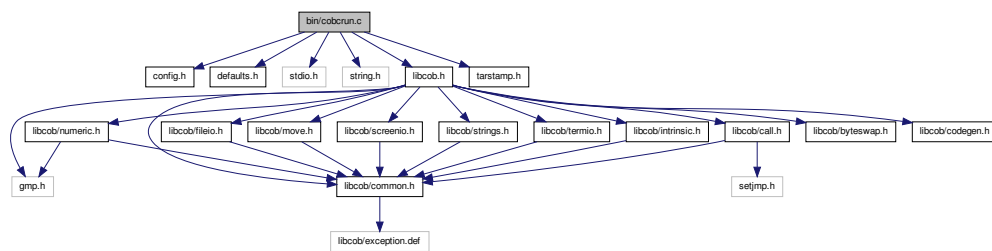
# Chapter 7

## File Documentation

### 7.1 bin/cobcrun.c File Reference

```
#include "config.h"  
#include "defaults.h"  
#include <stdio.h>  
#include <string.h>  
#include <libcob.h>  
#include <tarstamp.h>
```

Include dependency graph for cobcrun.c:



### Functions

- int [main](#) (int argc, char \*\*argv)

#### 7.1.1 Function Documentation

### 7.1.1.1 int main ( int argc, char \*\* argv )

Definition at line 63 of file cobcrun.c.

```
{
    union {
        int      (*func)();
        void     *func_void;
    } unifunc;

    if (argc <= 1) {
        print_usage ();
        return 1;
    }
    /* Quick check without getopt */
    if (!strncmp (argv[1], "--version", 10) ||
        !strncmp (argv[1], "-v", 4)) {
        print_version ();
        return 0;
    }
    if (!strncmp (argv[1], "--help", 10) ||
        !strncmp (argv[1], "-h", 4)) {
        print_usage ();
        return 0;
    }
    if (strlen (argv[1]) > 31) {
        fprintf (stderr, "Invalid PROGRAM name\n");
        return 1;
    }
    cob_init (argc - 1, &argv[1]);
    unifunc.func_void = cob_resolve (argv[1]);
    if (unifunc.func_void == NULL) {
        cob_call_error ();
    }
    cob_stop_run ( unifunc.func() );
}
```

## 7.2 cobc/cobc.c File Reference

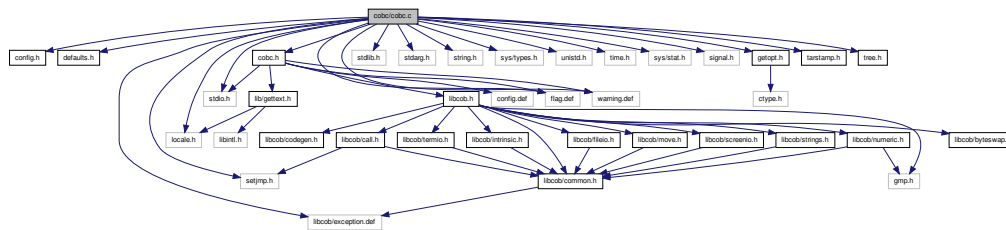
```
#include "config.h"
#include "defaults.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <setjmp.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>
#include <sys/stat.h>
```

```

#include <signal.h>
#include <getopt.h>
#include <locale.h>
#include <tarstamp.h>
#include "cobc.h"
#include "tree.h"
#include <libcob/exception.def>
#include "flag.def"
#include "warning.def"

```

Include dependency graph for cobc.c:



## Defines

- #define `COB_EXCEPTION`(code, tag, name, critical) {name, 0x##code, 0},
- #define `CB_FLAG`(var, name, doc) int var = 0;
- #define `CB_WARNDEF`(var, name, wall, doc) int var = 0;
- #define `PATHSEPS` ":"
- #define `COB_NUM_CSYS` sizeof(cob\_csyns) / sizeof(char \*)
- #define `CB_FLAG`(var, name, doc)
- #define `CB_WARNDEF`(var, name, wall, doc)
- #define `CB_WARNDEF`(var, name, wall, doc)
- #define `CB_FLAG`(var, name, doc)
- #define `CB_WARNDEF`(var, name, wall, doc) var = 0;
- #define `CB_WARNDEF`(var, name, wall, doc) if (wall) var = 1;
- #define `CB_WARNDEF`(var, name, wall, doc) var = 1;

## Typedefs

- typedef void(\* `cob_sighandler_t`)(int)

## Enumerations

- enum `cb_compile_level` {  
    `CB_LEVEL_PREPROCESS` = 1, `CB_LEVEL_TRANSLATE`, `CB_LEVEL_COMPILE`,  
    `CB_LEVEL_ASSEMBLE`,  
    `CB_LEVEL_MODULE`, `CB_LEVEL_LIBRARY`, `CB_LEVEL_EXECUTABLE` }

## Functions

- void `cobc_abort` (const char \*filename, const int linenum)
- void `cobc_tree_cast_error` (cb\_tree x, const char \*file, const int linenum, const int tagnum)
- void \* `cobc_malloc` (const size\_t size)
- void \* `cobc_realloc` (void \*prevptr, const size\_t size)
- struct `cb_text_list` \* `cb_text_list_add` (struct `cb_text_list` \*list, const char \*text)
- size\_t `cobc_check_valid_name` (char \*name)
- int `main` (int argc, char \*argv[])

## Variables

- int `cb_source_format` = `CB_FORMAT_FIXED`
- int `cb_display_sign` = `COB_DISPLAY_SIGN_ASCII`
- struct `cb_exception` `cb_exception_table` []
- int `cb_id` = 1
- int `cb_attr_id` = 1
- int `cb_literal_id` = 1
- int `cb_field_id` = 1
- int `cb_storage_id` = 1
- int `cb_flag_main` = 0
- int `errorcount` = 0
- int `warningcount` = 0
- int `alt_ebcdic` = 0
- int `optimize_flag` = 0
- char \* `cb_source_file` = NULL
- char \* `cb_oc_build_stamp` = NULL
- char \* `source_name`
- char \* `demangle_name`
- int `cb_source_line` = 0
- FILE \* `cb_storage_file`
- char \* `cb_storage_file_name`
- FILE \* `cb_listing_file` = NULL
- FILE \* `cb_depend_file` = NULL
- char \* `cb_depend_target` = NULL
- struct `cb_text_list` \* `cb_depend_list` = NULL
- struct `cb_text_list` \* `cb_include_list` = NULL



- struct `cb_text_list` \* `cb_extension_list` = NULL
- int `cb_saveargc`
- char \*\* `cb_saveargv`
- const char \* `cob_config_dir`

## 7.2.1 Define Documentation

7.2.1.1 `#define CB_FLAG( var, name, doc ) int var = 0;`

Definition at line 97 of file `cobc.c`.

7.2.1.2 `#define CB_FLAG( var, name, doc )`

**Value:**

```
if (strcmp (name, "static-call")) \
    printf (" -f%-19s %s\n", name, gettext (doc));
```

Definition at line 97 of file `cobc.c`.

7.2.1.3 `#define CB_FLAG( var, name, doc )`

**Value:**

```
{"f" name, no_argument, &var, 1}, \
 {"fno-" name, no_argument, &var, 0},
```

Definition at line 97 of file `cobc.c`.

7.2.1.4 `#define CB_WARNDEF( var, name, wall, doc ) var = 1;`

Definition at line 102 of file `cobc.c`.

7.2.1.5 `#define CB_WARNDEF( var, name, wall, doc )`

**Value:**

```
printf (" -W%-19s %s", name, gettext (doc)); \
if (!wall) { \
    puts (_(" (NOT set with -Wall)")); \
} else { \
    printf ("\n"); \
}
```

Definition at line 102 of file `cobc.c`.

7.2.1.6 `#define CB_WARNDEF( var, name, wall, doc ) var = 0;`

Definition at line 102 of file cobc.c.

7.2.1.7 `#define CB_WARNDEF( var, name, wall, doc )`

**Value:**

```
{ "W" name, no_argument, &var, 1 },          \
  { "Wno-" name, no_argument, &var, 0 },
```

Definition at line 102 of file cobc.c.

7.2.1.8 `#define CB_WARNDEF( var, name, wall, doc ) int var = 0;`

Definition at line 102 of file cobc.c.

7.2.1.9 `#define CB_WARNDEF( var, name, wall, doc ) if (wall) var = 1;`

Definition at line 102 of file cobc.c.

7.2.1.10 `#define COB_EXCEPTION( code, tag, name, critical ) { name, 0x##code, 0 },`

Definition at line 88 of file cobc.c.

7.2.1.11 `#define COB_NUM_CSYSNS sizeof(cob_csyns) / sizeof(char *)`

Definition at line 201 of file cobc.c.

7.2.1.12 `#define PATHSEPS ":"`

Definition at line 109 of file cobc.c.

## 7.2.2 Typedef Documentation

7.2.2.1 `typedef void(* cob_sighandler_t)(int)`

Definition at line 249 of file cobc.c.

## 7.2.3 Enumeration Type Documentation

7.2.3.1 enum `cb_compile_level`

Enumerator:

```
CB_LEVEL_PREPROCESS  
CB_LEVEL_TRANSLATE  
CB_LEVEL_COMPILE  
CB_LEVEL_ASSEMBLE  
CB_LEVEL_MODULE  
CB_LEVEL_LIBRARY  
CB_LEVEL_EXECUTABLE
```

Definition at line 65 of file `cobc.c`.

```
    {  
        CB_LEVEL_PREPROCESS = 1,  
        CB_LEVEL_TRANSLATE,  
        CB_LEVEL_COMPILE,  
        CB_LEVEL_ASSEMBLE,  
        CB_LEVEL_MODULE,  
        CB_LEVEL_LIBRARY,  
        CB_LEVEL_EXECUTABLE  
    };
```

## 7.2.4 Function Documentation

7.2.4.1 `struct cb_text_list* cb_text_list.add ( struct cb_text_list * list, const char * text )`  
[read]

Definition at line 355 of file `cobc.c`.

```
{  
    struct cb_text_list *p;  
    struct cb_text_list *l;  
  
    p = cobc_malloc (sizeof (struct cb_text_list));  
    p->text = strdup (text);  
    p->next = NULL;  
    if (!list) {  
        return p;  
    } else {  
        for (l = list; l->next; l = l->next) { ; }  
        l->next = p;  
        return list;  
    }  
}
```

7.2.4.2 `void cobc.abort ( const char * filename, const int linenum )`

Definition at line 310 of file `cobc.c`.

```

{
    fprintf (stderr, "%s:%d: Internal compiler error\n", filename, linenum);
    (void)longjmp (cob_jmpbuf, 1);
}

```

#### 7.2.4.3 `size_t cobc_check_valid_name ( char * name )`

Definition at line 373 of file `cobc.c`.

```

{
    size_t n;

    for (n = 0; n < COB_NUM_CSYSNS; ++n) {
        if (!strcmp (name, cob_csyns[n])) {
            return 1;
        }
    }
    return 0;
}

```

#### 7.2.4.4 `void* cobc_malloc ( const size_t size )`

Definition at line 327 of file `cobc.c`.

```

{
    void *mptr;

    mptr = calloc (1, size);
    if (!mptr) {
        fprintf (stderr, "Cannot allocate %d bytes of memory - Aborting\n",
            (int)size);
        fflush (stderr);
        (void)longjmp (cob_jmpbuf, 1);
    }
    return mptr;
}

```

#### 7.2.4.5 `void* cobc_realloc ( void * prevptr, const size_t size )`

Definition at line 341 of file `cobc.c`.

```

{
    void *mptr;

    mptr = realloc (prevptr, size);
    if (!mptr) {
        fprintf (stderr, "Cannot reallocate %d bytes of memory - Aborting\n",
            (int)size);
        fflush (stderr);
        (void)longjmp (cob_jmpbuf, 1);
    }
    return mptr;
}

```

### 7.2.4.6 void cobc\_tree\_cast\_error ( cb\_tree x, const char \* *filen*, const int *linenum*, const int *tagnum* )

Definition at line 317 of file cobc.c.

```
{
    fprintf (stderr, "%s:%d: Invalid type cast from '%s'\n",
             filen, linenum, x ? cb_name (x) : "null");
    fprintf (stderr, "Tag 1 %d Tag 2 %d\n", x ? CB_TREE_TAG(x) : 0,
             tagnum);
    (void)longjmp (cob_jmpbuf, 1);
}
```

### 7.2.4.7 int main ( int *argc*, char \* *argv*[] )

Definition at line 1686 of file cobc.c.

```
{
    struct filename      *fn;
    char                 *p;
    int                  status = 1;
    int                  year;
    int                  day;
    char                 month[32];
    char                 buff[COB_SMALL_BUFF];

#ifdef HAVE_SIGNAL_H
    if ((intsig = signal (SIGINT, cobc_sig_handler)) == SIG_IGN) {
        (void)signal (SIGINT, SIG_IGN);
    }
#endif
#ifdef SIGHUP
    if ((hupsig = signal (SIGHUP, cobc_sig_handler)) == SIG_IGN) {
        (void)signal (SIGHUP, SIG_IGN);
    }
#endif
#ifdef SIGQUIT
    if ((qutsig = signal (SIGQUIT, cobc_sig_handler)) == SIG_IGN) {
        (void)signal (SIGQUIT, SIG_IGN);
    }
#endif
#endif
#endif

    cb_saveargc = argc;
    cb_saveargv = argv;

#ifdef HAVE_SETLOCALE
    setlocale (LC_ALL, "");
#endif

#ifdef ENABLE_NLS
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);
#endif

#ifdef _WIN32
    cob_process_id = getpid ();
#endif
}
```

```

/* Initialize global variables */
memset (buff, 0, sizeof(buff));
memset (month, 0, sizeof(month));
day = 0;
year = 0;
sscanf (__DATE__, "%s %d %d", month, &day, &year);
if (day && year) {
    sprintf (buff, "%s %2.2d %4.4d %s", month, day, year, __TIME__);
} else {
    sprintf (buff, "%s %s", __DATE__, __TIME__);
}
cb_oc_build_stamp = cobc_malloc (strlen (buff) + 1);
strcpy (cb_oc_build_stamp, buff);

output_name = NULL;

if ((p = getenv ("TMPDIR")) != NULL) {
    cob_tmpdir = p;
} else if ((p = getenv ("TMP")) != NULL) {
    cob_tmpdir = p;
    sprintf (buff, "TMPDIR=%s", p);
    p = strdup (buff);
    putenv (p);
} else {
    cob_tmpdir = "/tmp";
    putenv ((char *) "TMPDIR=/tmp");
}
cob_cc = getenv ("COB_CC");
if (cob_cc == NULL) {
    cob_cc = COB_CC;
}
cobc_init_var (cob_cflags, "COB_CFLAGS", COB_CFLAGS);
cobc_init_var (cob_libs, "COB_LIBS", COB_LIBS);
cob_ldflags = getenv ("COB_LDFLAGS");
if (cob_ldflags == NULL) {
    cob_ldflags = COB_LDFLAGS;
}
cob_config_dir = getenv ("COB_CONFIG_DIR");
if (cob_config_dir == NULL) {
    cob_config_dir = COB_CONFIG_DIR;
}
cob_copy_dir = getenv ("COB_COPY_DIR");
if (cob_copy_dir == NULL) {
    cob_copy_dir = COB_COPY_DIR;
}
memset (cob_define_flags, 0, sizeof (cob_define_flags));

p = getenv ("COB_LDADD");
if (p) {
    strcat (cob_libs, " ");
    strcat (cob_libs, p);
}
p = getenv ("COB_EBCDIC");
if (p && (*p == 'F' || *p == 'f')) {
    alt_ebcdic = 1;
}

/* Process command line arguments */
iargs = process_command_line (argc, argv);

/* Check the filename */
if (iargs == argc) {

```

```

        fprintf (stderr, "cobc: No input files\n");
        exit (1);
    }

    /* processes COBCPY environment variable */
    process_env_copy_path ();

    cb_include_list = cb_text_list_add (cb_include_list, cob_copy_dir);

    file_list = NULL;

    if (setjmp (cob_jmpbuf) != 0) {
        fprintf (stderr, "Aborting compile of %s at line %d\n",
                cb_source_file, cb_source_line);
        fflush (stderr);
        if (yyout) {
            fflush (yyout);
        }
        if (cb_storage_file) {
            fflush (cb_storage_file);
        }
        status = 1;
        cobc_clean_up (status);
        return status;
    }

    /* Defaults are set here */
    if (!cb_flag_syntax_only) {
        if (!wants_nonfinal) {
            if (cb_flag_main) {
                cb_compile_level = CB_LEVEL_EXECUTABLE;
            }
            if (cb_flag_module) {
                cb_compile_level = CB_LEVEL_MODULE;
            }
            if (cb_flag_library) {
                cb_compile_level = CB_LEVEL_LIBRARY;
            }
        }
        if (cb_compile_level == 0 && !wants_nonfinal) {
            cb_compile_level = CB_LEVEL_MODULE;
            cb_flag_module = 1;
        }
        if (wants_nonfinal && cb_compile_level != CB_LEVEL_PREPROCESS &&
            !cb_flag_main && !cb_flag_module && !cb_flag_library) {
            cb_flag_module = 1;
        }
    } else {
        cb_compile_level = CB_LEVEL_TRANSLATE;
    }

    if (output_name && cb_compile_level < CB_LEVEL_LIBRARY &&
        (argc - iargs) > 1) {
        fprintf (stderr, "cobc: -o option invalid in this combination\n");
;
        exit (1);
    }
    if (cb_flag_sign_ascii && cb_flag_sign_ebcdic) {
        fprintf (stderr, "Only one of -fsign-ascii or -fsign-ebcdic may b
e specified\n");
        exit (1);
    }
}

```

```

if (cb_flag_sign_ascii) {
    cb_display_sign = COB_DISPLAY_SIGN_ASCII;
}
if (cb_flag_sign_ebcdic) {
    cb_display_sign = COB_DISPLAY_SIGN_EBCDIC;
}
if (cb_flag_notrunc) {
    cb_binary_truncate = 0;
    cb_pretty_display = 0;
}

while (iargs < argc) {
    fn = process_filename (argv[iargs++]);
    if (!fn) {
        status = 1;
        cobc_clean_up (status);
        return status;
    }
    /* Preprocess */
    if (cb_compile_level >= CB_LEVEL_PREPROCESS && fn->
need_preprocess) {
        if (preprocess (fn) != 0) {
            cobc_clean_up (status);
            return status;
        }
    }
}
for (fn = file_list; fn; fn = fn->next) {
    cb_id = 1;
    cb_attr_id = 1;
    cb_literal_id = 1;
    cb_field_id = 1;
    cb_storage_id = 1;
    iparams++;
    demangle_name = fn->demangle_source;
    if (iparams > 1 && cb_compile_level == CB_LEVEL_EXECUTABLE &&
!cb_flag_syntax_only) {
        local_level = cb_compile_level;
        cb_flag_main = 0;
        cb_compile_level = CB_LEVEL_ASSEMBLE;
    }
    /* Translate */
    if (cb_compile_level >= CB_LEVEL_TRANSLATE && fn->need_translate)
{
        if (process_translate (fn) != 0) {
            cobc_clean_up (status);
            return status;
        }
    }
    if (cb_flag_syntax_only) {
        continue;
    }

    /* Compile */
    if (cb_compile_level == CB_LEVEL_COMPILE) {
        if (process_compile (fn) != 0) {
            cobc_clean_up (status);
            return status;
        }
    }

    /* Build module */

```



```
        if (cb_compile_level == CB_LEVEL_MODULE && fn->need_assemble) {
            if (process_module_direct (fn) != 0) {
                cobc_clean_up (status);
                return status;
            }
        } else {
            /* Assemble */
            if (cb_compile_level >= CB_LEVEL_ASSEMBLE && fn->
need_assemble) {
                if (process_assemble (fn) != 0) {
                    cobc_clean_up (status);
                    return status;
                }
            }

            /* Build module */
            if (cb_compile_level == CB_LEVEL_MODULE) {
                if (process_module (fn) != 0) {
                    cobc_clean_up (status);
                    return status;
                }
            }
        }
    }

    if (!cb_flag_syntax_only) {
        /* Link */
        if (local_level == CB_LEVEL_EXECUTABLE) {
            cb_compile_level = CB_LEVEL_EXECUTABLE;
            cb_flag_main = 1;
        }
        if (cb_compile_level == CB_LEVEL_LIBRARY) {
            if (process_library (file_list) != 0) {
                cobc_clean_up (status);
                return status;
            }
        } else if (cb_compile_level == CB_LEVEL_EXECUTABLE) {
            if (process_link (file_list) != 0) {
                cobc_clean_up (status);
                return status;
            }
        }
    }

    /* We have successfully completed */
    status = 0;
    cobc_clean_up (status);

    return status;
}
```

## 7.2.5 Variable Documentation

### 7.2.5.1 int alt\_ebcdic = 0

Definition at line 121 of file cobc.c.

**7.2.5.2 int cb\_attr\_id = 1**

Definition at line 113 of file cobc.c.

**7.2.5.3 FILE\* cb\_depend\_file = NULL**

Definition at line 134 of file cobc.c.

**7.2.5.4 struct cb\_text\_list\* cb\_depend\_list = NULL**

Definition at line 136 of file cobc.c.

**7.2.5.5 char\* cb\_depend\_target = NULL**

Definition at line 135 of file cobc.c.

**7.2.5.6 int cb\_display\_sign = COB\_DISPLAY\_SIGN\_ASCII**

Definition at line 84 of file cobc.c.

**7.2.5.7 struct cb\_exception cb\_exception\_table[]**

Definition at line 89 of file cobc.c.

**7.2.5.8 struct cb\_text\_list\* cb\_extension\_list = NULL**

Definition at line 138 of file cobc.c.

**7.2.5.9 int cb\_field\_id = 1**

Definition at line 115 of file cobc.c.

**7.2.5.10 int cb\_flag\_main = 0**

Definition at line 117 of file cobc.c.

**7.2.5.11 int cb\_id = 1**

Definition at line 112 of file cobc.c.

**7.2.5.12** `struct cb_text_list* cb_include_list = NULL`

Definition at line 137 of file cobc.c.

**7.2.5.13** `FILE* cb_listing_file = NULL`

Definition at line 133 of file cobc.c.

**7.2.5.14** `int cb_literal_id = 1`

Definition at line 114 of file cobc.c.

**7.2.5.15** `char* cb_oc_build_stamp = NULL`

Definition at line 125 of file cobc.c.

**7.2.5.16** `int cb_saveargc`

Definition at line 140 of file cobc.c.

**7.2.5.17** `char** cb_saveargv`

Definition at line 141 of file cobc.c.

**7.2.5.18** `char* cb_source_file = NULL`

Definition at line 124 of file cobc.c.

**7.2.5.19** `int cb_source_format = CB_FORMAT_FIXED`

Definition at line 79 of file cobc.c.

**7.2.5.20** `int cb_source_line = 0`

Definition at line 128 of file cobc.c.

**7.2.5.21** `FILE* cb_storage_file`

Definition at line 130 of file cobc.c.

**7.2.5.22 char\* cb\_storage\_file\_name**

Definition at line 131 of file cobc.c.

**7.2.5.23 int cb\_storage\_id = 1**

Definition at line 116 of file cobc.c.

**7.2.5.24 const char\* cob\_config\_dir**

Definition at line 143 of file cobc.c.

**7.2.5.25 char\* demangle\_name**

Definition at line 127 of file cobc.c.

**7.2.5.26 int errorcount = 0**

Definition at line 119 of file cobc.c.

**7.2.5.27 int optimize\_flag = 0**

Definition at line 122 of file cobc.c.

**7.2.5.28 char\* source\_name**

Definition at line 126 of file cobc.c.

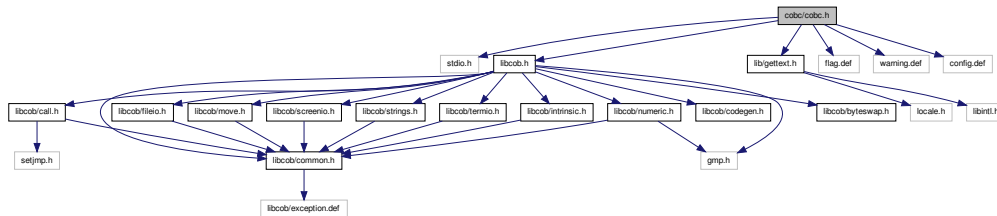
**7.2.5.29 int warningcount = 0**

Definition at line 120 of file cobc.c.

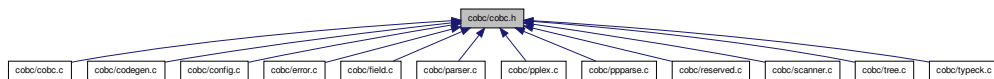
## 7.3 cobc/cobc.h File Reference

```
#include <stdio.h>
#include <libcob.h>
#include "lib/gettext.h"
#include "flag.def"
#include "warning.def"
#include "config.def"
```

Include dependency graph for cobc.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [cb\\_exception](#)
- struct [cb\\_text\\_list](#)
- struct [cb\\_replace\\_list](#)
- struct [local\\_filename](#)
- struct [filename](#)
- struct [noreserve](#)

## Defines

- #define [COB\\_NON\\_ALIGNED](#)
- #define [ABORT\(\)](#) `cobc_abort(__FILE__, __LINE__)`
- #define [CB\\_FORMAT\\_FIXED](#) 0
- #define [CB\\_FORMAT\\_FREE](#) 1
- #define [CB\\_EXCEPTION\\_NAME\(id\)](#) `cb_exception_table[id].name`
- #define [CB\\_EXCEPTION\\_CODE\(id\)](#) `cb_exception_table[id].code`
- #define [CB\\_EXCEPTION\\_ENABLE\(id\)](#) `cb_exception_table[id].enable`
- #define [CB\\_FLAG\(var, name, doc\)](#) `extern int var;`
- #define [CB\\_WARNDEF\(var, name, wall, doc\)](#) `extern int var;`
- #define [CB\\_CONFIG\\_ANY\(type, var, name\)](#) `extern type var;`
- #define [CB\\_CONFIG\\_INT\(var, name\)](#) `extern int var;`
- #define [CB\\_CONFIG\\_STRING\(var, name\)](#) `extern const char *var;`
- #define [CB\\_CONFIG\\_BOOLEAN\(var, name\)](#) `extern int var;`
- #define [CB\\_CONFIG\\_SUPPORT\(var, name\)](#) `extern enum cb_support var;`

## Enumerations

- enum `cb_assign_clause` { `CB_ASSIGN_COBOL2002`, `CB_ASSIGN_MF`, `CB_ASSIGN_IBM` }
- enum `cb_binary_byteorder` { `CB_BYTEORDER_NATIVE`, `CB_BYTEORDER_BIG_ENDIAN` }
- enum `cb_binary_size` { `CB_BINARY_SIZE_2_4_8`, `CB_BINARY_SIZE_1_2_4_8`, `CB_BINARY_SIZE_1_8` }
- enum `cb_operation_type` { `CB_OPERATION_READ`, `CB_OPERATION_WRITE`, `CB_OPERATION_ASSIGN` }
- enum `cb_support` {  
`CB_OK`, `CB_WARNING`, `CB_ARCHAIC`, `CB_OBSOLETE`,  
`CB_SKIP`, `CB_IGNORE`, `CB_ERROR`, `CB_UNCONFORMABLE` }

## Functions

- struct `cb_text_list` \* `cb_text_list_add` (struct `cb_text_list` \*list, const char \*name)
- void \* `cobc_malloc` (const size\_t size)
- void \* `cobc_realloc` (void \*prevptr, const size\_t size)
- void `cobc_abort` (const char \*filename, const int linenum)
- size\_t `cobc_check_valid_name` (char \*name)
- int `cb_load_std` (const char \*name)
- int `cb_load_conf` (const char \*fname, const int check\_nodf, const int prefix\_dir)
- int `pplex` (void)
- int `ppparse` (void)
- int `ppopen` (const char \*name, struct `cb_replace_list` \*replace\_list)
- int `ppcopy` (const char \*name, const char \*lib, struct `cb_replace_list` \*replace\_list)
- void `pp_set_replace_list` (struct `cb_replace_list` \*replace\_list)
- int `yylex` (void)
- int `yyparse` (void)
- void `cb_warning` (const char \*fmt,...)
- void `cb_error` (const char \*fmt,...)
- int `cb_verify` (const enum `cb_support` tag, const char \*feature)

## Variables

- int `cb_source_format`
- int `cb_display_sign`
- struct `cb_exception` `cb_exception_table` []
- int `cb_id`
- int `cb_attr_id`
- int `cb_literal_id`
- int `cb_field_id`
- int `cb_storage_id`

- int `cb_flag_main`
- int `errorcount`
- int `warningcount`
- int `alt_ebcdic`
- int `optimize_flag`
- int `has_external`
- char \* `cb_oc_build_stamp`
- char \* `cb_source_file`
- int `cb_source_line`
- const char \* `cob_config_dir`
- char \* `source_name`
- char \* `demangle_name`
- FILE \* `cb_storage_file`
- char \* `cb_storage_file_name`
- char \*\* `cb_saveargv`
- int `cb_saveargc`
- FILE \* `cb_listing_file`
- FILE \* `cb_depend_file`
- char \* `cb_depend_target`
- struct `cb_text_list` \* `cb_depend_list`
- struct `cb_text_list` \* `cb_include_list`
- struct `cb_text_list` \* `cb_extension_list`
- struct `cb_program` \* `current_program`
- struct `cb_statement` \* `current_statement`
- struct `cb_label` \* `current_section`
- struct `cb_label` \* `current_paragraph`
- size\_t `functions_are_all`
- struct `noreserve` \* `norestab`
- FILE \* `ppin`
- FILE \* `ppout`
- FILE \* `yyin`
- FILE \* `yyout`
- size\_t `sending_id`
- size\_t `suppress_warn`

### 7.3.1 Define Documentation

#### 7.3.1.1 #define ABORT( ) cobc\_abort(...FILE...,...LINE...)

Definition at line 37 of file `cobc.h`.

#### 7.3.1.2 #define CB\_CONFIG\_ANY( type, var, name ) extern type var;

Definition at line 198 of file `cobc.h`.

7.3.1.3 `#define CB_CONFIG_BOOLEAN( var, name ) extern int var;`

Definition at line 201 of file `cobc.h`.

7.3.1.4 `#define CB_CONFIG_INT( var, name ) extern int var;`

Definition at line 199 of file `cobc.h`.

7.3.1.5 `#define CB_CONFIG_STRING( var, name ) extern const char *var;`

Definition at line 200 of file `cobc.h`.

7.3.1.6 `#define CB_CONFIG_SUPPORT( var, name ) extern enum cb_support var;`

Definition at line 202 of file `cobc.h`.

7.3.1.7 `#define CB_EXCEPTION_CODE( id ) cb_exception_table[id].code`

Definition at line 53 of file `cobc.h`.

7.3.1.8 `#define CB_EXCEPTION_ENABLE( id ) cb_exception_table[id].enable`

Definition at line 54 of file `cobc.h`.

7.3.1.9 `#define CB_EXCEPTION_NAME( id ) cb_exception_table[id].name`

Definition at line 52 of file `cobc.h`.

7.3.1.10 `#define CB_FLAG( var, name, doc ) extern int var;`

Definition at line 57 of file `cobc.h`.

7.3.1.11 `#define CB_FORMAT_FIXED 0`

Definition at line 39 of file `cobc.h`.

7.3.1.12 `#define CB_FORMAT_FREE 1`

Definition at line 40 of file `cobc.h`.



7.3.1.13 `#define CB_WARNDEF( var, name, wall, doc ) extern int var;`

Definition at line 62 of file cobic.h.

7.3.1.14 `#define COB_NON_ALIGNED`

Definition at line 30 of file cobic.h.

## 7.3.2 Enumeration Type Documentation

7.3.2.1 `enum cb_assign_clause`

Enumerator:

**`CB_ASSIGN_COBOL2002`**

**`CB_ASSIGN_MF`**

**`CB_ASSIGN_IBM`**

Definition at line 158 of file cobic.h.

```
    {
        CB_ASSIGN_COBOL2002, /* COBOL 2002 standard */
        CB_ASSIGN_MF,      /* Micro Focus COBOL compatibility */
        CB_ASSIGN_IBM     /* IBM COBOL compatibility */
    };
```

7.3.2.2 `enum cb_binary_byteorder`

Enumerator:

**`CB_BYTEORDER_NATIVE`**

**`CB_BYTEORDER_BIG_ENDIAN`**

Definition at line 164 of file cobic.h.

```
    {
        CB_BYTEORDER_NATIVE,
        CB_BYTEORDER_BIG_ENDIAN
    };
```

7.3.2.3 `enum cb_binary_size`

Enumerator:

**`CB_BINARY_SIZE_2_4_8`**

**`CB_BINARY_SIZE_1_2_4_8`**

**CB\_BINARY\_SIZE\_1\_\_8**

Definition at line 169 of file cobc.h.

```

        {
    CB_BINARY_SIZE_2_4_8,      /* 2,4,8 bytes */
    CB_BINARY_SIZE_1_2_4_8,  /* 1,2,4,8 bytes */
    CB_BINARY_SIZE_1__8      /* 1,2,3,4,5,6,7,8 bytes */
};

```

## 7.3.2.4 enum cb\_operation\_type

Enumerator:

**CB\_OPERATION\_READ**  
**CB\_OPERATION\_WRITE**  
**CB\_OPERATION\_ASSIGN**

Definition at line 175 of file cobc.h.

```

        {
    CB_OPERATION_READ,
    CB_OPERATION_WRITE,
    CB_OPERATION_ASSIGN
};

```

## 7.3.2.5 enum cb\_support

Enumerator:

**CB\_OK**  
**CB\_WARNING**  
**CB\_ARCHAIC**  
**CB\_OBSOLETE**  
**CB\_SKIP**  
**CB\_IGNORE**  
**CB\_ERROR**  
**CB\_UNCONFORMABLE**

Definition at line 181 of file cobc.h.

```

        {
    CB_OK,
    CB_WARNING,
    CB_ARCHAIC,
    CB_OBSOLETE,
    CB_SKIP,
    CB_IGNORE,
    CB_ERROR,
    CB_UNCONFORMABLE
};

```

### 7.3.3 Function Documentation

#### 7.3.3.1 void `cb_error` ( const char \* *fmt*, ... )

Definition at line 85 of file `error.c`.

```
{
    va_list ap;

    va_start (ap, fmt);
    print_error (NULL, 0, "Error: ", fmt, ap);
    va_end (ap);

    errorcount++;
}
```

#### 7.3.3.2 int `cb_load_conf` ( const char \* *fname*, const int *check\_nodef*, const int *prefix\_dir* )

Definition at line 112 of file `config.c`.

```
{
    char          *s;
    char          *e;
    const char    *name;
    const char    *val;
    void          *var;
    FILE          *fp;
    char          *nores;
    struct noreserve *norespnr;
    int           i;
    int           j;
    int           ret;
    int           saveret;
    int           line;
    char          buff[COB_SMALL_BUFF];

    /* initialize the config table */
    if (check_nodef) {
        for (i = 0; config_table[i].name; i++) {
            config_table[i].val = NULL;
        }
    }

    if (prefix_dir) {
        snprintf (buff, COB_SMALL_MAX, "%s/%s", cob_config_dir, fname);
        name = buff;
    } else {
        name = fname;
    }

    /* open the config file */
    fp = fopen (name, "r");
    if (fp == NULL) {
        perror (name);
        return -1;
    }
}
```

```

/* read the config file */
ret = 0;
line = 0;
while (fgets (buff, COB_SMALL_BUFF, fp)) {
    line++;

    /* skip comments */
    if (buff[0] == '#') {
        continue;
    }

    /* skip blank lines */
    for (s = buff; *s; s++) {
        if (isgraph (*s)) {
            break;
        }
    }
    if (!*s) {
        continue;
    }

    /* get the tag */
    s = strpbrk (buff, " \t:=");
    if (!s) {
        fprintf (stderr, "%s:%d: invalid line\n", fname, line);
        ret = -1;
        continue;
    }
    *s = 0;

    /* find the entry */
    for (i = 0; config_table[i].name; i++) {
        if (strcmp (buff, config_table[i].name) == 0) {
            break;
        }
    }
    if (!config_table[i].name) {
        fprintf (stderr, "%s:%d: unknown tag '%s'\n", fname, line
, buff);
        ret = -1;
        continue;
    }

    /* get the value */
    for (s++; *s && strchr (" \t:=", *s); s++) {
        ;
    }
    e = s + strlen (s) - 1;
    for (; e >= s && strchr (" \t\r\n", *e); e--) {
        ;
    }
    e[1] = 0;
    config_table[i].val = s;

    /* set the value */
    name = config_table[i].name;
    var = config_table[i].var;
    val = config_table[i].val;
    switch (config_table[i].type) {
    case ANY:
        if (strcmp (name, "assign-clause") == 0) {
            if (strcmp (val, "cobol2002") == 0) {

```

```

        unsupported_value (fname, line, val);
        ret = -1;
    } else if (strcmp (val, "mf") == 0) {
        cb_assign_clause = CB_ASSIGN_MF;
    } else if (strcmp (val, "ibm") == 0) {
        cb_assign_clause = CB_ASSIGN_IBM;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
} else if (strcmp (name, "binary-size") == 0) {
    if (strcmp (val, "2-4-8") == 0) {
        cb_binary_size = CB_BINARY_SIZE_2_4_8;
    } else if (strcmp (val, "1-2-4-8") == 0) {
        cb_binary_size = CB_BINARY_SIZE_1_2_4_8;
    } else if (strcmp (val, "1--8") == 0) {
        cb_binary_size = CB_BINARY_SIZE_1__8;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
} else if (strcmp (name, "binary-byteorder") == 0) {
    if (strcmp (val, "native") == 0) {
        cb_binary_byteorder =
CB_BYTEORDER_NATIVE;
    } else if (strcmp (val, "big-endian") == 0) {
        cb_binary_byteorder =
CB_BYTEORDER_BIG_ENDIAN;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
}
}
break;
case INT:
    for (j = 0; val[j]; j++) {
        if (!isdigit (val[j])) {
            invalid_value (fname, line, name);
            ret = -1;
            break;
        }
    }
    *((int *)var) = atoi (val);
    break;
case STRING:
    val = read_string (val);

    if (strcmp (name, "include") == 0) {
        /* include another conf file */
        saveret = ret;
        if (cb_load_conf (val, 0, 1) != 0) {
            return -1;
        }
        ret = saveret;
    } else if (strcmp (name, "not-reserved") == 0) {
        nores = read_string (val);
        norespstr = cobc_malloc (sizeof (struct noreserve)
);
        norespstr->noresword = cobc_malloc (strlen (nores)
+ 1);
        strcpy (norespstr->noresword, nores);
        norespstr->next = norestab;

```

```

        norestab = norespnr;
    } else {
        *((const char **)var) = val;
    }
    break;
case BOOLEAN:
    if (strcmp (val, "yes") == 0) {
        *((int *)var) = 1;
    } else if (strcmp (val, "no") == 0) {
        *((int *)var) = 0;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
    break;
case SUPPORT:
    if (strcmp (val, "ok") == 0) {
        *((enum cb_support *)var) = CB_OK;
    } else if (strcmp (val, "warning") == 0) {
        *((enum cb_support *)var) = CB_WARNING;
    } else if (strcmp (val, "archaic") == 0) {
        *((enum cb_support *)var) = CB_ARCHAIC;
    } else if (strcmp (val, "obsolete") == 0) {
        *((enum cb_support *)var) = CB_OBSOLETE;
    } else if (strcmp (val, "skip") == 0) {
        *((enum cb_support *)var) = CB_SKIP;
    } else if (strcmp (val, "ignore") == 0) {
        *((enum cb_support *)var) = CB_IGNORE;
    } else if (strcmp (val, "error") == 0) {
        *((enum cb_support *)var) = CB_ERROR;
    } else if (strcmp (val, "unconformable") == 0) {
        *((enum cb_support *)var) = CB_UNCONFORMABLE;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
    break;
default:
    fprintf (stderr, _("%s:%d: invalid type for '%s'\n"),
            fname, line, name);
    ret = -1;
    break;
}
}
fclose (fp);

/* checks for no definition */
if (check_nodef) {
    for (i = 2; config_table[i].name; i++) {
        if (config_table[i].val == NULL) {
            fprintf (stderr, "%s: no definition of '%s'\n",
                    fname, config_table[i].name);
            ret = -1;
        }
    }
}

return ret;
}

```

### 7.3.3.3 int cb\_load\_std ( const char \* name )

Definition at line 106 of file config.c.

```
{
    return cb_load_conf (name, 1, 1);
}
```

### 7.3.3.4 struct cb\_text\_list\* cb\_text\_list\_add ( struct cb\_text\_list \* list, const char \* name ) [read]

Definition at line 355 of file cobc.c.

```
{
    struct cb_text_list *p;
    struct cb_text_list *l;

    p = cobc_malloc (sizeof (struct cb_text_list));
    p->text = strdup (text);
    p->next = NULL;
    if (!list) {
        return p;
    } else {
        for (l = list; l->next; l = l->next) { ; }
        l->next = p;
        return list;
    }
}
```

### 7.3.3.5 int cb\_verify ( const enum cb\_support tag, const char \* feature )

Definition at line 122 of file error.c.

```
{
    switch (tag) {
        case CB_OK:
            return 1;
        case CB_WARNING:
            return 1;
        case CB_ARCHAIC:
            if (cb_warn_archaic) {
                cb_warning (_("%s is archaic in %s"), feature, cb_config_
name);
            }
            return 1;
        case CB_OBSOLETE:
            if (cb_warn_obsolete) {
                cb_warning (_("%s is obsolete in %s"), feature, cb_config
_name);
            }
            return 1;
        case CB_SKIP:
            return 0;
        case CB_IGNORE:

```

```

        cb_warning (_("%s ignored"), feature);
        return 0;
    case CB_ERROR:
        return 0;
    case CB_UNCONFORMABLE:
        cb_error (_("%s does not conform to %s"), feature, cb_config_name
);
        return 0;
    }
    return 0;
}

```

### 7.3.3.6 void cb\_warning ( const char \* *fmt*, ... )

Definition at line 73 of file error.c.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error (NULL, 0, "Warning: ", fmt, ap);
    va_end (ap);

    warningcount++;
}

```

### 7.3.3.7 void cobc\_abort ( const char \* *filename*, const int *linenum* )

Definition at line 310 of file cobc.c.

```

{
    fprintf (stderr, "%s:%d: Internal compiler error\n", filename, linenum);
    (void)longjmp (cob_jmpbuf, 1);
}

```

### 7.3.3.8 size\_t cobc\_check\_valid\_name ( char \* *name* )

Definition at line 373 of file cobc.c.

```

{
    size_t n;

    for (n = 0; n < COB_NUM_CSYSNS; ++n) {
        if (!strcmp (name, cob_csyns[n])) {
            return 1;
        }
    }
    return 0;
}

```



### 7.3.3.9 void\* cobc\_malloc ( const size\_t size )

Definition at line 327 of file cobc.c.

```
{
    void *mptr;

    mptr = calloc (1, size);
    if (!mptr) {
        fprintf (stderr, "Cannot allocate %d bytes of memory - Aborting\n",
                (int)size);
        fflush (stderr);
        (void)longjmp (cob_jmpbuf, 1);
    }
    return mptr;
}
```

### 7.3.3.10 void\* cobc\_realloc ( void \*prevptr, const size\_t size )

Definition at line 341 of file cobc.c.

```
{
    void *mptr;

    mptr = realloc (prevptr, size);
    if (!mptr) {
        fprintf (stderr, "Cannot reallocate %d bytes of memory - Aborting\n",
                (int)size);
        fflush (stderr);
        (void)longjmp (cob_jmpbuf, 1);
    }
    return mptr;
}
```

### 7.3.3.11 void pp\_set\_replace\_list ( struct cb\_replace\_list \* replace\_list )

### 7.3.3.12 int ppcopy ( const char \* name, const char \* lib, struct cb\_replace\_list \* replace\_list )

### 7.3.3.13 int pplex ( void )

### 7.3.3.14 int ppopen ( const char \* name, struct cb\_replace\_list \* replace\_list )

### 7.3.3.15 int ppparse ( void )

### 7.3.3.16 int yylex ( void )

### 7.3.3.17 int yyparse ( void )

## 7.3.4 Variable Documentation

**7.3.4.1 int alt\_ebcdic**

Definition at line 121 of file cobc.c.

**7.3.4.2 int cb\_attr\_id**

Definition at line 113 of file cobc.c.

**7.3.4.3 FILE\* cb\_depend\_file**

Definition at line 134 of file cobc.c.

**7.3.4.4 struct cb\_text\_list\* cb\_depend\_list**

Definition at line 136 of file cobc.c.

**7.3.4.5 char\* cb\_depend\_target**

Definition at line 135 of file cobc.c.

**7.3.4.6 int cb\_display\_sign**

Definition at line 84 of file cobc.c.

**7.3.4.7 struct cb\_exception cb\_exception\_table[]****7.3.4.8 struct cb\_text\_list\* cb\_extension\_list**

Definition at line 138 of file cobc.c.

**7.3.4.9 int cb\_field\_id**

Definition at line 115 of file cobc.c.

**7.3.4.10 int cb\_flag\_main**

Definition at line 117 of file cobc.c.

**7.3.4.11 int cb\_id**

Definition at line 112 of file cobc.c.

**7.3.4.12 struct cb\_text\_list\* cb\_include\_list**

Definition at line 137 of file cobc.c.

**7.3.4.13 FILE\* cb\_listing\_file**

Definition at line 133 of file cobc.c.

**7.3.4.14 int cb\_literal\_id**

Definition at line 114 of file cobc.c.

**7.3.4.15 char\* cb\_oc\_build\_stamp**

Definition at line 125 of file cobc.c.

**7.3.4.16 int cb\_saveargc**

Definition at line 140 of file cobc.c.

**7.3.4.17 char\*\* cb\_saveargv**

Definition at line 141 of file cobc.c.

**7.3.4.18 char\* cb\_source\_file**

Definition at line 124 of file cobc.c.

**7.3.4.19 int cb\_source\_format**

Definition at line 79 of file cobc.c.

**7.3.4.20 int cb\_source\_line**

Definition at line 128 of file cobc.c.

**7.3.4.21 FILE\* cb\_storage\_file**

Definition at line 130 of file cobc.c.

**7.3.4.22 char\* cb\_storage\_file\_name**

Definition at line 131 of file cobc.c.

**7.3.4.23 int cb\_storage\_id**

Definition at line 116 of file cobc.c.

**7.3.4.24 const char\* cob\_config\_dir**

Definition at line 143 of file cobc.c.

**7.3.4.25 struct cb\_label\* current\_paragraph**

Definition at line 986 of file parser.c.

**7.3.4.26 struct cb\_program\* current\_program**

Definition at line 983 of file parser.c.

**7.3.4.27 struct cb\_label\* current\_section**

Definition at line 985 of file parser.c.

**7.3.4.28 struct cb\_statement\* current\_statement**

Definition at line 984 of file parser.c.

**7.3.4.29 char\* demangle\_name**

Definition at line 127 of file cobc.c.

**7.3.4.30 int errorcount**

Definition at line 119 of file cobc.c.

**7.3.4.31 size\_t functions\_are\_all**

Definition at line 987 of file parser.c.

**7.3.4.32 int has\_external**

Definition at line 138 of file codegen.c.

**7.3.4.33 struct noreserve\* norestab**

Definition at line 52 of file config.c.

**7.3.4.34 int optimize\_flag**

Definition at line 122 of file cobc.c.

**7.3.4.35 FILE\* ppin****7.3.4.36 FILE\* ppout****7.3.4.37 size\_t sending\_id**

Definition at line 74 of file typeck.c.

**7.3.4.38 char\* source\_name**

Definition at line 126 of file cobc.c.

**7.3.4.39 size\_t suppress\_warn**

Definition at line 75 of file typeck.c.

**7.3.4.40 int warningcount**

Definition at line 120 of file cobc.c.

**7.3.4.41 FILE\* yyin****7.3.4.42 FILE\* yyout**

## 7.4 cobc/codegen.c File Reference

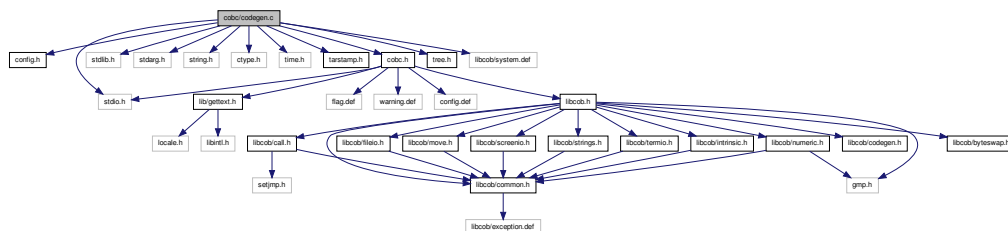
```
#include "config.h"  
#include <stdio.h>  
#include <stdlib.h>
```

```

#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <tarstamp.h>
#include "cobc.h"
#include "tree.h"
#include "libcob/system.def"

```

Include dependency graph for codegen.c:



## Classes

- struct **label\_list**
- struct **attr\_list**
- struct **literal\_list**
- struct **field\_list**
- struct **call\_list**
- struct **base\_list**
- struct **local\_list**
- struct [sort\\_list](#)
- struct [system\\_table](#)

## Defines

- #define [COB\\_USE\\_SETJMP](#) 0
- #define [COB\\_MAX\\_SUBSCRIPTS](#) 16
- #define [INITIALIZE\\_NONE](#) 0
- #define [INITIALIZE\\_ONE](#) 1
- #define [INITIALIZE\\_DEFAULT](#) 2
- #define [INITIALIZE\\_COMPOUND](#) 3
- #define [INITIALIZE\\_EXTERNAL](#) 4
- #define [COB\\_SYSTEM\\_GEN](#)(x, y, z) { x, #z },

## Functions

- void `codegen` (struct `cb_program` \*prog, int nested)

## Variables

- int `has_external` = 0

### 7.4.1 Define Documentation

#### 7.4.1.1 `#define COB_MAX_SUBSCRIPTS 16`

Definition at line 36 of file `codegen.c`.

#### 7.4.1.2 `#define COB_SYSTEM_GEN( x, y, z ) { x, #z },`

#### 7.4.1.3 `#define COB_USE_SETJMP 0`

Definition at line 35 of file `codegen.c`.

#### 7.4.1.4 `#define INITIALIZE_COMPOUND 3`

Definition at line 41 of file `codegen.c`.

#### 7.4.1.5 `#define INITIALIZE_DEFAULT 2`

Definition at line 40 of file `codegen.c`.

#### 7.4.1.6 `#define INITIALIZE_EXTERNAL 4`

Definition at line 42 of file `codegen.c`.

#### 7.4.1.7 `#define INITIALIZE_NONE 0`

Definition at line 38 of file `codegen.c`.

#### 7.4.1.8 `#define INITIALIZE_ONE 1`

Definition at line 39 of file `codegen.c`.

## 7.4.2 Function Documentation

### 7.4.2.1 void codegen ( struct cb\_program \* prog, int nested )

Definition at line 4683 of file codegen.c.

```

{
    int                i;
    cb_tree            l;
    struct attr_list   *j;
    struct literal_list *m;
    struct field_list  *k;
    struct call_list   *clp;
    struct base_list   *blp;
    unsigned char      *s;
    struct cb_program  *cp;
    cb_tree            l1;
    cb_tree            l2;
    const char         *prevprog;
    time_t             loctime;
    char               locbuff[48];

    current_prog = prog;
    param_id = 0;
    stack_id = 0;
    num_cob_fields = 0;
    progid = 0;
    loop_counter = 0;
    output_indent_level = 0;
    last_line = 0;
    needs_exit_prog = 0;
    gen_custom = 0;
    call_cache = NULL;
    label_cache = NULL;
    local_cache = NULL;
    excp_current_program_id = prog->orig_source_name;
    excp_current_section = NULL;
    excp_current_paragraph = NULL;
    memset ((char *)i_counters, 0, sizeof (i_counters));

    output_target = yyout;

    if (!nested) {
        gen_ebcdic = 0;
        gen_ebcdic_ascii = 0;
        gen_full_ebcdic = 0;
        gen_native = 0;
        attr_cache = NULL;
        base_cache = NULL;
        literal_cache = NULL;
        field_cache = NULL;

        loctime = time (NULL);
        strftime (locbuff, sizeof(locbuff) - 1, "%b %d %Y %H:%M:%S %Z",
                 localtime (&loctime));
        output_header (output_target, locbuff);
        output_header (cb_storage_file, locbuff);
        for (cp = prog; cp; cp = cp->next_program) {
            output_header (cp->local_storage_file, locbuff);
        }
    }
}

```



```

        output_storage ("/* Frame stack declaration */\n");
        output_storage ("struct cob_frame {\n");
        output_storage ("\tint\tperform_through;\n");
#ifdef __GNUC__
        output_storage ("\tint\treturn_address;\n");
#elif COB_USE_SETJMP
        output_storage ("\tjmp_buf\treturn_address;\n");
#else
        output_storage ("\tvoid\t*return_address;\n");
#endif

        output_storage ("};\n\n");
        output_storage ("/* Union for CALL statement */\n");
        output_storage ("union cob_call_union {\n");
        output_storage ("\tvoid *(*funcptr)();\n");
        output_storage ("\tint (*funcint)();\n");
        output_storage ("\tvoid *func_void;\n");
        output_storage ("};\n");
        output_storage ("union cob_call_union\tcob_unifunc;\n\n");

        output ("#define __USE_STRING_INLINES 1\n");
#ifdef __XOPEN_SOURCE_EXTENDED
        output ("#ifndef __XOPEN_SOURCE_EXTENDED\n");
        output ("#define __XOPEN_SOURCE_EXTENDED 1\n");
        output ("#endif\n");
#endif

        output ("#include <stdio.h>\n");
        output ("#include <stdlib.h>\n");
        output ("#include <string.h>\n");
        output ("#include <math.h>\n");
#if COB_USE_SETJMP
        output ("#include <setjmp.h>\n");
#endif
#ifdef WORDS_BIGENDIAN
        output ("#define WORDS_BIGENDIAN 1\n");
#endif
#ifdef HAVE_BUILTIN_EXPECT
        output ("#define HAVE_BUILTIN_EXPECT\n");
#endif

        if (optimize_flag) {
            output ("#define COB_LOCAL_INLINE\n");
        }
        output ("#include <libcob.h>\n\n");

        output ("#define COB_SOURCE_FILE          \"%s\"\n",
cb_source_file);
        output ("#define COB_PACKAGE_VERSION          \"%s\"\n",
PACKAGE_VERSION);
        output ("#define COB_PATCH_LEVEL              %d\n",
PATCH_LEVEL);
        output ("/* Global variables */\n");
        output ("#include \"%s\"\n", cb_storage_file_name);

        for (cp = prog; cp; cp = cp->next_program) {
            if (cp->gen_decset) {
                output("static void\n");
                output("cob_decimal_set_int (cob_decimal *d, cons
t int n)\n");
                output("{\n");
                output("    mpz_set_si (d->value, n);\n");
                output("    d->scale = 0;\n");
                output("}\n");
                break;
            }
        }

```

```

    }
    for (cp = prog; cp; cp = cp->next_program) {
        if (cp->gen_udecset) {
            output("static void\n");
            output("cob_decimal_set_uint (cob_decimal *d, con
st unsigned int n)\n");
            output("{\n");
            output("    mpz_set_ui (d->value, n);\n");
            output("    d->scale = 0;\n");
            output("}\n\n");
            break;
        }
    }
    for (cp = prog; cp; cp = cp->next_program) {
        if (cp->gen_ptrmanip) {
            output("static void\n");
            output("cob_pointer_manip (cob_field *f1, cob_fie
ld *f2, size_t addsub)\n");
            output("{\n");
            output("    unsigned char *tmptr;\n");
            output("    memcpy (&tmptr, f1->data, sizeof(
void *));\n");
            output("    if (addsub) {\n");
            output("        tmptr -= cob_get_int (f2
);\n");
            output("    } else {\n");
            output("        tmptr += cob_get_int (f2
);\n");
            output("    }\n");
            output("    memcpy (f1->data, &tmptr, sizeof(
void *));\n");
            output("}\n\n");
            break;
        }
    }
    output ("/* Function prototypes */\n\n");
    for (cp = prog; cp; cp = cp->next_program) {
        /* Build parameter list */
        for (l = cp->entry_list; l; l = CB_CHAIN (l)) {
            for (l1 = CB_VALUE (l); l1; l1 = CB_CHAIN (l1)) {
                for (l2 = cp->parameter_list; l2; l2 =
CB_CHAIN (l2)) {
                    if (strcasecmp (cb_field (
CB_VALUE (l1))->name,
                                cb_field (
CB_VALUE (l2))->name) == 0) {
                        break;
                    }
                }
                if (l2 == NULL) {
                    cp->parameter_list = cb_list_add
(cp->parameter_list, CB_VALUE (l1));
                }
            }
        }
        if (cp->flag_main) {
            output ("int %s ();\n", cp->program_id);
        } else {
            for (l = cp->entry_list; l; l = CB_CHAIN (l)) {
                output_entry_function (cp, l, cp->

```

```

parameter_list, 0);
        }
        }
        output ("static int %s_ (const int", cp->program_id);
        if (!cp->flag_chained) {
            for (l = cp->parameter_list; l; l = CB_CHAIN (l))
        {
                output ("", unsigned char *");
            }
        }
        output ("");\n");
    }
    output ("\n");
}

/* Class-names */
if (!prog->nested_level && prog->class_name_list) {
    output ("/* Class names */\n");
    for (l = prog->class_name_list; l; l = CB_CHAIN (l)) {
        output_class_name_definition (CB_CLASS_NAME (CB_VALUE (l)
));
    }
}

/* Main function */
if (prog->flag_main) {
    output_main_function (prog);
}

/* Functions */
if (!nested) {
    output ("/* Functions */\n\n");
}
for (l = prog->entry_list; l; l = CB_CHAIN (l)) {
    output_entry_function (prog, l, prog->parameter_list, 1);
}
output_internal_function (prog, prog->parameter_list);

if (!prog->next_program) {
    output ("/* End functions */\n\n");
}

if (gen_native || gen_full_ebcdic || gen_ebcdic_ascii || prog->
alphabet_name_list) {
    (void)lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
}

output_target = cb_storage_file;

/* Program local stuff */
if (call_cache) {
    output_local ("\n/* Call pointers */\n");
    for (clp = call_cache; clp; clp = clp->next) {
        output_local ("static union cob_call_union\ncall_%s = { N
ULL };\n", clp->callname);
    }
    output_local ("\n");
}

for (i = 0; i < COB_MAX_SUBSCRIPTS; i++) {
    if (i_counters[i]) {
        output_local ("int\t\t\ti%d;\n", i);
    }
}

```

```

    }
}

if (num_cob_fields) {
    output_local ("\n/* Local cob_field items */\n");
    for (i = 0; i < num_cob_fields; i++) {
        output_local ("cob_field\tf%d;\n", i);
    }
    output_local ("\n");
}

/* Skip to next nested program */

if (prog->next_program) {
    codegen (prog->next_program, 1);
    return;
}

/* Finalize the storage file */

if (base_cache) {
    output_storage ("\n/* Storage */\n");
    base_cache = list_cache_sort (base_cache, &base_cache_cmp);
    prevprog = NULL;
    for (blp = base_cache; blp; blp = blp->next) {
        if (blp->curr_prog != prevprog) {
            prevprog = blp->curr_prog;
            output_storage ("\n/* PROGRAM-ID : %s */\n", prev
prog);
        }
#ifdef HAVE_ATTRIBUTE_ALIGNED
        output_storage ("static unsigned char %s%d[%d] __attribut
e__((aligned));",
#else
        output_storage ("static unsigned char %s%d[%d];",
#endif
                        CB_PREFIX_BASE, blp->f->id,
                        blp->f->memory_size);
        output_storage ("\t/* %s */\n", blp->f->name);
    }
    output_storage ("\n/* End of storage */\n\n");
}

if (attr_cache) {
    output_storage ("\n/* Attributes */\n\n");
    attr_cache = attr_list_reverse (attr_cache);
    for (j = attr_cache; j; j = j->next) {
        output_storage ("static const cob_field_attr %s%d = ",
                        CB_PREFIX_ATTR, j->id);
        output_storage ("{%d, %d, %d, %d, ", j->type, j->digits,
                        j->scale, j->flags);
        if (j->pic) {
            output_storage ("\n");
            for (s = j->pic; *s; s += 5) {
                output_storage ("%c\\%03o\\%03o\\%03o\\%0
3o",
                                s[0], s[1], s[2], s[3], s[4]);
            }
            output_storage ("\n");
        } else {
            output_storage ("NULL");
        }
    }
}

```

```

        output_storage (");\n");
    }
}

if (field_cache) {
    output_storage ("\n/* Fields */\n");
    field_cache = list_cache_sort (field_cache, &field_cache_cmp);
    prevprog = NULL;
    for (k = field_cache; k; k = k->next) {
        if (k->curr_prog != prevprog) {
            prevprog = k->curr_prog;
            output_storage ("\n/* PROGRAM-ID : %s */\n", prev
prog);
        }
        output ("static cob_field %s%d\t= ", CB_PREFIX_FIELD, k->
f->id);
        if (!k->f->flag_local && !k->f->flag_item_external) {
            output_field (k->x);
        } else {
            output ("{");
            output_size (k->x);
            output (" , NULL, ");
            output_attr (k->x);
            output ("}");
        }
        output (";\t/* %s */\n", k->f->name);
    }
    output_storage ("\n/* End of fields */\n\n");
}

if (literal_cache) {
    output_storage ("/* Constants */\n");
    literal_cache = literal_list_reverse (literal_cache);
    for (m = literal_cache; m; m = m->next) {
        output ("static cob_field %s%d\t= ", CB_PREFIX_CONST, m->
id);
        output_field (m->x);
        output (";\n");
    }
    output ("\n");
}

if (gen_ebcdic) {
    output_storage ("/* EBCDIC translate table */\n");
    output ("static const unsigned char\tcob_a2e[256] = {\n");
    if (alt_ebcdic) {
        output ("\t0x00, 0x01, 0x02, 0x03, 0x37, 0x2D, 0x2E, 0x2F
,\n");
        output ("\t0x16, 0x05, 0x25, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
,\n");
        output ("\t0x10, 0x11, 0x12, 0x13, 0x3C, 0x3D, 0x32, 0x26
,\n");
        output ("\t0x18, 0x19, 0x3F, 0x27, 0x1C, 0x1D, 0x1E, 0x1F
,\n");
        output ("\t0x40, 0x5A, 0x7F, 0x7B, 0x5B, 0x6C, 0x50, 0x7D
,\n");
        output ("\t0x4D, 0x5D, 0x5C, 0x4E, 0x6B, 0x60, 0x4B, 0x61
,\n");
        output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7
,\n");
        output ("\t0xF8, 0xF9, 0x7A, 0x5E, 0x4C, 0x7E, 0x6E, 0x6F
,\n");
        output ("\t0x7C, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7

```

```

, \n" );
, \n" );
output ( "\t0xC8, 0xC9, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6
, \n" );
output ( "\t0xD7, 0xD8, 0xD9, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6
, \n" );
output ( "\t0xE7, 0xE8, 0xE9, 0xAD, 0xE0, 0xBD, 0x5F, 0x6D
, \n" );
output ( "\t0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87
, \n" );
output ( "\t0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96
, \n" );
output ( "\t0x97, 0x98, 0x99, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6
, \n" );
output ( "\t0xA7, 0xA8, 0xA9, 0xC0, 0x6A, 0xD0, 0xA1, 0x07
, \n" );
output ( "\t0x68, 0xDC, 0x51, 0x42, 0x43, 0x44, 0x47, 0x48
, \n" );
output ( "\t0x52, 0x53, 0x54, 0x57, 0x56, 0x58, 0x63, 0x67
, \n" );
output ( "\t0x71, 0x9C, 0x9E, 0xCB, 0xCC, 0xCD, 0xDB, 0xDD
, \n" );
output ( "\t0xDF, 0xEC, 0xFC, 0xB0, 0xB1, 0xB2, 0x3E, 0xB4
, \n" );
output ( "\t0x45, 0x55, 0xCE, 0xDE, 0x49, 0x69, 0x9A, 0x9B
, \n" );
output ( "\t0xAB, 0x9F, 0xBA, 0xB8, 0xB7, 0xAA, 0x8A, 0x8B
, \n" );
output ( "\t0xB6, 0xB5, 0x62, 0x4F, 0x64, 0x65, 0x66, 0x20
, \n" );
output ( "\t0x21, 0x22, 0x70, 0x23, 0x72, 0x73, 0x74, 0xBE
, \n" );
output ( "\t0x76, 0x77, 0x78, 0x80, 0x24, 0x15, 0x8C, 0x8D
, \n" );
output ( "\t0x8E, 0x41, 0x06, 0x17, 0x28, 0x29, 0x9D, 0x2A
, \n" );
output ( "\t0x2B, 0x2C, 0x09, 0x0A, 0xAC, 0x4A, 0xAE, 0xAF
, \n" );
output ( "\t0x1B, 0x30, 0x31, 0xFA, 0x1A, 0x33, 0x34, 0x35
, \n" );
output ( "\t0x36, 0x59, 0x08, 0x38, 0xBC, 0x39, 0xA0, 0xBF
, \n" );
output ( "\t0xCA, 0x3A, 0xFE, 0x3B, 0x04, 0xCF, 0xDA, 0x14
, \n" );
output ( "\t0xE1, 0x8F, 0x46, 0x75, 0xFD, 0xEB, 0xEE, 0xED
, \n" );
output ( "\t0x90, 0xEF, 0xB3, 0xFB, 0xB9, 0xEA, 0xBB, 0xFF
\n" );
} else {
/* MF */
output ( "\t0x00, 0x01, 0x02, 0x03, 0x1D, 0x19, 0x1A, 0x1B
, \n" );
output ( "\t0x0F, 0x04, 0x16, 0x06, 0x07, 0x08, 0x09, 0x0A
, \n" );
output ( "\t0x0B, 0x0C, 0x0D, 0x0E, 0x1E, 0x1F, 0x1C, 0x17
, \n" );
output ( "\t0x10, 0x11, 0x20, 0x18, 0x12, 0x13, 0x14, 0x15
, \n" );
output ( "\t0x21, 0x27, 0x3A, 0x36, 0x28, 0x30, 0x26, 0x38
, \n" );
output ( "\t0x24, 0x2A, 0x29, 0x25, 0x2F, 0x2C, 0x22, 0x2D
, \n" );
output ( "\t0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A

```

```

, \n");
output ("\t0x7B, 0x7C, 0x35, 0x2B, 0x23, 0x39, 0x32, 0x33
, \n");
output ("\t0x37, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D
, \n");
output ("\t0x5E, 0x5F, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66
, \n");
output ("\t0x67, 0x68, 0x69, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F
, \n");
output ("\t0x70, 0x71, 0x72, 0x7D, 0x6A, 0x7E, 0x7F, 0x31
, \n");
output ("\t0x34, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41
, \n");
output ("\t0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49
, \n");
output ("\t0x4A, 0x4B, 0x4C, 0x4E, 0x4F, 0x50, 0x51, 0x52
, \n");
output ("\t0x53, 0x54, 0x55, 0x56, 0x2E, 0x60, 0x4D, 0x05
, \n");
output ("\t0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87
, \n");
output ("\t0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F
, \n");
output ("\t0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97
, \n");
output ("\t0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F
, \n");
output ("\t0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7
, \n");
output ("\t0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF
, \n");
output ("\t0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7
, \n");
output ("\t0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF
, \n");
output ("\t0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7
, \n");
output ("\t0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF
, \n");
output ("\t0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7
, \n");
output ("\t0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF
, \n");
output ("\t0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7
, \n");
output ("\t0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF
, \n");
output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7
, \n");
output ("\t0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
\n");
}
output ("};\n");
output_storage ("\n");
}
if (gen_full_ebcdic) {
output ("static const unsigned char\tcob_ebcdic[256] = {\n");
output ("\t0x00, 0x01, 0x02, 0x03, 0x37, 0x2D, 0x2E, 0x2F,\n");
output ("\t0x16, 0x05, 0x25, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,\n");
output ("\t0x10, 0x11, 0x12, 0x13, 0x3C, 0x3D, 0x32, 0x26,\n");
output ("\t0x18, 0x19, 0x3F, 0x27, 0x1C, 0x1D, 0x1E, 0x1F,\n");
output ("\t0x40, 0x5A, 0x7F, 0x7B, 0x5B, 0x6C, 0x50, 0x7D,\n");
}

```

```

output ("\t0x4D, 0x5D, 0x5C, 0x4E, 0x6B, 0x60, 0x4B, 0x61, \n");
output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, \n");
output ("\t0xF8, 0xF9, 0x7A, 0x5E, 0x4C, 0x7E, 0x6E, 0x6F, \n");
output ("\t0x7C, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, \n");
output ("\t0xC8, 0xC9, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, \n");
output ("\t0xD7, 0xD8, 0xD9, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, \n");
output ("\t0xE7, 0xE8, 0xE9, 0xAD, 0xE0, 0xBD, 0x5F, 0x6D, \n");
output ("\t0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, \n");
output ("\t0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, \n");
output ("\t0x97, 0x98, 0x99, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, \n");
output ("\t0xA7, 0xA8, 0xA9, 0xC0, 0x6A, 0xD0, 0xA1, 0x07, \n");
output ("\t0x68, 0xDC, 0x51, 0x42, 0x43, 0x44, 0x47, 0x48, \n");
output ("\t0x52, 0x53, 0x54, 0x57, 0x56, 0x58, 0x63, 0x67, \n");
output ("\t0x71, 0x9C, 0x9E, 0xCB, 0xCC, 0xCD, 0xDB, 0xDD, \n");
output ("\t0xDF, 0xEC, 0xFC, 0xB0, 0xB1, 0xB2, 0x3E, 0xB4, \n");
output ("\t0x45, 0x55, 0xCE, 0xDE, 0x49, 0x69, 0x9A, 0x9B, \n");
output ("\t0xAB, 0x9F, 0xBA, 0xB8, 0xB7, 0xAA, 0x8A, 0x8B, \n");
output ("\t0xB6, 0xB5, 0x62, 0x4F, 0x64, 0x65, 0x66, 0x20, \n");
output ("\t0x21, 0x22, 0x70, 0x23, 0x72, 0x73, 0x74, 0xBE, \n");
output ("\t0x76, 0x77, 0x78, 0x80, 0x24, 0x15, 0x8C, 0x8D, \n");
output ("\t0x8E, 0x41, 0x06, 0x17, 0x28, 0x29, 0x9D, 0x2A, \n");
output ("\t0x2B, 0x2C, 0x09, 0x0A, 0xAC, 0x4A, 0xAE, 0xAF, \n");
output ("\t0x1B, 0x30, 0x31, 0xFA, 0x1A, 0x33, 0x34, 0x35, \n");
output ("\t0x36, 0x59, 0x08, 0x38, 0xBC, 0x39, 0xA0, 0xBF, \n");
output ("\t0xCA, 0x3A, 0xFE, 0x3B, 0x04, 0xCF, 0xDA, 0x14, \n");
output ("\t0xE1, 0x8F, 0x46, 0x75, 0xFD, 0xEB, 0xEE, 0xED, \n");
output ("\t0x90, 0xEF, 0xB3, 0xFB, 0xB9, 0xEA, 0xBB, 0xFF, \n");
output ("");
i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
output
("static cob_field f_ebcdic = { 256, (unsigned char *)cob_ebc
dic, &%s%d }; \n",
    CB_PREFIX_ATTR, i);
    output_storage (" \n");
}
if (gen_ebcdic_ascii) {
    output ("static const unsigned char \tcob_ebcdic_ascii[256] = { \n"
);
    output ("\t0x00, 0x01, 0x02, 0x03, 0xEC, 0x09, 0xCA, 0x7F, \n");
    output ("\t0xE2, 0xD2, 0xD3, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, \n");
    output ("\t0x10, 0x11, 0x12, 0x13, 0xEF, 0xC5, 0x08, 0xCB, \n");
    output ("\t0x18, 0x19, 0xDC, 0xD8, 0x1C, 0x1D, 0x1E, 0x1F, \n");
    output ("\t0xB7, 0xB8, 0xB9, 0xBB, 0xC4, 0x0A, 0x17, 0x1B, \n");
    output ("\t0xCC, 0xCD, 0xCF, 0xD0, 0xD1, 0x05, 0x06, 0x07, \n");
    output ("\t0xD9, 0xDA, 0x16, 0xDD, 0xDE, 0xDF, 0xE0, 0x04, \n");
    output ("\t0xE3, 0xE5, 0xE9, 0xEB, 0x14, 0x15, 0x9E, 0x1A, \n");
    output ("\t0x20, 0xC9, 0x83, 0x84, 0x85, 0xA0, 0xF2, 0x86, \n");
    output ("\t0x87, 0xA4, 0xD5, 0x2E, 0x3C, 0x28, 0x2B, 0xB3, \n");
    output ("\t0x26, 0x82, 0x88, 0x89, 0x8A, 0xA1, 0x8C, 0x8B, \n");
    output ("\t0x8D, 0xE1, 0x21, 0x24, 0x2A, 0x29, 0x3B, 0x5E, \n");
    output ("\t0x2D, 0x2F, 0xB2, 0x8E, 0xB4, 0xB5, 0xB6, 0x8F, \n");
    output ("\t0x80, 0xA5, 0x7C, 0x2C, 0x25, 0x5F, 0x3E, 0x3F, \n");
    output ("\t0xBA, 0x90, 0xBC, 0xBD, 0xBE, 0xF3, 0xC0, 0xC1, \n");
    output ("\t0xC2, 0x60, 0x3A, 0x23, 0x40, 0x27, 0x3D, 0x22, \n");
    output ("\t0xC3, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, \n");
    output ("\t0x68, 0x69, 0xAE, 0xAF, 0xC6, 0xC7, 0xC8, 0xF1, \n");
    output ("\t0xF8, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, \n");
    output ("\t0x71, 0x72, 0xA6, 0xA7, 0x91, 0xCE, 0x92, 0xA9, \n");
    output ("\t0xE6, 0x7E, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, \n");
    output ("\t0x79, 0x7A, 0xAD, 0xA8, 0xD4, 0x5B, 0xD6, 0xD7, \n");
    output ("\t0x9B, 0x9C, 0x9D, 0xFA, 0x9F, 0xB1, 0xB0, 0xAC, \n");
    output ("\t0xAB, 0xFC, 0xAA, 0xFE, 0xE4, 0x5D, 0xBF, 0xE7, \n");

```



```

        output ("\t0x7B, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,\n");
        output ("\t0x48, 0x49, 0xE8, 0x93, 0x94, 0x95, 0xA2, 0xED,\n");
        output ("\t0x7D, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50,\n");
        output ("\t0x51, 0x52, 0xEE, 0x96, 0x81, 0x97, 0xA3, 0x98,\n");
        output ("\t0x5C, 0xF0, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58,\n");
        output ("\t0x59, 0x5A, 0xFD, 0xF5, 0x99, 0xF7, 0xF6, 0xF9,\n");
        output ("\t0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,\n");
        output ("\t0x38, 0x39, 0xDB, 0xFB, 0x9A, 0xF4, 0xEA, 0xFF\n");
        output ("");\n");
        i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
        output
            ("static cob_field f_ebcdic_ascii = { 256, (unsigned char *)c
ob_ebcdic_ascii, &%s%d };\n",
            CB_PREFIX_ATTR, i);
        output_storage ("\n");
    }
    if (gen_native) {
        output ("static const unsigned char\tcob_native[256] = {\n");
        output ("\t0, 1, 2, 3, 4, 5, 6, 7,\n");
        output ("\t8, 9, 10, 11, 12, 13, 14, 15,\n");
        output ("\t16, 17, 18, 19, 20, 21, 22, 23,\n");
        output ("\t24, 25, 26, 27, 28, 29, 30, 31,\n");
        output ("\t32, 33, 34, 35, 36, 37, 38, 39,\n");
        output ("\t40, 41, 42, 43, 44, 45, 46, 47,\n");
        output ("\t48, 49, 50, 51, 52, 53, 54, 55,\n");
        output ("\t56, 57, 58, 59, 60, 61, 62, 63,\n");
        output ("\t64, 65, 66, 67, 68, 69, 70, 71,\n");
        output ("\t72, 73, 74, 75, 76, 77, 78, 79,\n");
        output ("\t80, 81, 82, 83, 84, 85, 86, 87,\n");
        output ("\t88, 89, 90, 91, 92, 93, 94, 95,\n");
        output ("\t96, 97, 98, 99, 100, 101, 102, 103,\n");
        output ("\t104, 105, 106, 107, 108, 109, 110, 111,\n");
        output ("\t112, 113, 114, 115, 116, 117, 118, 119,\n");
        output ("\t120, 121, 122, 123, 124, 125, 126, 127,\n");
        output ("\t128, 129, 130, 131, 132, 133, 134, 135,\n");
        output ("\t136, 137, 138, 139, 140, 141, 142, 143,\n");
        output ("\t144, 145, 146, 147, 148, 149, 150, 151,\n");
        output ("\t152, 153, 154, 155, 156, 157, 158, 159,\n");
        output ("\t160, 161, 162, 163, 164, 165, 166, 167,\n");
        output ("\t168, 169, 170, 171, 172, 173, 174, 175,\n");
        output ("\t176, 177, 178, 179, 180, 181, 182, 183,\n");
        output ("\t184, 185, 186, 187, 188, 189, 190, 191,\n");
        output ("\t192, 193, 194, 195, 196, 197, 198, 199,\n");
        output ("\t200, 201, 202, 203, 204, 205, 206, 207,\n");
        output ("\t208, 209, 210, 211, 212, 213, 214, 215,\n");
        output ("\t216, 217, 218, 219, 220, 221, 222, 223,\n");
        output ("\t224, 225, 226, 227, 228, 229, 230, 231,\n");
        output ("\t232, 233, 234, 235, 236, 237, 238, 239,\n");
        output ("\t240, 241, 242, 243, 244, 245, 246, 247,\n");
        output ("\t248, 249, 250, 251, 252, 253, 254, 255\n");
        output ("");\n");
        i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
        output
            ("static cob_field f_native = { 256, (unsigned char *)cob_nat
ive, &%s%d };\n",
            CB_PREFIX_ATTR, i);
        output_storage ("\n");
    }
}

```

### 7.4.3 Variable Documentation

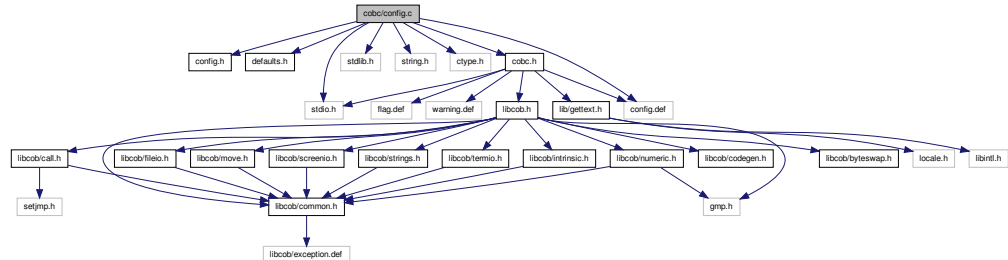
#### 7.4.3.1 int has\_external = 0

Definition at line 138 of file codegen.c.

## 7.5 cobc/config.c File Reference

```
#include "config.h"
#include "defaults.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobc.h"
#include "config.def"
```

Include dependency graph for config.c:



### Defines

- #define **CB\_CONFIG\_ANY**(type, var, name) type var;
- #define **CB\_CONFIG\_INT**(var, name) int var;
- #define **CB\_CONFIG\_STRING**(var, name) const char \*var;
- #define **CB\_CONFIG\_BOOLEAN**(var, name) int var;
- #define **CB\_CONFIG\_SUPPORT**(var, name) enum cb\_support var;
- #define **CB\_CONFIG\_ANY**(type, var, name) {ANY, name, &var, NULL},
- #define **CB\_CONFIG\_INT**(var, name) {INT, name, &var, NULL},
- #define **CB\_CONFIG\_STRING**(var, name) {STRING, name, &var, NULL},
- #define **CB\_CONFIG\_BOOLEAN**(var, name) {BOOLEAN, name, &var, NULL},
- #define **CB\_CONFIG\_SUPPORT**(var, name) {SUPPORT, name, &var, NULL},

## Enumerations

- enum `cb_config_type` {  
    `ANY`, `INT`, `STRING`, `BOOLEAN`,  
    `SUPPORT` }

## Functions

- int `cb_load_std` (const char \*`name`)
- int `cb_load_conf` (const char \*`fname`, const int `check_noddef`, const int `prefix_dir`)

## Variables

- struct `noreserve` \* `norestab` = NULL

### 7.5.1 Define Documentation

7.5.1.1 `#define CB_CONFIG_ANY( type, var, name ) type var;`

Definition at line 36 of file config.c.

7.5.1.2 `#define CB_CONFIG_ANY( type, var, name ) {ANY, name, &var, NULL},`

Definition at line 36 of file config.c.

7.5.1.3 `#define CB_CONFIG_BOOLEAN( var, name ) int var;`

Definition at line 39 of file config.c.

7.5.1.4 `#define CB_CONFIG_BOOLEAN( var, name ) {BOOLEAN, name, &var, NULL},`

Definition at line 39 of file config.c.

7.5.1.5 `#define CB_CONFIG_INT( var, name ) int var;`

Definition at line 37 of file config.c.

7.5.1.6 `#define CB_CONFIG_INT( var, name ) {INT, name, &var, NULL},`

Definition at line 37 of file config.c.

7.5.1.7 `#define CB_CONFIG_STRING( var, name ) {STRING, name, &var, NULL},`

Definition at line 38 of file config.c.

7.5.1.8 `#define CB_CONFIG_STRING( var, name ) const char *var;`

Definition at line 38 of file config.c.

7.5.1.9 `#define CB_CONFIG_SUPPORT( var, name ) {SUPPORT, name, &var, NULL},`

Definition at line 40 of file config.c.

7.5.1.10 `#define CB_CONFIG_SUPPORT( var, name ) enum cb_support var;`

Definition at line 40 of file config.c.

## 7.5.2 Enumeration Type Documentation

### 7.5.2.1 enum cb\_config\_type

**Enumerator:**

***ANY***

***INT***

***STRING***

***BOOLEAN***

***SUPPORT***

Definition at line 43 of file config.c.

```

    {
        ANY,
        INT,
        STRING,
        BOOLEAN,
        SUPPORT
    };
    /* integer */
    /* "..." */
    /* 'yes', 'no' */
    /* 'ok', 'archaic', 'obsolete',
    'skip', 'ignore', 'unconformable' */

```

## 7.5.3 Function Documentation

7.5.3.1 `int cb_load_conf ( const char * fname, const int check_nodef, const int prefix_dir )`

Definition at line 112 of file config.c.

```
{
    char          *s;
    char          *e;
    const char    *name;
    const char    *val;
    void          *var;
    FILE          *fp;
    char          *nores;
    struct noreserve *norespnr;
    int           i;
    int           j;
    int           ret;
    int           saveret;
    int           line;
    char          buff[COB_SMALL_BUFF];

    /* initialize the config table */
    if (check_nodef) {
        for (i = 0; config_table[i].name; i++) {
            config_table[i].val = NULL;
        }
    }

    if (prefix_dir) {
        snprintf (buff, COB_SMALL_MAX, "%s/%s", cob_config_dir, fname);
        name = buff;
    } else {
        name = fname;
    }

    /* open the config file */
    fp = fopen (name, "r");
    if (fp == NULL) {
        perror (name);
        return -1;
    }

    /* read the config file */
    ret = 0;
    line = 0;
    while (fgets (buff, COB_SMALL_BUFF, fp)) {
        line++;

        /* skip comments */
        if (buff[0] == '#') {
            continue;
        }

        /* skip blank lines */
        for (s = buff; *s; s++) {
            if (isgraph (*s)) {
                break;
            }
        }
        if (!*s) {
            continue;
        }

        /* get the tag */
        s = strpbrk (buff, " \\t:=");
        if (!s) {
            fprintf (stderr, "%s:%d: invalid line\n", fname, line);
            ret = -1;
        }
    }
}
```

```

        continue;
    }
    *s = 0;

    /* find the entry */
    for (i = 0; config_table[i].name; i++) {
        if (strcmp (buff, config_table[i].name) == 0) {
            break;
        }
    }
    if (!config_table[i].name) {
        fprintf (stderr, "%s:%d: unknown tag '%s'\n", fname, line
, buff);
        ret = -1;
        continue;
    }

    /* get the value */
    for (s++; *s && strchr (" \t:=", *s); s++) {
        ;
    }
    e = s + strlen (s) - 1;
    for (; e >= s && strchr (" \t\r\n", *e); e--) {
        ;
    }
    e[1] = 0;
    config_table[i].val = s;

    /* set the value */
    name = config_table[i].name;
    var = config_table[i].var;
    val = config_table[i].val;
    switch (config_table[i].type) {
    case ANY:
        if (strcmp (name, "assign-clause") == 0) {
            if (strcmp (val, "cobol2002") == 0) {
                unsupported_value (fname, line, val);
                ret = -1;
            } else if (strcmp (val, "mf") == 0) {
                cb_assign_clause = CB_ASSIGN_MF;
            } else if (strcmp (val, "ibm") == 0) {
                cb_assign_clause = CB_ASSIGN_IBM;
            } else {
                invalid_value (fname, line, name);
                ret = -1;
            }
        }
        } else if (strcmp (name, "binary-size") == 0) {
            if (strcmp (val, "2-4-8") == 0) {
                cb_binary_size = CB_BINARY_SIZE_2_4_8;
            } else if (strcmp (val, "1-2-4-8") == 0) {
                cb_binary_size = CB_BINARY_SIZE_1_2_4_8;
            } else if (strcmp (val, "1--8") == 0) {
                cb_binary_size = CB_BINARY_SIZE_1__8;
            } else {
                invalid_value (fname, line, name);
                ret = -1;
            }
        }
        } else if (strcmp (name, "binary-byteorder") == 0) {
            if (strcmp (val, "native") == 0) {
                cb_binary_byteorder =
CB_BYTEORDER_NATIVE;
            } else if (strcmp (val, "big-endian") == 0) {

```

```

                                cb_binary_byteorder =
CB_BYTEORDER_BIG_ENDIAN;
                                } else {
                                    invalid_value (fname, line, name);
                                    ret = -1;
                                }
                                }
                                break;
case INT:
    for (j = 0; val[j]; j++) {
        if (!isdigit (val[j])) {
            invalid_value (fname, line, name);
            ret = -1;
            break;
        }
    }
    *((int *)var) = atoi (val);
    break;
case STRING:
    val = read_string (val);

    if (strcmp (name, "include") == 0) {
        /* include another conf file */
        saveret = ret;
        if (cb_load_conf (val, 0, 1) != 0) {
            return -1;
        }
        ret = saveret;
    } else if (strcmp (name, "not-reserved") == 0) {
        nores = read_string (val);
        norespstr = cobc_malloc (sizeof (struct noreserve)
);
                                norespstr->noresword = cobc_malloc (strlen (nores)
+ 1);

                                strcpy (norespstr->noresword, nores);
                                norespstr->next = norestab;
                                norestab = norespstr;
    } else {
        *((const char **)var) = val;
    }
    break;
case BOOLEAN:
    if (strcmp (val, "yes") == 0) {
        *((int *)var) = 1;
    } else if (strcmp (val, "no") == 0) {
        *((int *)var) = 0;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
    break;
case SUPPORT:
    if (strcmp (val, "ok") == 0) {
        *((enum cb_support *)var) = CB_OK;
    } else if (strcmp (val, "warning") == 0) {
        *((enum cb_support *)var) = CB_WARNING;
    } else if (strcmp (val, "archaic") == 0) {
        *((enum cb_support *)var) = CB_ARCHAIC;
    } else if (strcmp (val, "obsolete") == 0) {
        *((enum cb_support *)var) = CB_OBSOLETE;
    } else if (strcmp (val, "skip") == 0) {
        *((enum cb_support *)var) = CB_SKIP;

```

```

    } else if (strcmp (val, "ignore") == 0) {
        *((enum cb_support *)var) = CB_IGNORE;
    } else if (strcmp (val, "error") == 0) {
        *((enum cb_support *)var) = CB_ERROR;
    } else if (strcmp (val, "unconformable") == 0) {
        *((enum cb_support *)var) = CB_UNCONFORMABLE;
    } else {
        invalid_value (fname, line, name);
        ret = -1;
    }
    break;
default:
    fprintf (stderr, _("%s:%d: invalid type for '%s'\n"),
            fname, line, name);
    ret = -1;
    break;
}
}
fclose (fp);

/* checks for no definition */
if (check_nodef) {
    for (i = 2; config_table[i].name; i++) {
        if (config_table[i].val == NULL) {
            fprintf (stderr, "%s: no definition of '%s'\n",
                    fname, config_table[i].name);
            ret = -1;
        }
    }
}

return ret;
}

```

### 7.5.3.2 int cb\_load\_std ( const char \* name )

Definition at line 106 of file config.c.

```

{
    return cb_load_conf (name, 1, 1);
}

```

## 7.5.4 Variable Documentation

### 7.5.4.1 const char\* name

Definition at line 56 of file config.c.

### 7.5.4.2 struct noreserve\* noresstab = NULL

Definition at line 52 of file config.c.



7.5.4.3 enum `cb_config_type` type

Definition at line 55 of file `config.c`.

7.5.4.4 `char* val`

Definition at line 58 of file `config.c`.

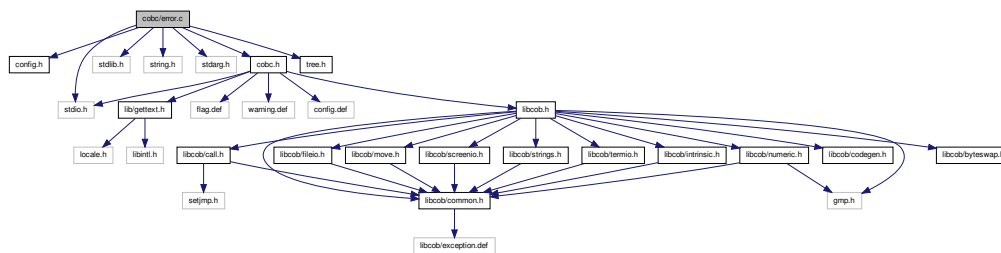
7.5.4.5 `void* var`

Definition at line 57 of file `config.c`.

7.6 `cobc/error.c` File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "cobc.h"
#include "tree.h"
```

Include dependency graph for `error.c`:



## Functions

- `char * check_filler_name` (`char *name`)
- `void cb_warning` (`const char *fmt,...`)
- `void cb_error` (`const char *fmt,...`)
- `void cb_warning_x` (`cb_tree x, const char *fmt,...`)
- `void cb_error_x` (`cb_tree x, const char *fmt,...`)

- int [cb\\_verify](#) (const enum [cb\\_support](#) tag, const char \*feature)
- void [redefinition\\_error](#) ([cb\\_tree](#) x)
- void [redefinition\\_warning](#) ([cb\\_tree](#) x, [cb\\_tree](#) y)
- void [undefined\\_error](#) ([cb\\_tree](#) x)
- void [ambiguous\\_error](#) ([cb\\_tree](#) x)
- void [group\\_error](#) ([cb\\_tree](#) x, const char \*clause)
- void [level\\_redundant\\_error](#) ([cb\\_tree](#) x, const char \*clause)
- void [level\\_require\\_error](#) ([cb\\_tree](#) x, const char \*clause)
- void [level\\_except\\_error](#) ([cb\\_tree](#) x, const char \*clause)

## 7.6.1 Function Documentation

### 7.6.1.1 void [ambiguous\\_error](#) ([cb\\_tree](#) x )

Definition at line 197 of file [error.c](#).

```

{
    struct cb_word *w;
    struct cb_field *p;
    struct cb_label *l2;
    cb_tree l;
    cb_tree y;

    w = CB_REFERENCE (x)->word;
    if (w->error == 0) {
        if (!errnamebuff) {
            errnamebuff = cobc_malloc (COB_NORMAL_BUFF);
        }
        /* display error on the first time */
        snprintf (errnamebuff, COB_NORMAL_MAX, "'%s'", CB_NAME (x));
        for (l = CB_REFERENCE (x)->chain; l; l = CB_REFERENCE (l)->chain)
        {
            strcat (errnamebuff, " in '");
            strcat (errnamebuff, CB_NAME (l));
            strcat (errnamebuff, "'");
        }
        cb_error_x (x, _("%s ambiguous; need qualification"), errnamebuff);
    }

    w->error = 1;

    /* display all fields with the same name */
    for (l = w->items; l; l = CB_CHAIN (l)) {
        y = CB_VALUE (l);
        snprintf (errnamebuff, COB_NORMAL_MAX, "'%s' ", w->name);

        switch (CB_TREE_TAG (y)) {
            case CB_TAG_FIELD:
                for (p = CB_FIELD (y)->parent; p; p = p->parent)
                {
                    strcat (errnamebuff, "in '");
                    strcat (errnamebuff, p->name);
                    strcat (errnamebuff, "' ");
                }
                break;
            case CB_TAG_LABEL:
                l2 = CB_LABEL (y);

```

```

        if (l2->section) {
            strcat (errnamebuff, "in '");
            strcat (errnamebuff, (const char *) (l2->
section->name));
            strcat (errnamebuff, "' ");
        }
        break;
default:
    break;
    }
    strcat (errnamebuff, _("defined here"));
    cb_error_x (y, errnamebuff);
    }
}

```

#### 7.6.1.2 void `cb_error` ( const char \* *fmt*, ... )

Definition at line 85 of file error.c.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error (NULL, 0, "Error: ", fmt, ap);
    va_end (ap);

    errorcount++;
}

```

#### 7.6.1.3 void `cb_error_x` ( `cb_tree` *x*, const char \* *fmt*, ... )

Definition at line 110 of file error.c.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error ((char *) (x->source_file), x->source_line, "Error: ", fmt, ap
);
    va_end (ap);

    errorcount++;
}

```

#### 7.6.1.4 int `cb_verify` ( const enum `cb_support` *tag*, const char \* *feature* )

Definition at line 122 of file error.c.

```

{
    switch (tag) {

```

```

    case CB_OK:
        return 1;
    case CB_WARNING:
        return 1;
    case CB_ARCHAIC:
        if (cb_warn_archaic) {
            cb_warning (_("%s is archaic in %s"), feature, cb_config_
name);
        }
        return 1;
    case CB_OBSOLETE:
        if (cb_warn_obsolete) {
            cb_warning (_("%s is obsolete in %s"), feature, cb_config
_name);
        }
        return 1;
    case CB_SKIP:
        return 0;
    case CB_IGNORE:
        cb_warning (_("%s ignored"), feature);
        return 0;
    case CB_ERROR:
        return 0;
    case CB_UNCONFORMABLE:
        cb_error (_("%s does not conform to %s"), feature, cb_config_name
);
        return 0;
    }
    return 0;
}

```

#### 7.6.1.5 void cb\_warning ( const char \* *fmt*, ... )

Definition at line 73 of file error.c.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error (NULL, 0, "Warning: ", fmt, ap);
    va_end (ap);

    warningcount++;
}

```

#### 7.6.1.6 void cb\_warning\_x ( cb\_tree x, const char \* *fmt*, ... )

Definition at line 98 of file error.c.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error ((char *) (x->source_file), x->source_line, "Warning: ", fmt,
ap);
}

```

```
        va_end (ap);

        warningcount++;
    }
}
```

#### 7.6.1.7 char\* check\_filler\_name ( char \* name )

Definition at line 64 of file error.c.

```
{
    if (!memcmp (name, "WORK$", 5)) {
        name = (char *)"FILLER";
    }
    return name;
}
```

#### 7.6.1.8 void group\_error ( cb\_tree x, const char \* clause )

Definition at line 250 of file error.c.

```
{
    cb_error_x (x, _("Group item '%s' cannot have %s clause"),
        check_filler_name (cb_name (x)), clause);
}
```

#### 7.6.1.9 void level\_except\_error ( cb\_tree x, const char \* clause )

Definition at line 270 of file error.c.

```
{
    cb_error_x (x, _("Level %02d item '%s' cannot have other than %s clause")
        ,
        cb_field (x)->level, check_filler_name (cb_name (x)), clause)
    ;
}
```

#### 7.6.1.10 void level\_redundant\_error ( cb\_tree x, const char \* clause )

Definition at line 256 of file error.c.

```
{
    cb_error_x (x, _("Level %02d item '%s' cannot have %s clause"),
        cb_field (x)->level, check_filler_name (cb_name (x)), clause)
    ;
}
```

**7.6.1.11 void level\_require\_error ( cb\_tree x, const char \* clause )**

Definition at line 263 of file error.c.

```

{
    cb_error_x (x, _("Level %02d item '%s' requires %s clause"),
               cb_field (x)->level, check_filler_name (cb_name (x)), clause)
;
}

```

**7.6.1.12 void redefinition\_error ( cb\_tree x )**

Definition at line 154 of file error.c.

```

{
    struct cb_word *w;

    w = CB_REFERENCE (x)->word;
    cb_error_x (x, _("Redefinition of '%s'"), w->name);
    cb_error_x (CB_VALUE (w->items), _("%s' previously defined here"), w->
name);
}

```

**7.6.1.13 void redefinition\_warning ( cb\_tree x, cb\_tree y )**

Definition at line 164 of file error.c.

```

{
    struct cb_word *w;

    w = CB_REFERENCE (x)->word;
    cb_warning_x (x, _("Redefinition of '%s'"), w->name);
    if (y) {
        cb_warning_x (y, _("%s' previously defined here"), w->name);
    } else {
        cb_warning_x (CB_VALUE (w->items), _("%s' previously defined her
e"), w->name);
    }
}

```

**7.6.1.14 void undefined\_error ( cb\_tree x )**

Definition at line 178 of file error.c.

```

{
    struct cb_reference *r;
    cb_tree c;

    if (!errnamebuff) {
        errnamebuff = cobc_malloc (COB_NORMAL_BUFF);
    }
}

```

```

    }
    r = CB_REFERENCE (x);
    snprintf (errnamebuff, COB_NORMAL_MAX, "%s'", CB_NAME (x));
    for (c = r->chain; c; c = CB_REFERENCE (c)->chain) {
        strcat (errnamebuff, " in ");
        strcat (errnamebuff, CB_NAME (c));
        strcat (errnamebuff, "'");
    }
    cb_error_x (x, _("%s undefined"), errnamebuff);
}

```

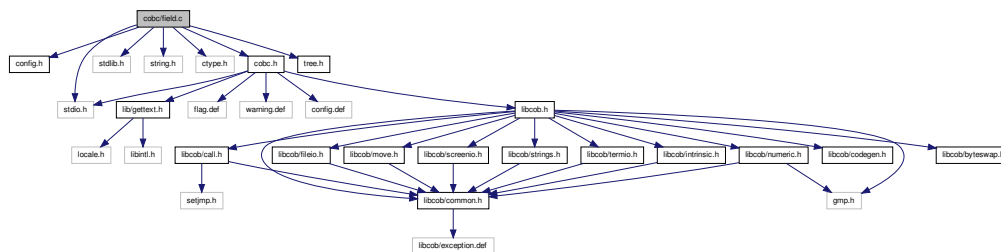
## 7.7 cobc/field.c File Reference

```

#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobc.h"
#include "tree.h"

```

Include dependency graph for field.c:



## Functions

- `int cb_get_level (cb_tree x)`
- `cb_tree cb_build_field_tree (cb_tree level, cb_tree name, struct cb_field *last_field, enum cb_storage storage, struct cb_file *fn)`
- `struct cb_field * cb_resolve_redefines (struct cb_field *field, cb_tree redefines)`
- `void cb_validate_field (struct cb_field *f)`
- `void cb_validate_88_item (struct cb_field *f)`
- `struct cb_field * cb_validate_78_item (struct cb_field *f)`
- `void cb_clear_real_field (void)`

## Variables

- `size_t cb_needs_01 = 0`

### 7.7.1 Function Documentation

7.7.1.1 `cb_tree cb_build_field_tree ( cb_tree level, cb_tree name, struct cb_field * last_field, enum cb_storage storage, struct cb_file * fn )`

Definition at line 78 of file field.c.

```
{
    struct cb_reference    *r;
    struct cb_field        *f;
    struct cb_field        *p;
    struct cb_field        *field_fill;
    cb_tree                dummy_fill;
    cb_tree                l;
    cb_tree                x;
    int                    lv;

    if (level == cb_error_node || name == cb_error_node) {
        return cb_error_node;
    }

    /* check the level number */
    lv = cb_get_level (level);
    if (!lv) {
        return cb_error_node;
    }

    /* build the field */
    r = CB_REFERENCE (name);
    f = CB_FIELD (cb_build_field (name));
    f->storage = storage;
    last_real_field = last_field;
    if (lv == 78) {
        f->level = 01;
        f->flag_item_78 = 1;
        return CB_TREE (f);
    } else {
        f->level = lv;
    }
    if (f->level == 01 && storage == CB_STORAGE_FILE) {
        if (fn->external) {
            f->flag_external = 1;
            has_external = 1;
        } else if (fn->global) {
            f->flag_is_global = 1;
        }
    }
    if (last_field) {
        if (last_field->level == 77 && f->level != 01 &&
            f->level != 77 && f->level != 66 && f->level != 88) {
            cb_error_x (name, _("Level number must begin with 01 or 7
7"));
            return cb_error_node;
        }
    }
}
```



```

/* checks for redefinition */
if (cb_warn_redefinition) {
    if (r->word->count > 1) {
        if (f->level == 01 || f->level == 77) {
            redefinition_warning (name, NULL);
        } else {
            for (l = r->word->items; l; l = CB_CHAIN (l)) {
                x = CB_VALUE (l);
                if (!CB_FIELD_P (x)
                    || CB_FIELD (x)->level == 01
                    || CB_FIELD (x)->level == 77
                    || (f->level == last_field->level
                        && CB_FIELD (x)->parent == last_f
ield->parent)) {
                    redefinition_warning (name, x);
                    break;
                }
            }
        }
    }
}

if (last_field && last_field->level == 88) {
    last_field = last_field->parent;
}

/* link the field into the tree */
if (f->level == 01 || f->level == 77) {
    /* top level */
    cb_needs_01 = 0;
    if (last_field) {
/*
        cb_field_add (cb_field_founder (last_field), f);
*/
        cb_field_founder (last_field)->sister = f;
    }
} else if (!last_field || cb_needs_01) {
    /* invalid top level */
    cb_error_x (name, _("Level number must begin with 01 or 77"));
    return cb_error_node;
} else if (f->level == 66) {
    /* level 66 */
    f->parent = cb_field_founder (last_field);
    for (p = f->parent->children; p && p->sister; p = p->sister) ;
    if (p) {
        p->sister = f;
    }
} else if (f->level == 88) {
    /* level 88 */
    f->parent = last_field;
} else if (f->level > last_field->level) {
    /* lower level */
    last_field->children = f;
    f->parent = last_field;
} else if (f->level == last_field->level) {
    /* same level */
same_level:
    last_field->sister = f;
    f->parent = last_field->parent;
} else {
    /* upper level */

```

```

for (p = last_field->parent; p; p = p->parent) {
    if (p->level == f->level) {
        last_field = p;
        goto same_level;
    }
    if (cb_relax_level_hierarchy && p->level < f->level) {
        break;
    }
}
if (cb_relax_level_hierarchy) {
    dummy_fill = cb_build_filler ();
    field_fill = CB_FIELD (cb_build_field (dummy_fill));
    cb_warning_x (name, _("No previous data item of level %02
d"), f->level);
    field_fill->level = f->level;
    field_fill->storage = storage;
    field_fill->children = p->children;
    field_fill->parent = p;
    for (p = p->children; p != NULL; p = p->sister) {
        p->parent = field_fill;
    }
    field_fill->parent->children = field_fill;
    field_fill->sister = f;
    f->parent = field_fill->parent;
    last_field = field_fill;
} else {
    cb_error_x (name, _("No previous data item of level %02d"
), f->level);
    return cb_error_node;
}
}

/* inherit parent's properties */
if (f->parent) {
    f->usage = f->parent->usage;
    f->indexes = f->parent->indexes;
    f->flag_sign_leading = f->parent->flag_sign_leading;
    f->flag_sign_separate = f->parent->flag_sign_separate;
    f->flag_is_global = f->parent->flag_is_global;
}
return CB_TREE (f);
}

```

### 7.7.1.2 void cb\_clear\_real\_field ( void )

Definition at line 1050 of file field.c.

```

{
    last_real_field = NULL;
}

```

### 7.7.1.3 int cb\_get\_level ( cb\_tree x )

Definition at line 41 of file field.c.

```

{

```

```

const char    *p;
const char    *name;
int           level = 0;

name = CB_NAME (x);
/* get level */
for (p = name; *p; p++) {
    if (!isdigit (*p)) {
        goto level_error;
    }
    level = level * 10 + (*p - '0');
}

/* check level */
switch (level) {
case 66:
case 77:
case 78:
case 88:
    break;
default:
    if (level < 1 || level > 49) {
        goto level_error;
    }
    break;
}

return level;

level_error:
    cb_error_x (x, _("Invalid level number '%s'"), name);
    return 0;
}

```

#### 7.7.1.4 struct `cb_field*` `cb_resolve_redefines` ( struct `cb_field` \* *field*, *cb\_tree redefines* ) [read]

Definition at line 231 of file field.c.

```

{
    struct cb_field    *f;
    struct cb_reference *r;
    const char        *name;
    cb_tree           x;

    r = CB_REFERENCE (redefines);
    name = CB_NAME (redefines);
    x = CB_TREE (field);

    /* check qualification */
    if (r->chain) {
        cb_error_x (x, _("%s' cannot be qualified here"), name);
        return NULL;
    }

    /* check subscripts */
    if (r->subs) {
        cb_error_x (x, _("%s' cannot be subscripted here"), name);
        return NULL;
    }
}

```

```

    }

    /* resolve the name in the current group (if any) */
    if (field->parent && field->parent->children) {
        for (f = field->parent->children; f; f = f->sister) {
            if (strcasemp (f->name, name) == 0) {
                break;
            }
        }
        if (f == NULL) {
            cb_error_x (x, _("%s' undefined in '%s'"), name, field->
parent->name);
            return NULL;
        }
    } else {
        if (cb_ref (redefines) == cb_error_node) {
            return NULL;
        }
        f = cb_field (redefines);
    }

    /* check level number */
    if (f->level != field->level) {
        cb_error_x (x, _("Level number of REDEFINES entries must be ident
ical"));
        return NULL;
    }
    if (f->level == 66 || f->level == 88) {
        cb_error_x (x, _("Level number of REDEFINES entry cannot be 66 or
88"));
        return NULL;
    }

    if (!cb_indirect_redefines && f->redefines) {
        cb_error_x (x, _("%s' not the original definition"), f->name);
        return NULL;
    }

    /* return the original definition */
    while (f->redefines) {
        f = f->redefines;
    }
    return f;
}

```

#### 7.7.1.5 struct `cb_field*` `cb.validate.78.item ( struct cb_field * f )` [read]

Definition at line 1033 of file `field.c`.

```

{
    cb_tree x;

    x = CB_TREE (f);
    if (!f->values) {
        level_require_error (x, "VALUE");
    }

    if (f->pic || f->flag_occurs) {
        level_except_error (x, "VALUE");
    }
}

```

```
    }
    cb_add_78 (f);
    return last_real_field;
}
```

#### 7.7.1.6 void cb\_validate\_88\_item ( struct cb\_field \* f )

Definition at line 1018 of file field.c.

```
{
    cb_tree x;

    x = CB_TREE (f);
    if (!f->values) {
        level_require_error (x, "VALUE");
    }

    if (f->pic || f->flag_occurs) {
        level_except_error (x, "VALUE");
    }
}
```

#### 7.7.1.7 void cb\_validate\_field ( struct cb\_field \* f )

Definition at line 973 of file field.c.

```
{
    struct cb_field      *c;

    if (validate_field_1 (f) != 0) {
        f->flag_invalid = 1;
        return;
    }
    /* RXW - Remove */
    if (f->flag_item_78) {
        f->flag_is_verified = 1;
        return;
    }

    /* setup parameters */
    if (f->storage == CB_STORAGE_LOCAL ||
        f->storage == CB_STORAGE_LINKAGE ||
        f->flag_item_based) {
        f->flag_local = 1;
    }
    if (f->storage == CB_STORAGE_LINKAGE || f->flag_item_based) {
        f->flag_base = 1;
    }
    setup_parameters (f);

    /* compute size */
    compute_size (f);
    if (!f->redefines) {
        f->memory_size = f->size * f->occurs_max;
    } else if (f->redefines->memory_size < f->size * f->occurs_max) {
```

```

        f->redefines->memory_size = f->size * f->occurs_max;
    }

    validate_field_value (f);
    if (f->flag_is_global) {
        f->count++;
        for (c = f->children; c; c = c->sister) {
            c->flag_is_global = 1;
            c->count++;
        }
    }
    f->flag_is_verified = 1;
}

```

## 7.7.2 Variable Documentation

### 7.7.2.1 size\_t cb\_needs\_01 = 0

Definition at line 33 of file field.c.

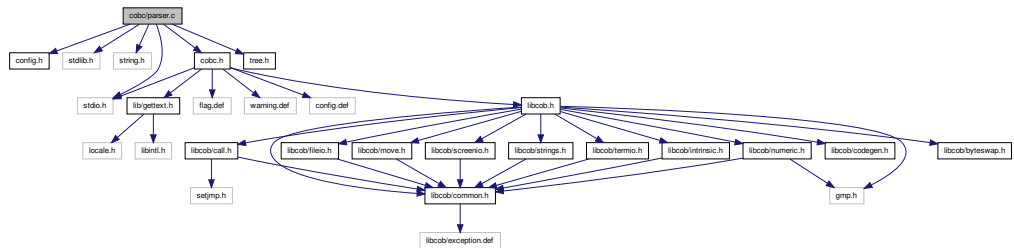
## 7.8 cobc/parser.c File Reference

```

#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "cobc.h"
#include "tree.h"
#include <stdio.h>

```

Include dependency graph for parser.c:



## Classes

- union [yyalloc](#)

## Defines

- #define [YYBISON](#) 1
- #define [YYSKELETON\\_NAME](#) "yacc.c"
- #define [YYPURE](#) 0
- #define [YYLSP\\_NEEDED](#) 0
- #define [YYTOKENTYPE](#)
- #define [TOKEN\\_EOF](#) 0
- #define [ACCEPT](#) 258
- #define [ACCESS](#) 259
- #define [ADD](#) 260
- #define [ADDRESS](#) 261
- #define [ADVANCING](#) 262
- #define [AFTER](#) 263
- #define [ALL](#) 264
- #define [ALLOCATE](#) 265
- #define [ALPHABET](#) 266
- #define [ALPHABETIC](#) 267
- #define [ALPHABETIC\\_LOWER](#) 268
- #define [ALPHABETIC\\_UPPER](#) 269
- #define [ALPHANUMERIC](#) 270
- #define [ALPHANUMERIC\\_EDITED](#) 271
- #define [ALSO](#) 272
- #define [ALTER](#) 273
- #define [ALTERNATE](#) 274
- #define [AND](#) 275
- #define [ANY](#) 276
- #define [ARE](#) 277
- #define [AREA](#) 278
- #define [ARGUMENT\\_NUMBER](#) 279
- #define [ARGUMENT\\_VALUE](#) 280
- #define [AS](#) 281
- #define [ASCENDING](#) 282
- #define [ASSIGN](#) 283
- #define [AT](#) 284
- #define [AUTO](#) 285
- #define [AUTOMATIC](#) 286
- #define [BACKGROUND\\_COLOR](#) 287
- #define [BASED](#) 288
- #define [BEFORE](#) 289
- #define [BELL](#) 290
- #define [BINARY](#) 291
- #define [BINARY\\_C\\_LONG](#) 292
- #define [BINARY\\_CHAR](#) 293
- #define [BINARY\\_DOUBLE](#) 294
- #define [BINARY\\_LONG](#) 295
- #define [BINARY\\_SHORT](#) 296

- #define [BLANK](#) 297
- #define [BLANK\\_LINE](#) 298
- #define [BLANK\\_SCREEN](#) 299
- #define [BLINK](#) 300
- #define [BLOCK](#) 301
- #define [BOTTOM](#) 302
- #define [BY](#) 303
- #define [BYTE\\_LENGTH](#) 304
- #define [CALL](#) 305
- #define [CANCEL](#) 306
- #define [CH](#) 307
- #define [CHAINING](#) 308
- #define [CHARACTER](#) 309
- #define [CHARACTERS](#) 310
- #define [CLASS](#) 311
- #define [CLOSE](#) 312
- #define [CODE](#) 313
- #define [CODE\\_SET](#) 314
- #define [COLLATING](#) 315
- #define [COL](#) 316
- #define [COLS](#) 317
- #define [COLUMN](#) 318
- #define [COLUMNS](#) 319
- #define [COMMA](#) 320
- #define [COMMAND\\_LINE](#) 321
- #define [COMMA\\_DELIM](#) 322
- #define [COMMIT](#) 323
- #define [COMMON](#) 324
- #define [COMP](#) 325
- #define [COMPUTE](#) 326
- #define [COMP\\_1](#) 327
- #define [COMP\\_2](#) 328
- #define [COMP\\_3](#) 329
- #define [COMP\\_4](#) 330
- #define [COMP\\_5](#) 331
- #define [COMP\\_X](#) 332
- #define [CONCATENATE\\_FUNC](#) 333
- #define [CONFIGURATION](#) 334
- #define [CONSTANT](#) 335
- #define [CONTAINS](#) 336
- #define [CONTENT](#) 337
- #define [CONTINUE](#) 338
- #define [CONTROL](#) 339
- #define [CONTROLS](#) 340
- #define [CONTROL\\_FOOTING](#) 341
- #define [CONTROL\\_HEADING](#) 342



- #define [CONVERTING](#) 343
- #define [CORRESPONDING](#) 344
- #define [COUNT](#) 345
- #define [CRT](#) 346
- #define [CURRENCY](#) 347
- #define [CURRENT\\_DATE\\_FUNC](#) 348
- #define [CURSOR](#) 349
- #define [CYCLE](#) 350
- #define [DATA](#) 351
- #define [DATE](#) 352
- #define [DAY](#) 353
- #define [DAY\\_OF\\_WEEK](#) 354
- #define [DE](#) 355
- #define [DEBUGGING](#) 356
- #define [DECIMAL\\_POINT](#) 357
- #define [DECLARATIVES](#) 358
- #define [DEFAULT](#) 359
- #define [DELETE](#) 360
- #define [DELIMITED](#) 361
- #define [DELIMITER](#) 362
- #define [DEPENDING](#) 363
- #define [DESCENDING](#) 364
- #define [DETAIL](#) 365
- #define [DISK](#) 366
- #define [DISPLAY](#) 367
- #define [DIVIDE](#) 368
- #define [DIVISION](#) 369
- #define [DOWN](#) 370
- #define [DUPLICATES](#) 371
- #define [DYNAMIC](#) 372
- #define [EBCDIC](#) 373
- #define [ELSE](#) 374
- #define [END](#) 375
- #define [END\\_ACCEPT](#) 376
- #define [END\\_ADD](#) 377
- #define [END\\_CALL](#) 378
- #define [END\\_COMPUTE](#) 379
- #define [END\\_DELETE](#) 380
- #define [END\\_DISPLAY](#) 381
- #define [END\\_DIVIDE](#) 382
- #define [END\\_EVALUATE](#) 383
- #define [END\\_FUNCTION](#) 384
- #define [END\\_IF](#) 385
- #define [END\\_MULTIPLY](#) 386
- #define [END\\_PERFORM](#) 387
- #define [END\\_PROGRAM](#) 388

- #define [END\\_READ](#) 389
- #define [END\\_RETURN](#) 390
- #define [END\\_REWRITE](#) 391
- #define [END\\_SEARCH](#) 392
- #define [END\\_START](#) 393
- #define [END\\_STRING](#) 394
- #define [END\\_SUBTRACT](#) 395
- #define [END\\_UNSTRING](#) 396
- #define [END\\_WRITE](#) 397
- #define [ENTRY](#) 398
- #define [ENVIRONMENT](#) 399
- #define [ENVIRONMENT\\_NAME](#) 400
- #define [ENVIRONMENT\\_VALUE](#) 401
- #define [EOL](#) 402
- #define [EOP](#) 403
- #define [EOS](#) 404
- #define [EQUAL](#) 405
- #define [EQUALS](#) 406
- #define [ERASE](#) 407
- #define [ERROR](#) 408
- #define [ESCAPE](#) 409
- #define [EVALUATE](#) 410
- #define [EVENT\\_STATUS](#) 411
- #define [EXCEPTION](#) 412
- #define [EXCLUSIVE](#) 413
- #define [EXIT](#) 414
- #define [EXTEND](#) 415
- #define [EXTERNAL](#) 416
- #define [FD](#) 417
- #define [FILE\\_CONTROL](#) 418
- #define [FILE\\_ID](#) 419
- #define [FILLER](#) 420
- #define [FINAL](#) 421
- #define [FIRST](#) 422
- #define [FOOTING](#) 423
- #define [FOR](#) 424
- #define [FOREGROUND\\_COLOR](#) 425
- #define [FOREVER](#) 426
- #define [FREE](#) 427
- #define [FROM](#) 428
- #define [FULL](#) 429
- #define [FUNCTION](#) 430
- #define [FUNCTION\\_ID](#) 431
- #define [FUNCTION\\_NAME](#) 432
- #define [GE](#) 433
- #define [GENERATE](#) 434

- #define [GIVING](#) 435
- #define [GLOBAL](#) 436
- #define [GO](#) 437
- #define [GOBACK](#) 438
- #define [GREATER](#) 439
- #define [GROUP](#) 440
- #define [HEADING](#) 441
- #define [HIGHLIGHT](#) 442
- #define [HIGH\\_VALUE](#) 443
- #define [IDENTIFICATION](#) 444
- #define [IF](#) 445
- #define [IGNORE](#) 446
- #define [IGNORING](#) 447
- #define [IN](#) 448
- #define [INDEX](#) 449
- #define [INDEXED](#) 450
- #define [INDICATE](#) 451
- #define [INITIALIZE](#) 452
- #define [INITIALIZED](#) 453
- #define [INITIATE](#) 454
- #define [INPUT](#) 455
- #define [INPUT\\_OUTPUT](#) 456
- #define [INSPECT](#) 457
- #define [INTO](#) 458
- #define [INTRINSIC](#) 459
- #define [INVALID](#) 460
- #define [INVALID\\_KEY](#) 461
- #define [IS](#) 462
- #define [I\\_O](#) 463
- #define [I\\_O\\_CONTROL](#) 464
- #define [JUSTIFIED](#) 465
- #define [KEY](#) 466
- #define [LABEL](#) 467
- #define [LAST](#) 468
- #define [LAST\\_DETAIL](#) 469
- #define [LE](#) 470
- #define [LEADING](#) 471
- #define [LEFT](#) 472
- #define [LENGTH](#) 473
- #define [LESS](#) 474
- #define [LIMIT](#) 475
- #define [LIMITS](#) 476
- #define [LINAGE](#) 477
- #define [LINAGE\\_COUNTER](#) 478
- #define [LINE](#) 479
- #define [LINES](#) 480

- #define [LINKAGE](#) 481
- #define [LITERAL](#) 482
- #define [LOCALE](#) 483
- #define [LOCALE\\_DT\\_FUNC](#) 484
- #define [LOCAL\\_STORAGE](#) 485
- #define [LOCK](#) 486
- #define [LOWER\\_CASE\\_FUNC](#) 487
- #define [LOWLIGHT](#) 488
- #define [LOW\\_VALUE](#) 489
- #define [MANUAL](#) 490
- #define [MEMORY](#) 491
- #define [MERGE](#) 492
- #define [MINUS](#) 493
- #define [MNEMONIC\\_NAME](#) 494
- #define [MODE](#) 495
- #define [MOVE](#) 496
- #define [MULTIPLE](#) 497
- #define [MULTIPLY](#) 498
- #define [NATIONAL](#) 499
- #define [NATIONAL\\_EDITED](#) 500
- #define [NATIVE](#) 501
- #define [NE](#) 502
- #define [NEGATIVE](#) 503
- #define [NEXT](#) 504
- #define [NEXT\\_SENTENCE](#) 505
- #define [NO](#) 506
- #define [NOT](#) 507
- #define [NOT\\_END](#) 508
- #define [NOT\\_EOP](#) 509
- #define [NOT\\_EXCEPTION](#) 510
- #define [NOT\\_INVALID\\_KEY](#) 511
- #define [NOT\\_OVERFLOW](#) 512
- #define [NOT\\_SIZE\\_ERROR](#) 513
- #define [NO\\_ADVANCING](#) 514
- #define [NUMBER](#) 515
- #define [NUMBERS](#) 516
- #define [NUMERIC](#) 517
- #define [NUMERIC\\_EDITED](#) 518
- #define [NUMVALC\\_FUNC](#) 519
- #define [OBJECT\\_COMPUTER](#) 520
- #define [OCCURS](#) 521
- #define [OF](#) 522
- #define [OFF](#) 523
- #define [OMITTED](#) 524
- #define [ON](#) 525
- #define [ONLY](#) 526

- #define [OPEN](#) 527
- #define [OPTIONAL](#) 528
- #define [OR](#) 529
- #define [ORDER](#) 530
- #define [ORGANIZATION](#) 531
- #define [OTHER](#) 532
- #define [OUTPUT](#) 533
- #define [OVERFLOW](#) 534
- #define [OVERLINE](#) 535
- #define [PACKED\\_DECIMAL](#) 536
- #define [PADDING](#) 537
- #define [PAGE](#) 538
- #define [PAGE\\_FOOTING](#) 539
- #define [PAGE\\_HEADING](#) 540
- #define [PARAGRAPH](#) 541
- #define [PERFORM](#) 542
- #define [PICTURE](#) 543
- #define [PLUS](#) 544
- #define [POINTER](#) 545
- #define [POSITION](#) 546
- #define [POSITIVE](#) 547
- #define [PRESENT](#) 548
- #define [PREVIOUS](#) 549
- #define [PRINTER](#) 550
- #define [PRINTING](#) 551
- #define [PROCEDURE](#) 552
- #define [PROCEDURES](#) 553
- #define [PROCEED](#) 554
- #define [PROGRAM](#) 555
- #define [PROGRAM\\_ID](#) 556
- #define [PROGRAM\\_NAME](#) 557
- #define [PROGRAM\\_POINTER](#) 558
- #define [PROMPT](#) 559
- #define [QUOTE](#) 560
- #define [RANDOM](#) 561
- #define [RD](#) 562
- #define [READ](#) 563
- #define [RECORD](#) 564
- #define [RECORDING](#) 565
- #define [RECORDS](#) 566
- #define [RECURSIVE](#) 567
- #define [REDEFINES](#) 568
- #define [REEL](#) 569
- #define [REFERENCE](#) 570
- #define [RELATIVE](#) 571
- #define [RELEASE](#) 572

- #define [REMAINDER](#) 573
- #define [REMOVAL](#) 574
- #define [RENAMES](#) 575
- #define [REPLACING](#) 576
- #define [REPORT](#) 577
- #define [REPORTING](#) 578
- #define [REPORTS](#) 579
- #define [REPORT\\_FOOTING](#) 580
- #define [REPORT\\_HEADING](#) 581
- #define [REPOSITORY](#) 582
- #define [REQUIRED](#) 583
- #define [RESERVE](#) 584
- #define [RETURN](#) 585
- #define [RETURNING](#) 586
- #define [REVERSE\\_FUNC](#) 587
- #define [REVERSE\\_VIDEO](#) 588
- #define [REWIND](#) 589
- #define [REWRITE](#) 590
- #define [RIGHT](#) 591
- #define [ROLLBACK](#) 592
- #define [ROUNDED](#) 593
- #define [RUN](#) 594
- #define [SAME](#) 595
- #define [SCREEN](#) 596
- #define [SCREEN\\_CONTROL](#) 597
- #define [SCROLL](#) 598
- #define [SD](#) 599
- #define [SEARCH](#) 600
- #define [SECTION](#) 601
- #define [SECURE](#) 602
- #define [SEGMENT\\_LIMIT](#) 603
- #define [SELECT](#) 604
- #define [SEMI\\_COLON](#) 605
- #define [SENTENCE](#) 606
- #define [SEPARATE](#) 607
- #define [SEQUENCE](#) 608
- #define [SEQUENTIAL](#) 609
- #define [SET](#) 610
- #define [SHARING](#) 611
- #define [SIGN](#) 612
- #define [SIGNED](#) 613
- #define [SIGNED\\_INT](#) 614
- #define [SIGNED\\_LONG](#) 615
- #define [SIGNED\\_SHORT](#) 616
- #define [SIZE](#) 617
- #define [SIZE\\_ERROR](#) 618

- #define [SORT](#) 619
- #define [SORT\\_MERGE](#) 620
- #define [SOURCE](#) 621
- #define [SOURCE\\_COMPUTER](#) 622
- #define [SPACE](#) 623
- #define [SPECIAL\\_NAMES](#) 624
- #define [STANDARD](#) 625
- #define [STANDARD\\_1](#) 626
- #define [STANDARD\\_2](#) 627
- #define [START](#) 628
- #define [STATUS](#) 629
- #define [STOP](#) 630
- #define [STRING](#) 631
- #define [SUBSTITUTE\\_FUNC](#) 632
- #define [SUBSTITUTE\\_CASE\\_FUNC](#) 633
- #define [SUBTRACT](#) 634
- #define [SUM](#) 635
- #define [SUPPRESS](#) 636
- #define [SYMBOLIC](#) 637
- #define [SYNCHRONIZED](#) 638
- #define [TALLYING](#) 639
- #define [TAPE](#) 640
- #define [TERMINATE](#) 641
- #define [TEST](#) 642
- #define [THAN](#) 643
- #define [THEN](#) 644
- #define [THRU](#) 645
- #define [TIME](#) 646
- #define [TIMES](#) 647
- #define [TO](#) 648
- #define [TOK\\_FALSE](#) 649
- #define [TOK\\_FILE](#) 650
- #define [TOK\\_INITIAL](#) 651
- #define [TOK\\_NULL](#) 652
- #define [TOK\\_TRUE](#) 653
- #define [TOP](#) 654
- #define [TRAILING](#) 655
- #define [TRANSFORM](#) 656
- #define [TRIM\\_FUNCTION](#) 657
- #define [TYPE](#) 658
- #define [UNDERLINE](#) 659
- #define [UNIT](#) 660
- #define [UNLOCK](#) 661
- #define [UNSIGNED](#) 662
- #define [UNSIGNED\\_INT](#) 663
- #define [UNSIGNED\\_LONG](#) 664

- #define [UNSIGNED\\_SHORT](#) 665
- #define [UNSTRING](#) 666
- #define [UNTIL](#) 667
- #define [UP](#) 668
- #define [UPDATE](#) 669
- #define [UPON](#) 670
- #define [UPON\\_ARGUMENT\\_NUMBER](#) 671
- #define [UPON\\_COMMAND\\_LINE](#) 672
- #define [UPON\\_ENVIRONMENT\\_NAME](#) 673
- #define [UPON\\_ENVIRONMENT\\_VALUE](#) 674
- #define [UPPER\\_CASE\\_FUNC](#) 675
- #define [USAGE](#) 676
- #define [USE](#) 677
- #define [USING](#) 678
- #define [VALUE](#) 679
- #define [VARYING](#) 680
- #define [WAIT](#) 681
- #define [WHEN](#) 682
- #define [WHEN\\_COMPILED\\_FUNC](#) 683
- #define [WITH](#) 684
- #define [WORD](#) 685
- #define [WORDS](#) 686
- #define [WORKING\\_STORAGE](#) 687
- #define [WRITE](#) 688
- #define [YYYYDDD](#) 689
- #define [YYYYMMDD](#) 690
- #define [ZERO](#) 691
- #define [UNARY\\_SIGN](#) 692
- #define [yyerror](#) cb\_error
- #define [YYDEBUG](#) 1
- #define [YERROR\\_VERBOSE](#) 1
- #define [PENDING](#)(x) cb\_warning (\_,("%s' not implemented"), x)
- #define [emit\\_statement](#)(x) [current\\_program->exec\\_list](#) = cb\_cons (x, [current\\_program->exec\\_list](#))
- #define [push\\_expr](#)(type, node) [current\\_expr](#) = cb\_build\_list (cb\_int (type), node, [current\\_expr](#))
- #define [TERM\\_NONE](#) 0
- #define [TERM\\_ACCEPT](#) 1
- #define [TERM\\_ADD](#) 2
- #define [TERM\\_CALL](#) 3
- #define [TERM\\_COMPUTE](#) 4
- #define [TERM\\_DELETE](#) 5
- #define [TERM\\_DISPLAY](#) 6
- #define [TERM\\_DIVIDE](#) 7
- #define [TERM\\_EVALUATE](#) 8
- #define [TERM\\_IF](#) 9



- #define `TERM_MULTIPLY` 10
- #define `TERM_PERFORM` 11
- #define `TERM_READ` 12
- #define `TERM_RECEIVE` 13
- #define `TERM_RETURN` 14
- #define `TERM_REWRITE` 15
- #define `TERM_SEARCH` 16
- #define `TERM_START` 17
- #define `TERM_STRING` 18
- #define `TERM_SUBTRACT` 19
- #define `TERM_UNSTRING` 20
- #define `TERM_WRITE` 21
- #define `TERM_MAX` 22
- #define `YYERROR_VERBOSE` 1
- #define `YYSTACK_ALLOC` malloc
- #define `YYSTACK_FREE` free
- #define `YYSTACK_GAP_MAXIMUM` (sizeof (union `yyalloc`) - 1)
- #define `YYSTACK_BYTES`(N)
- #define `YYCOPY`(To, From, Count)
- #define `YYSTACK_RELOCATE`(Stack)
- #define `YYFINAL` 3
- #define `YYLAST` 5462
- #define `YYNTOKENS` 451
- #define `YYNNTS` 690
- #define `YYNRULES` 1518
- #define `YYNSTATES` 2240
- #define `YYUNDEFTOK` 2
- #define `YYMAXUTOK` 692
- #define `YYTRANSLATE`(YYX) ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)
- #define `YYPACT_NINF` -1922
- #define `YYTABLE_NINF` -1518
- #define `YYSIZE_T` unsigned int
- #define `yyerrok` (yyerrstatus = 0)
- #define `yyclearin` (yychar = YYEMPTY)
- #define `YYEMPTY` (-2)
- #define `YYEOF` 0
- #define `YYACCEPT` goto yyacceptlab
- #define `YYABORT` goto yyabortlab
- #define `YYERROR` goto yyerrlab1
- #define `YYFAIL` goto yyerrlab
- #define `YYRECOVERING`() (!yyerrstatus)
- #define `YYBACKUP`(Token, Value)
- #define `YYTERROR` 1
- #define `YYERRCODE` 256
- #define `YYLLOC_DEFAULT`(Current, Rhs, N)

- #define `YLEX` `yylex` ()
- #define `YFPRINTF` `fprintf`
- #define `YDPRINTF`(Args)
- #define `YDSYMPRINT`(Args)
- #define `YDSYMPRINTF`(Title, Token, Value, Location)
- #define `Y_STACK_PRINT`(Bottom, Top)
- #define `Y_REDUCE_PRINT`(Rule)
- #define `YINITDEPTH` 200
- #define `YMAXDEPTH` 10000
- #define `YPOPSTACK` (`yyvsp--`, `yyssp--`)

### Typedefs

- typedef short `yysigned_char`

### Enumerations

- enum `y tokentype` {
  - `TOKEN_EOF` = 0, `ACCEPT` = 258, `ACCESS` = 259, `ADD` = 260,
  - `ADDRESS` = 261, `ADVANCING` = 262, `AFTER` = 263, `ALL` = 264,
  - `ALLOCATE` = 265, `ALPHABET` = 266, `ALPHABETIC` = 267, `ALPHABETIC_`  
`LOWER` = 268,
  - `ALPHABETIC_UPPER` = 269, `ALPHANUMERIC` = 270, `ALPHANUMERIC_EDITED`  
= 271, `ALSO` = 272,
  - `ALTER` = 273, `ALTERNATE` = 274, `AND` = 275, `ANY` = 276,
  - `ARE` = 277, `AREA` = 278, `ARGUMENT_NUMBER` = 279, `ARGUMENT_VALUE`  
= 280,
  - `AS` = 281, `ASCENDING` = 282, `ASSIGN` = 283, `AT` = 284,
  - `AUTO` = 285, `AUTOMATIC` = 286, `BACKGROUND_COLOR` = 287, `BASED` = 288,
  - `BEFORE` = 289, `BELL` = 290, `BINARY` = 291, `BINARY_C_LONG` = 292,
  - `BINARY_CHAR` = 293, `BINARY_DOUBLE` = 294, `BINARY_LONG` = 295, `BINARY_`  
`SHORT` = 296,
  - `BLANK` = 297, `BLANK_LINE` = 298, `BLANK_SCREEN` = 299, `BLINK` = 300,
  - `BLOCK` = 301, `BOTTOM` = 302, `BY` = 303, `BYTE_LENGTH` = 304,
  - `CALL` = 305, `CANCEL` = 306, `CH` = 307, `CHAINING` = 308,
  - `CHARACTER` = 309, `CHARACTERS` = 310, `CLASS` = 311, `CLOSE` = 312,
  - `CODE` = 313, `CODE_SET` = 314, `COLLATING` = 315, `COL` = 316,
  - `COLS` = 317, `COLUMN` = 318, `COLUMNS` = 319, `COMMA` = 320,
  - `COMMAND_LINE` = 321, `COMMA_DELIM` = 322, `COMMIT` = 323, `COMMON` =  
324,
  - `COMP` = 325, `COMPUTE` = 326, `COMP_1` = 327, `COMP_2` = 328,

COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING =  
343, CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE =  
360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING =  
364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE =  
380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_-  
FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM  
= 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH  
= 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING  
= 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_-  
NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION =  
444,

IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOW-  
LIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED =  
500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_-  
OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUM-  
BERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_-  
COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL =  
536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID =  
556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE  
= 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,

REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING =  
580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE  
= 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_  
VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT  
= 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
COMMAND\_LINE = 672,  
UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
UPPER\_CASE\_FUNC = 675, USAGE = 676,  
USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
TOKEN\_EOF = 0, ACCEPT = 258, ACCESS = 259, ADD = 260,

ADDRESS = 261, ADVANCING = 262, AFTER = 263, ALL = 264,  
ALLOCATE = 265, ALPHABET = 266, ALPHABETIC = 267, ALPHABETIC\_-  
LOWER = 268,  
ALPHABETIC\_UPPER = 269, ALPHANUMERIC = 270, ALPHANUMERIC\_EDITED  
= 271, ALSO = 272,  
ALTER = 273, ALTERNATE = 274, AND = 275, ANY = 276,  
ARE = 277, AREA = 278, ARGUMENT\_NUMBER = 279, ARGUMENT\_VALUE  
= 280,  
AS = 281, ASCENDING = 282, ASSIGN = 283, AT = 284,  
AUTO = 285, AUTOMATIC = 286, BACKGROUND\_COLOR = 287, BASED = 288,  
BEFORE = 289, BELL = 290, BINARY = 291, BINARY\_C\_LONG = 292,  
BINARY\_CHAR = 293, BINARY\_DOUBLE = 294, BINARY\_LONG = 295, BINARY\_-  
SHORT = 296,  
BLANK = 297, BLANK\_LINE = 298, BLANK\_SCREEN = 299, BLINK = 300,  
BLOCK = 301, BOTTOM = 302, BY = 303, BYTE\_LENGTH = 304,  
CALL = 305, CANCEL = 306, CH = 307, CHAINING = 308,  
CHARACTER = 309, CHARACTERS = 310, CLASS = 311, CLOSE = 312,  
CODE = 313, CODE\_SET = 314, COLLATING = 315, COL = 316,  
COLS = 317, COLUMN = 318, COLUMNS = 319, COMMA = 320,  
COMMAND\_LINE = 321, COMMA\_DELIM = 322, COMMIT = 323, COMMON =  
324,  
COMP = 325, COMPUTE = 326, COMP\_1 = 327, COMP\_2 = 328,  
COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING =  
343, CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE =  
360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING =  
364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE =  
380,

END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM = 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH = 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING = 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION = 444,  
IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOWLIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED = 500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,

NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUMBERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL = 536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID = 556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE = 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,  
REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING = 580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE = 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT = 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES = 624,



**STANDARD** = 625, **STANDARD\_1** = 626, **STANDARD\_2** = 627, **START** = 628,  
**STATUS** = 629, **STOP** = 630, **STRING** = 631, **SUBSTITUTE\_FUNC** = 632,  
**SUBSTITUTE\_CASE\_FUNC** = 633, **SUBTRACT** = 634, **SUM** = 635, **SUPPRESS**  
= 636,  
**SYMBOLIC** = 637, **SYNCHRONIZED** = 638, **TALLYING** = 639, **TAPE** = 640,  
**TERMINATE** = 641, **TEST** = 642, **THAN** = 643, **THEN** = 644,  
**THRU** = 645, **TIME** = 646, **TIMES** = 647, **TO** = 648,  
**TOK\_FALSE** = 649, **TOK\_FILE** = 650, **TOK\_INITIAL** = 651, **TOK\_NULL** = 652,  
**TOK\_TRUE** = 653, **TOP** = 654, **TRAILING** = 655, **TRANSFORM** = 656,  
**TRIM\_FUNCTION** = 657, **TYPE** = 658, **UNDERLINE** = 659, **UNIT** = 660,  
**UNLOCK** = 661, **UNSIGNED** = 662, **UNSIGNED\_INT** = 663, **UNSIGNED\_LONG**  
= 664,  
**UNSIGNED\_SHORT** = 665, **UNSTRING** = 666, **UNTIL** = 667, **UP** = 668,  
**UPDATE** = 669, **UPON** = 670, **UPON\_ARGUMENT\_NUMBER** = 671, **UPON\_**  
**COMMAND\_LINE** = 672,  
**UPON\_ENVIRONMENT\_NAME** = 673, **UPON\_ENVIRONMENT\_VALUE** = 674,  
**UPPER\_CASE\_FUNC** = 675, **USAGE** = 676,  
**USE** = 677, **USING** = 678, **VALUE** = 679, **VARYING** = 680,  
**WAIT** = 681, **WHEN** = 682, **WHEN\_COMPILED\_FUNC** = 683, **WITH** = 684,  
**WORD** = 685, **WORDS** = 686, **WORKING\_STORAGE** = 687, **WRITE** = 688,  
**YYYYDDD** = 689, **YYYYMMDD** = 690, **ZERO** = 691, **UNARY\_SIGN** = 692,  
**TOKEN\_EOF** = 0, **COPY** = 258, **REPLACE** = 259, **SUPPRESS** = 260,  
**PRINTING** = 261, **REPLACING** = 262, **OFF** = 263, **IN** = 264,  
**OF** = 265, **BY** = 266, **EQEQ** = 267, **TOKEN** = 268,  
**TOKEN\_EOF** = 0, **COPY** = 258, **REPLACE** = 259, **SUPPRESS** = 260,  
**PRINTING** = 261, **REPLACING** = 262, **OFF** = 263, **IN** = 264,  
**OF** = 265, **BY** = 266, **EQEQ** = 267, **TOKEN** = 268 }

## Functions

- int `yyparse` ()

## Variables

- struct `cb_program` \* `current_program` = NULL
- struct `cb_statement` \* `current_statement` = NULL
- struct `cb_label` \* `current_section` = NULL
- struct `cb_label` \* `current_paragraph` = NULL
- size\_t `functions_are_all` = 0
- int `non_const_word` = 0

- int [yydebug](#)
- int [yychar](#)
- YYSTYPE [yylval](#)
- int [yynerrs](#)

## 7.8.1 Define Documentation

### 7.8.1.1 #define ACCEPT 258

Definition at line 495 of file parser.c.

### 7.8.1.2 #define ACCESS 259

Definition at line 496 of file parser.c.

### 7.8.1.3 #define ADD 260

Definition at line 497 of file parser.c.

### 7.8.1.4 #define ADDRESS 261

Definition at line 498 of file parser.c.

### 7.8.1.5 #define ADVANCING 262

Definition at line 499 of file parser.c.

### 7.8.1.6 #define AFTER 263

Definition at line 500 of file parser.c.

### 7.8.1.7 #define ALL 264

Definition at line 501 of file parser.c.

### 7.8.1.8 #define ALLOCATE 265

Definition at line 502 of file parser.c.

### 7.8.1.9 #define ALPHABET 266

Definition at line 503 of file parser.c.

**7.8.1.10 #define ALPHABETIC 267**

Definition at line 504 of file parser.c.

**7.8.1.11 #define ALPHABETIC\_LOWER 268**

Definition at line 505 of file parser.c.

**7.8.1.12 #define ALPHABETIC\_UPPER 269**

Definition at line 506 of file parser.c.

**7.8.1.13 #define ALPHANUMERIC 270**

Definition at line 507 of file parser.c.

**7.8.1.14 #define ALPHANUMERIC\_EDITED 271**

Definition at line 508 of file parser.c.

**7.8.1.15 #define ALSO 272**

Definition at line 509 of file parser.c.

**7.8.1.16 #define ALTER 273**

Definition at line 510 of file parser.c.

**7.8.1.17 #define ALTERNATE 274**

Definition at line 511 of file parser.c.

**7.8.1.18 #define AND 275**

Definition at line 512 of file parser.c.

**7.8.1.19 #define ANY 276**

Definition at line 513 of file parser.c.

7.8.1.20 `#define ARE 277`

Definition at line 514 of file parser.c.

7.8.1.21 `#define AREA 278`

Definition at line 515 of file parser.c.

7.8.1.22 `#define ARGUMENT_NUMBER 279`

Definition at line 516 of file parser.c.

7.8.1.23 `#define ARGUMENT_VALUE 280`

Definition at line 517 of file parser.c.

7.8.1.24 `#define AS 281`

Definition at line 518 of file parser.c.

7.8.1.25 `#define ASCENDING 282`

Definition at line 519 of file parser.c.

7.8.1.26 `#define ASSIGN 283`

Definition at line 520 of file parser.c.

7.8.1.27 `#define AT 284`

Definition at line 521 of file parser.c.

7.8.1.28 `#define AUTO 285`

Definition at line 522 of file parser.c.

7.8.1.29 `#define AUTOMATIC 286`

Definition at line 523 of file parser.c.

**7.8.1.30 #define BACKGROUND\_COLOR 287**

Definition at line 524 of file parser.c.

**7.8.1.31 #define BASED 288**

Definition at line 525 of file parser.c.

**7.8.1.32 #define BEFORE 289**

Definition at line 526 of file parser.c.

**7.8.1.33 #define BELL 290**

Definition at line 527 of file parser.c.

**7.8.1.34 #define BINARY 291**

Definition at line 528 of file parser.c.

**7.8.1.35 #define BINARY\_C\_LONG 292**

Definition at line 529 of file parser.c.

**7.8.1.36 #define BINARY\_CHAR 293**

Definition at line 530 of file parser.c.

**7.8.1.37 #define BINARY\_DOUBLE 294**

Definition at line 531 of file parser.c.

**7.8.1.38 #define BINARY\_LONG 295**

Definition at line 532 of file parser.c.

**7.8.1.39 #define BINARY\_SHORT 296**

Definition at line 533 of file parser.c.

7.8.1.40 `#define BLANK 297`

Definition at line 534 of file parser.c.

7.8.1.41 `#define BLANK_LINE 298`

Definition at line 535 of file parser.c.

7.8.1.42 `#define BLANK_SCREEN 299`

Definition at line 536 of file parser.c.

7.8.1.43 `#define BLINK 300`

Definition at line 537 of file parser.c.

7.8.1.44 `#define BLOCK 301`

Definition at line 538 of file parser.c.

7.8.1.45 `#define BOTTOM 302`

Definition at line 539 of file parser.c.

7.8.1.46 `#define BY 303`

Definition at line 540 of file parser.c.

7.8.1.47 `#define BYTE_LENGTH 304`

Definition at line 541 of file parser.c.

7.8.1.48 `#define CALL 305`

Definition at line 542 of file parser.c.

7.8.1.49 `#define CANCEL 306`

Definition at line 543 of file parser.c.

**7.8.1.50 #define CH 307**

Definition at line 544 of file parser.c.

**7.8.1.51 #define CHAINING 308**

Definition at line 545 of file parser.c.

**7.8.1.52 #define CHARACTER 309**

Definition at line 546 of file parser.c.

**7.8.1.53 #define CHARACTERS 310**

Definition at line 547 of file parser.c.

**7.8.1.54 #define CLASS 311**

Definition at line 548 of file parser.c.

**7.8.1.55 #define CLOSE 312**

Definition at line 549 of file parser.c.

**7.8.1.56 #define CODE 313**

Definition at line 550 of file parser.c.

**7.8.1.57 #define CODE\_SET 314**

Definition at line 551 of file parser.c.

**7.8.1.58 #define COL 316**

Definition at line 553 of file parser.c.

**7.8.1.59 #define COLLATING 315**

Definition at line 552 of file parser.c.

7.8.1.60 `#define COLS 317`

Definition at line 554 of file parser.c.

7.8.1.61 `#define COLUMN 318`

Definition at line 555 of file parser.c.

7.8.1.62 `#define COLUMNS 319`

Definition at line 556 of file parser.c.

7.8.1.63 `#define COMMA 320`

Definition at line 557 of file parser.c.

7.8.1.64 `#define COMMA_DELIM 322`

Definition at line 559 of file parser.c.

7.8.1.65 `#define COMMAND_LINE 321`

Definition at line 558 of file parser.c.

7.8.1.66 `#define COMMIT 323`

Definition at line 560 of file parser.c.

7.8.1.67 `#define COMMON 324`

Definition at line 561 of file parser.c.

7.8.1.68 `#define COMP 325`

Definition at line 562 of file parser.c.

7.8.1.69 `#define COMP_1 327`

Definition at line 564 of file parser.c.



**7.8.1.70 #define COMP\_2 328**

Definition at line 565 of file parser.c.

**7.8.1.71 #define COMP\_3 329**

Definition at line 566 of file parser.c.

**7.8.1.72 #define COMP\_4 330**

Definition at line 567 of file parser.c.

**7.8.1.73 #define COMP\_5 331**

Definition at line 568 of file parser.c.

**7.8.1.74 #define COMP\_X 332**

Definition at line 569 of file parser.c.

**7.8.1.75 #define COMPUTE 326**

Definition at line 563 of file parser.c.

**7.8.1.76 #define CONCATENATE\_FUNC 333**

Definition at line 570 of file parser.c.

**7.8.1.77 #define CONFIGURATION 334**

Definition at line 571 of file parser.c.

**7.8.1.78 #define CONSTANT 335**

Definition at line 572 of file parser.c.

**7.8.1.79 #define CONTAINS 336**

Definition at line 573 of file parser.c.

7.8.1.80 `#define CONTENT 337`

Definition at line 574 of file parser.c.

7.8.1.81 `#define CONTINUE 338`

Definition at line 575 of file parser.c.

7.8.1.82 `#define CONTROL 339`

Definition at line 576 of file parser.c.

7.8.1.83 `#define CONTROL_FOOTING 341`

Definition at line 578 of file parser.c.

7.8.1.84 `#define CONTROL_HEADING 342`

Definition at line 579 of file parser.c.

7.8.1.85 `#define CONTROLS 340`

Definition at line 577 of file parser.c.

7.8.1.86 `#define CONVERTING 343`

Definition at line 580 of file parser.c.

7.8.1.87 `#define CORRESPONDING 344`

Definition at line 581 of file parser.c.

7.8.1.88 `#define COUNT 345`

Definition at line 582 of file parser.c.

7.8.1.89 `#define CRT 346`

Definition at line 583 of file parser.c.

7.8.1.90 `#define CURRENCY` 347

Definition at line 584 of file parser.c.

7.8.1.91 `#define CURRENT_DATE_FUNC` 348

Definition at line 585 of file parser.c.

7.8.1.92 `#define CURSOR` 349

Definition at line 586 of file parser.c.

7.8.1.93 `#define CYCLE` 350

Definition at line 587 of file parser.c.

7.8.1.94 `#define DATA` 351

Definition at line 588 of file parser.c.

7.8.1.95 `#define DATE` 352

Definition at line 589 of file parser.c.

7.8.1.96 `#define DAY` 353

Definition at line 590 of file parser.c.

7.8.1.97 `#define DAY_OF_WEEK` 354

Definition at line 591 of file parser.c.

7.8.1.98 `#define DE` 355

Definition at line 592 of file parser.c.

7.8.1.99 `#define DEBUGGING` 356

Definition at line 593 of file parser.c.

7.8.1.100 `#define DECIMAL_POINT 357`

Definition at line 594 of file parser.c.

7.8.1.101 `#define DECLARATIVES 358`

Definition at line 595 of file parser.c.

7.8.1.102 `#define DEFAULT 359`

Definition at line 596 of file parser.c.

7.8.1.103 `#define DELETE 360`

Definition at line 597 of file parser.c.

7.8.1.104 `#define DELIMITED 361`

Definition at line 598 of file parser.c.

7.8.1.105 `#define DELIMITER 362`

Definition at line 599 of file parser.c.

7.8.1.106 `#define DEPENDING 363`

Definition at line 600 of file parser.c.

7.8.1.107 `#define DESCENDING 364`

Definition at line 601 of file parser.c.

7.8.1.108 `#define DETAIL 365`

Definition at line 602 of file parser.c.

7.8.1.109 `#define DISK 366`

Definition at line 603 of file parser.c.

7.8.1.110 `#define DISPLAY 367`

Definition at line 604 of file parser.c.

7.8.1.111 `#define DIVIDE 368`

Definition at line 605 of file parser.c.

7.8.1.112 `#define DIVISION 369`

Definition at line 606 of file parser.c.

7.8.1.113 `#define DOWN 370`

Definition at line 607 of file parser.c.

7.8.1.114 `#define DUPLICATES 371`

Definition at line 608 of file parser.c.

7.8.1.115 `#define DYNAMIC 372`

Definition at line 609 of file parser.c.

7.8.1.116 `#define EBCDIC 373`

Definition at line 610 of file parser.c.

7.8.1.117 `#define ELSE 374`

Definition at line 611 of file parser.c.

7.8.1.118 `#define emit_statement( x ) current_program->exec_list = cb_cons (x,  
current_program->exec_list)`

Definition at line 951 of file parser.c.

7.8.1.119 `#define END 375`

Definition at line 612 of file parser.c.

7.8.1.120 `#define END_ACCEPT 376`

Definition at line 613 of file parser.c.

7.8.1.121 `#define END_ADD 377`

Definition at line 614 of file parser.c.

7.8.1.122 `#define END_CALL 378`

Definition at line 615 of file parser.c.

7.8.1.123 `#define END_COMPUTE 379`

Definition at line 616 of file parser.c.

7.8.1.124 `#define END_DELETE 380`

Definition at line 617 of file parser.c.

7.8.1.125 `#define END_DISPLAY 381`

Definition at line 618 of file parser.c.

7.8.1.126 `#define END_DIVIDE 382`

Definition at line 619 of file parser.c.

7.8.1.127 `#define END_EVALUATE 383`

Definition at line 620 of file parser.c.

7.8.1.128 `#define END_FUNCTION 384`

Definition at line 621 of file parser.c.

7.8.1.129 `#define END_IF 385`

Definition at line 622 of file parser.c.

7.8.1.130 `#define END_MULTIPLY 386`

Definition at line 623 of file parser.c.

7.8.1.131 `#define END_PERFORM 387`

Definition at line 624 of file parser.c.

7.8.1.132 `#define END_PROGRAM 388`

Definition at line 625 of file parser.c.

7.8.1.133 `#define END_READ 389`

Definition at line 626 of file parser.c.

7.8.1.134 `#define END_RETURN 390`

Definition at line 627 of file parser.c.

7.8.1.135 `#define END_REWRITE 391`

Definition at line 628 of file parser.c.

7.8.1.136 `#define END_SEARCH 392`

Definition at line 629 of file parser.c.

7.8.1.137 `#define END_START 393`

Definition at line 630 of file parser.c.

7.8.1.138 `#define END_STRING 394`

Definition at line 631 of file parser.c.

7.8.1.139 `#define END_SUBTRACT 395`

Definition at line 632 of file parser.c.

7.8.1.140 **#define END\_UNSTRING 396**

Definition at line 633 of file parser.c.

7.8.1.141 **#define END\_WRITE 397**

Definition at line 634 of file parser.c.

7.8.1.142 **#define ENTRY 398**

Definition at line 635 of file parser.c.

7.8.1.143 **#define ENVIRONMENT 399**

Definition at line 636 of file parser.c.

7.8.1.144 **#define ENVIRONMENT\_NAME 400**

Definition at line 637 of file parser.c.

7.8.1.145 **#define ENVIRONMENT\_VALUE 401**

Definition at line 638 of file parser.c.

7.8.1.146 **#define EOL 402**

Definition at line 639 of file parser.c.

7.8.1.147 **#define EOP 403**

Definition at line 640 of file parser.c.

7.8.1.148 **#define EOS 404**

Definition at line 641 of file parser.c.

7.8.1.149 **#define EQUAL 405**

Definition at line 642 of file parser.c.



7.8.1.150 `#define EQUALS 406`

Definition at line 643 of file parser.c.

7.8.1.151 `#define ERASE 407`

Definition at line 644 of file parser.c.

7.8.1.152 `#define ERROR 408`

Definition at line 645 of file parser.c.

7.8.1.153 `#define ESCAPE 409`

Definition at line 646 of file parser.c.

7.8.1.154 `#define EVALUATE 410`

Definition at line 647 of file parser.c.

7.8.1.155 `#define EVENT_STATUS 411`

Definition at line 648 of file parser.c.

7.8.1.156 `#define EXCEPTION 412`

Definition at line 649 of file parser.c.

7.8.1.157 `#define EXCLUSIVE 413`

Definition at line 650 of file parser.c.

7.8.1.158 `#define EXIT 414`

Definition at line 651 of file parser.c.

7.8.1.159 `#define EXTEND 415`

Definition at line 652 of file parser.c.

7.8.1.160 `#define EXTERNAL 416`

Definition at line 653 of file parser.c.

7.8.1.161 `#define FD 417`

Definition at line 654 of file parser.c.

7.8.1.162 `#define FILE_CONTROL 418`

Definition at line 655 of file parser.c.

7.8.1.163 `#define FILE_ID 419`

Definition at line 656 of file parser.c.

7.8.1.164 `#define FILLER 420`

Definition at line 657 of file parser.c.

7.8.1.165 `#define FINAL 421`

Definition at line 658 of file parser.c.

7.8.1.166 `#define FIRST 422`

Definition at line 659 of file parser.c.

7.8.1.167 `#define FOOTING 423`

Definition at line 660 of file parser.c.

7.8.1.168 `#define FOR 424`

Definition at line 661 of file parser.c.

7.8.1.169 `#define FOREGROUND_COLOR 425`

Definition at line 662 of file parser.c.

7.8.1.170 **#define FOREVER 426**

Definition at line 663 of file parser.c.

7.8.1.171 **#define FREE 427**

Definition at line 664 of file parser.c.

7.8.1.172 **#define FROM 428**

Definition at line 665 of file parser.c.

7.8.1.173 **#define FULL 429**

Definition at line 666 of file parser.c.

7.8.1.174 **#define FUNCTION 430**

Definition at line 667 of file parser.c.

7.8.1.175 **#define FUNCTION\_ID 431**

Definition at line 668 of file parser.c.

7.8.1.176 **#define FUNCTION\_NAME 432**

Definition at line 669 of file parser.c.

7.8.1.177 **#define GE 433**

Definition at line 670 of file parser.c.

7.8.1.178 **#define GENERATE 434**

Definition at line 671 of file parser.c.

7.8.1.179 **#define GIVING 435**

Definition at line 672 of file parser.c.

7.8.1.180 `#define GLOBAL 436`

Definition at line 673 of file parser.c.

7.8.1.181 `#define GO 437`

Definition at line 674 of file parser.c.

7.8.1.182 `#define GOBACK 438`

Definition at line 675 of file parser.c.

7.8.1.183 `#define GREATER 439`

Definition at line 676 of file parser.c.

7.8.1.184 `#define GROUP 440`

Definition at line 677 of file parser.c.

7.8.1.185 `#define HEADING 441`

Definition at line 678 of file parser.c.

7.8.1.186 `#define HIGH_VALUE 443`

Definition at line 680 of file parser.c.

7.8.1.187 `#define HIGHLIGHT 442`

Definition at line 679 of file parser.c.

7.8.1.188 `#define I_O 463`

Definition at line 700 of file parser.c.

7.8.1.189 `#define I_O_CONTROL 464`

Definition at line 701 of file parser.c.

7.8.1.190 `#define IDENTIFICATION 444`

Definition at line 681 of file parser.c.

7.8.1.191 `#define IF 445`

Definition at line 682 of file parser.c.

7.8.1.192 `#define IGNORE 446`

Definition at line 683 of file parser.c.

7.8.1.193 `#define IGNORING 447`

Definition at line 684 of file parser.c.

7.8.1.194 `#define IN 448`

Definition at line 685 of file parser.c.

7.8.1.195 `#define INDEX 449`

Definition at line 686 of file parser.c.

7.8.1.196 `#define INDEXED 450`

Definition at line 687 of file parser.c.

7.8.1.197 `#define INDICATE 451`

Definition at line 688 of file parser.c.

7.8.1.198 `#define INITIALIZE 452`

Definition at line 689 of file parser.c.

7.8.1.199 `#define INITIALIZED 453`

Definition at line 690 of file parser.c.

7.8.1.200 `#define INITIATE 454`

Definition at line 691 of file parser.c.

7.8.1.201 `#define INPUT 455`

Definition at line 692 of file parser.c.

7.8.1.202 `#define INPUT_OUTPUT 456`

Definition at line 693 of file parser.c.

7.8.1.203 `#define INSPECT 457`

Definition at line 694 of file parser.c.

7.8.1.204 `#define INTO 458`

Definition at line 695 of file parser.c.

7.8.1.205 `#define INTRINSIC 459`

Definition at line 696 of file parser.c.

7.8.1.206 `#define INVALID 460`

Definition at line 697 of file parser.c.

7.8.1.207 `#define INVALID_KEY 461`

Definition at line 698 of file parser.c.

7.8.1.208 `#define IS 462`

Definition at line 699 of file parser.c.

7.8.1.209 `#define JUSTIFIED 465`

Definition at line 702 of file parser.c.

7.8.1.210 `#define KEY 466`

Definition at line 703 of file parser.c.

7.8.1.211 `#define LABEL 467`

Definition at line 704 of file parser.c.

7.8.1.212 `#define LAST 468`

Definition at line 705 of file parser.c.

7.8.1.213 `#define LAST_DETAIL 469`

Definition at line 706 of file parser.c.

7.8.1.214 `#define LE 470`

Definition at line 707 of file parser.c.

7.8.1.215 `#define LEADING 471`

Definition at line 708 of file parser.c.

7.8.1.216 `#define LEFT 472`

Definition at line 709 of file parser.c.

7.8.1.217 `#define LENGTH 473`

Definition at line 710 of file parser.c.

7.8.1.218 `#define LESS 474`

Definition at line 711 of file parser.c.

7.8.1.219 `#define LIMIT 475`

Definition at line 712 of file parser.c.

7.8.1.220 `#define LIMITS 476`

Definition at line 713 of file parser.c.

7.8.1.221 `#define LINAGE 477`

Definition at line 714 of file parser.c.

7.8.1.222 `#define LINAGE_COUNTER 478`

Definition at line 715 of file parser.c.

7.8.1.223 `#define LINE 479`

Definition at line 716 of file parser.c.

7.8.1.224 `#define LINES 480`

Definition at line 717 of file parser.c.

7.8.1.225 `#define LINKAGE 481`

Definition at line 718 of file parser.c.

7.8.1.226 `#define LITERAL 482`

Definition at line 719 of file parser.c.

7.8.1.227 `#define LOCAL_STORAGE 485`

Definition at line 722 of file parser.c.

7.8.1.228 `#define LOCALE 483`

Definition at line 720 of file parser.c.

7.8.1.229 `#define LOCALE_DT_FUNC 484`

Definition at line 721 of file parser.c.



7.8.1.230 `#define LOCK 486`

Definition at line 723 of file parser.c.

7.8.1.231 `#define LOW_VALUE 489`

Definition at line 726 of file parser.c.

7.8.1.232 `#define LOWER_CASE_FUNC 487`

Definition at line 724 of file parser.c.

7.8.1.233 `#define LOWLIGHT 488`

Definition at line 725 of file parser.c.

7.8.1.234 `#define MANUAL 490`

Definition at line 727 of file parser.c.

7.8.1.235 `#define MEMORY 491`

Definition at line 728 of file parser.c.

7.8.1.236 `#define MERGE 492`

Definition at line 729 of file parser.c.

7.8.1.237 `#define MINUS 493`

Definition at line 730 of file parser.c.

7.8.1.238 `#define MNEMONIC_NAME 494`

Definition at line 731 of file parser.c.

7.8.1.239 `#define MODE 495`

Definition at line 732 of file parser.c.

7.8.1.240 **#define MOVE 496**

Definition at line 733 of file parser.c.

7.8.1.241 **#define MULTIPLE 497**

Definition at line 734 of file parser.c.

7.8.1.242 **#define MULTIPLY 498**

Definition at line 735 of file parser.c.

7.8.1.243 **#define NATIONAL 499**

Definition at line 736 of file parser.c.

7.8.1.244 **#define NATIONAL\_EDITED 500**

Definition at line 737 of file parser.c.

7.8.1.245 **#define NATIVE 501**

Definition at line 738 of file parser.c.

7.8.1.246 **#define NE 502**

Definition at line 739 of file parser.c.

7.8.1.247 **#define NEGATIVE 503**

Definition at line 740 of file parser.c.

7.8.1.248 **#define NEXT 504**

Definition at line 741 of file parser.c.

7.8.1.249 **#define NEXT\_SENTENCE 505**

Definition at line 742 of file parser.c.

7.8.1.250 `#define NO 506`

Definition at line 743 of file parser.c.

7.8.1.251 `#define NO_ADVANCING 514`

Definition at line 751 of file parser.c.

7.8.1.252 `#define NOT 507`

Definition at line 744 of file parser.c.

7.8.1.253 `#define NOT_END 508`

Definition at line 745 of file parser.c.

7.8.1.254 `#define NOT_EOP 509`

Definition at line 746 of file parser.c.

7.8.1.255 `#define NOT_EXCEPTION 510`

Definition at line 747 of file parser.c.

7.8.1.256 `#define NOT_INVALID_KEY 511`

Definition at line 748 of file parser.c.

7.8.1.257 `#define NOT_OVERFLOW 512`

Definition at line 749 of file parser.c.

7.8.1.258 `#define NOT_SIZE_ERROR 513`

Definition at line 750 of file parser.c.

7.8.1.259 `#define NUMBER 515`

Definition at line 752 of file parser.c.

7.8.1.260 `#define NUMBERS 516`

Definition at line 753 of file parser.c.

7.8.1.261 `#define NUMERIC 517`

Definition at line 754 of file parser.c.

7.8.1.262 `#define NUMERIC_EDITED 518`

Definition at line 755 of file parser.c.

7.8.1.263 `#define NUMVALC_FUNC 519`

Definition at line 756 of file parser.c.

7.8.1.264 `#define OBJECT_COMPUTER 520`

Definition at line 757 of file parser.c.

7.8.1.265 `#define OCCURS 521`

Definition at line 758 of file parser.c.

7.8.1.266 `#define OF 522`

Definition at line 759 of file parser.c.

7.8.1.267 `#define OFF 523`

Definition at line 760 of file parser.c.

7.8.1.268 `#define OMITTED 524`

Definition at line 761 of file parser.c.

7.8.1.269 `#define ON 525`

Definition at line 762 of file parser.c.

7.8.1.270 **#define ONLY 526**

Definition at line 763 of file parser.c.

7.8.1.271 **#define OPEN 527**

Definition at line 764 of file parser.c.

7.8.1.272 **#define OPTIONAL 528**

Definition at line 765 of file parser.c.

7.8.1.273 **#define OR 529**

Definition at line 766 of file parser.c.

7.8.1.274 **#define ORDER 530**

Definition at line 767 of file parser.c.

7.8.1.275 **#define ORGANIZATION 531**

Definition at line 768 of file parser.c.

7.8.1.276 **#define OTHER 532**

Definition at line 769 of file parser.c.

7.8.1.277 **#define OUTPUT 533**

Definition at line 770 of file parser.c.

7.8.1.278 **#define OVERFLOW 534**

Definition at line 771 of file parser.c.

7.8.1.279 **#define OVERLINE 535**

Definition at line 772 of file parser.c.

7.8.1.280 `#define PACKED_DECIMAL 536`

Definition at line 773 of file parser.c.

7.8.1.281 `#define PADDING 537`

Definition at line 774 of file parser.c.

7.8.1.282 `#define PAGE 538`

Definition at line 775 of file parser.c.

7.8.1.283 `#define PAGE_FOOTING 539`

Definition at line 776 of file parser.c.

7.8.1.284 `#define PAGE_HEADING 540`

Definition at line 777 of file parser.c.

7.8.1.285 `#define PARAGRAPH 541`

Definition at line 778 of file parser.c.

7.8.1.286 `#define PENDING( x ) cb_warning ( _("%s' not implemented"), x)`

Definition at line 949 of file parser.c.

7.8.1.287 `#define PERFORM 542`

Definition at line 779 of file parser.c.

7.8.1.288 `#define PICTURE 543`

Definition at line 780 of file parser.c.

7.8.1.289 `#define PLUS 544`

Definition at line 781 of file parser.c.

7.8.1.290 `#define POINTER 545`

Definition at line 782 of file parser.c.

7.8.1.291 `#define POSITION 546`

Definition at line 783 of file parser.c.

7.8.1.292 `#define POSITIVE 547`

Definition at line 784 of file parser.c.

7.8.1.293 `#define PRESENT 548`

Definition at line 785 of file parser.c.

7.8.1.294 `#define PREVIOUS 549`

Definition at line 786 of file parser.c.

7.8.1.295 `#define PRINTER 550`

Definition at line 787 of file parser.c.

7.8.1.296 `#define PRINTING 551`

Definition at line 788 of file parser.c.

7.8.1.297 `#define PROCEDURE 552`

Definition at line 789 of file parser.c.

7.8.1.298 `#define PROCEDURES 553`

Definition at line 790 of file parser.c.

7.8.1.299 `#define PROCEED 554`

Definition at line 791 of file parser.c.

7.8.1.300 `#define PROGRAM 555`

Definition at line 792 of file parser.c.

7.8.1.301 `#define PROGRAM_ID 556`

Definition at line 793 of file parser.c.

7.8.1.302 `#define PROGRAM_NAME 557`

Definition at line 794 of file parser.c.

7.8.1.303 `#define PROGRAM_POINTER 558`

Definition at line 795 of file parser.c.

7.8.1.304 `#define PROMPT 559`

Definition at line 796 of file parser.c.

7.8.1.305 `#define push_expr( type, node ) current_expr = cb_build_list( cb_int( type), node, current_expr)`

Definition at line 954 of file parser.c.

7.8.1.306 `#define QUOTE 560`

Definition at line 797 of file parser.c.

7.8.1.307 `#define RANDOM 561`

Definition at line 798 of file parser.c.

7.8.1.308 `#define RD 562`

Definition at line 799 of file parser.c.

7.8.1.309 `#define READ 563`

Definition at line 800 of file parser.c.



7.8.1.310 **#define RECORD 564**

Definition at line 801 of file parser.c.

7.8.1.311 **#define RECORDING 565**

Definition at line 802 of file parser.c.

7.8.1.312 **#define RECORDS 566**

Definition at line 803 of file parser.c.

7.8.1.313 **#define RECURSIVE 567**

Definition at line 804 of file parser.c.

7.8.1.314 **#define REDEFINES 568**

Definition at line 805 of file parser.c.

7.8.1.315 **#define REEL 569**

Definition at line 806 of file parser.c.

7.8.1.316 **#define REFERENCE 570**

Definition at line 807 of file parser.c.

7.8.1.317 **#define RELATIVE 571**

Definition at line 808 of file parser.c.

7.8.1.318 **#define RELEASE 572**

Definition at line 809 of file parser.c.

7.8.1.319 **#define REMAINDER 573**

Definition at line 810 of file parser.c.

7.8.1.320 `#define REMOVAL 574`

Definition at line 811 of file parser.c.

7.8.1.321 `#define RENAMES 575`

Definition at line 812 of file parser.c.

7.8.1.322 `#define REPLACING 576`

Definition at line 813 of file parser.c.

7.8.1.323 `#define REPORT 577`

Definition at line 814 of file parser.c.

7.8.1.324 `#define REPORT_FOOTING 580`

Definition at line 817 of file parser.c.

7.8.1.325 `#define REPORT_HEADING 581`

Definition at line 818 of file parser.c.

7.8.1.326 `#define REPORTING 578`

Definition at line 815 of file parser.c.

7.8.1.327 `#define REPORTS 579`

Definition at line 816 of file parser.c.

7.8.1.328 `#define REPOSITORY 582`

Definition at line 819 of file parser.c.

7.8.1.329 `#define REQUIRED 583`

Definition at line 820 of file parser.c.

7.8.1.330 `#define RESERVE 584`

Definition at line 821 of file parser.c.

7.8.1.331 `#define RETURN 585`

Definition at line 822 of file parser.c.

7.8.1.332 `#define RETURNING 586`

Definition at line 823 of file parser.c.

7.8.1.333 `#define REVERSE_FUNC 587`

Definition at line 824 of file parser.c.

7.8.1.334 `#define REVERSE_VIDEO 588`

Definition at line 825 of file parser.c.

7.8.1.335 `#define REWIND 589`

Definition at line 826 of file parser.c.

7.8.1.336 `#define REWRITE 590`

Definition at line 827 of file parser.c.

7.8.1.337 `#define RIGHT 591`

Definition at line 828 of file parser.c.

7.8.1.338 `#define ROLLBACK 592`

Definition at line 829 of file parser.c.

7.8.1.339 `#define ROUNDED 593`

Definition at line 830 of file parser.c.

7.8.1.340 `#define RUN 594`

Definition at line 831 of file parser.c.

7.8.1.341 `#define SAME 595`

Definition at line 832 of file parser.c.

7.8.1.342 `#define SCREEN 596`

Definition at line 833 of file parser.c.

7.8.1.343 `#define SCREEN_CONTROL 597`

Definition at line 834 of file parser.c.

7.8.1.344 `#define SCROLL 598`

Definition at line 835 of file parser.c.

7.8.1.345 `#define SD 599`

Definition at line 836 of file parser.c.

7.8.1.346 `#define SEARCH 600`

Definition at line 837 of file parser.c.

7.8.1.347 `#define SECTION 601`

Definition at line 838 of file parser.c.

7.8.1.348 `#define SECURE 602`

Definition at line 839 of file parser.c.

7.8.1.349 `#define SEGMENT_LIMIT 603`

Definition at line 840 of file parser.c.

7.8.1.350 `#define SELECT 604`

Definition at line 841 of file parser.c.

7.8.1.351 `#define SEMI_COLON 605`

Definition at line 842 of file parser.c.

7.8.1.352 `#define SENTENCE 606`

Definition at line 843 of file parser.c.

7.8.1.353 `#define SEPARATE 607`

Definition at line 844 of file parser.c.

7.8.1.354 `#define SEQUENCE 608`

Definition at line 845 of file parser.c.

7.8.1.355 `#define SEQUENTIAL 609`

Definition at line 846 of file parser.c.

7.8.1.356 `#define SET 610`

Definition at line 847 of file parser.c.

7.8.1.357 `#define SHARING 611`

Definition at line 848 of file parser.c.

7.8.1.358 `#define SIGN 612`

Definition at line 849 of file parser.c.

7.8.1.359 `#define SIGNED 613`

Definition at line 850 of file parser.c.

7.8.1.360 `#define SIGNED_INT 614`

Definition at line 851 of file parser.c.

7.8.1.361 `#define SIGNED_LONG 615`

Definition at line 852 of file parser.c.

7.8.1.362 `#define SIGNED_SHORT 616`

Definition at line 853 of file parser.c.

7.8.1.363 `#define SIZE 617`

Definition at line 854 of file parser.c.

7.8.1.364 `#define SIZE_ERROR 618`

Definition at line 855 of file parser.c.

7.8.1.365 `#define SORT 619`

Definition at line 856 of file parser.c.

7.8.1.366 `#define SORT_MERGE 620`

Definition at line 857 of file parser.c.

7.8.1.367 `#define SOURCE 621`

Definition at line 858 of file parser.c.

7.8.1.368 `#define SOURCE_COMPUTER 622`

Definition at line 859 of file parser.c.

7.8.1.369 `#define SPACE 623`

Definition at line 860 of file parser.c.

**7.8.1.370 #define SPECIAL\_NAMES 624**

Definition at line 861 of file parser.c.

**7.8.1.371 #define STANDARD 625**

Definition at line 862 of file parser.c.

**7.8.1.372 #define STANDARD\_1 626**

Definition at line 863 of file parser.c.

**7.8.1.373 #define STANDARD\_2 627**

Definition at line 864 of file parser.c.

**7.8.1.374 #define START 628**

Definition at line 865 of file parser.c.

**7.8.1.375 #define STATUS 629**

Definition at line 866 of file parser.c.

**7.8.1.376 #define STOP 630**

Definition at line 867 of file parser.c.

**7.8.1.377 #define STRING 631**

Definition at line 868 of file parser.c.

**7.8.1.378 #define SUBSTITUTE\_CASE\_FUNC 633**

Definition at line 870 of file parser.c.

**7.8.1.379 #define SUBSTITUTE\_FUNC 632**

Definition at line 869 of file parser.c.

7.8.1.380 `#define SUBTRACT 634`

Definition at line 871 of file parser.c.

7.8.1.381 `#define SUM 635`

Definition at line 872 of file parser.c.

7.8.1.382 `#define SUPPRESS 636`

Definition at line 873 of file parser.c.

7.8.1.383 `#define SYMBOLIC 637`

Definition at line 874 of file parser.c.

7.8.1.384 `#define SYNCHRONIZED 638`

Definition at line 875 of file parser.c.

7.8.1.385 `#define TALLYING 639`

Definition at line 876 of file parser.c.

7.8.1.386 `#define TAPE 640`

Definition at line 877 of file parser.c.

7.8.1.387 `#define TERM_ACCEPT 1`

Definition at line 958 of file parser.c.

7.8.1.388 `#define TERM_ADD 2`

Definition at line 959 of file parser.c.

7.8.1.389 `#define TERM_CALL 3`

Definition at line 960 of file parser.c.



7.8.1.390 `#define TERM_COMPUTE 4`

Definition at line 961 of file parser.c.

7.8.1.391 `#define TERM_DELETE 5`

Definition at line 962 of file parser.c.

7.8.1.392 `#define TERM_DISPLAY 6`

Definition at line 963 of file parser.c.

7.8.1.393 `#define TERM_DIVIDE 7`

Definition at line 964 of file parser.c.

7.8.1.394 `#define TERM_EVALUATE 8`

Definition at line 965 of file parser.c.

7.8.1.395 `#define TERM_IF 9`

Definition at line 966 of file parser.c.

7.8.1.396 `#define TERM_MAX 22`

Definition at line 979 of file parser.c.

7.8.1.397 `#define TERM_MULTIPLY 10`

Definition at line 967 of file parser.c.

7.8.1.398 `#define TERM_NONE 0`

Definition at line 957 of file parser.c.

7.8.1.399 `#define TERM_PERFORM 11`

Definition at line 968 of file parser.c.

7.8.1.400 `#define TERM_READ 12`

Definition at line 969 of file parser.c.

7.8.1.401 `#define TERM_RECEIVE 13`

Definition at line 970 of file parser.c.

7.8.1.402 `#define TERM_RETURN 14`

Definition at line 971 of file parser.c.

7.8.1.403 `#define TERM_REWRITE 15`

Definition at line 972 of file parser.c.

7.8.1.404 `#define TERM_SEARCH 16`

Definition at line 973 of file parser.c.

7.8.1.405 `#define TERM_START 17`

Definition at line 974 of file parser.c.

7.8.1.406 `#define TERM_STRING 18`

Definition at line 975 of file parser.c.

7.8.1.407 `#define TERM_SUBTRACT 19`

Definition at line 976 of file parser.c.

7.8.1.408 `#define TERM_UNSTRING 20`

Definition at line 977 of file parser.c.

7.8.1.409 `#define TERM_WRITE 21`

Definition at line 978 of file parser.c.

7.8.1.410 `#define TERMINATE 641`

Definition at line 878 of file parser.c.

7.8.1.411 `#define TEST 642`

Definition at line 879 of file parser.c.

7.8.1.412 `#define THAN 643`

Definition at line 880 of file parser.c.

7.8.1.413 `#define THEN 644`

Definition at line 881 of file parser.c.

7.8.1.414 `#define THRU 645`

Definition at line 882 of file parser.c.

7.8.1.415 `#define TIME 646`

Definition at line 883 of file parser.c.

7.8.1.416 `#define TIMES 647`

Definition at line 884 of file parser.c.

7.8.1.417 `#define TO 648`

Definition at line 885 of file parser.c.

7.8.1.418 `#define TOK_FALSE 649`

Definition at line 886 of file parser.c.

7.8.1.419 `#define TOK_FILE 650`

Definition at line 887 of file parser.c.

7.8.1.420 `#define TOK_INITIAL 651`

Definition at line 888 of file parser.c.

7.8.1.421 `#define TOK_NULL 652`

Definition at line 889 of file parser.c.

7.8.1.422 `#define TOK_TRUE 653`

Definition at line 890 of file parser.c.

7.8.1.423 `#define TOKEN_EOF 0`

Definition at line 494 of file parser.c.

7.8.1.424 `#define TOP 654`

Definition at line 891 of file parser.c.

7.8.1.425 `#define TRAILING 655`

Definition at line 892 of file parser.c.

7.8.1.426 `#define TRANSFORM 656`

Definition at line 893 of file parser.c.

7.8.1.427 `#define TRIM_FUNCTION 657`

Definition at line 894 of file parser.c.

7.8.1.428 `#define TYPE 658`

Definition at line 895 of file parser.c.

7.8.1.429 `#define UNARY_SIGN 692`

Definition at line 929 of file parser.c.

7.8.1.430 `#define UNDERLINE 659`

Definition at line 896 of file parser.c.

7.8.1.431 `#define UNIT 660`

Definition at line 897 of file parser.c.

7.8.1.432 `#define UNLOCK 661`

Definition at line 898 of file parser.c.

7.8.1.433 `#define UNSIGNED 662`

Definition at line 899 of file parser.c.

7.8.1.434 `#define UNSIGNED_INT 663`

Definition at line 900 of file parser.c.

7.8.1.435 `#define UNSIGNED_LONG 664`

Definition at line 901 of file parser.c.

7.8.1.436 `#define UNSIGNED_SHORT 665`

Definition at line 902 of file parser.c.

7.8.1.437 `#define UNSTRING 666`

Definition at line 903 of file parser.c.

7.8.1.438 `#define UNTIL 667`

Definition at line 904 of file parser.c.

7.8.1.439 `#define UP 668`

Definition at line 905 of file parser.c.

7.8.1.440 `#define UPDATE 669`

Definition at line 906 of file parser.c.

7.8.1.441 `#define UPON 670`

Definition at line 907 of file parser.c.

7.8.1.442 `#define UPON_ARGUMENT_NUMBER 671`

Definition at line 908 of file parser.c.

7.8.1.443 `#define UPON_COMMAND_LINE 672`

Definition at line 909 of file parser.c.

7.8.1.444 `#define UPON_ENVIRONMENT_NAME 673`

Definition at line 910 of file parser.c.

7.8.1.445 `#define UPON_ENVIRONMENT_VALUE 674`

Definition at line 911 of file parser.c.

7.8.1.446 `#define UPPER_CASE_FUNC 675`

Definition at line 912 of file parser.c.

7.8.1.447 `#define USAGE 676`

Definition at line 913 of file parser.c.

7.8.1.448 `#define USE 677`

Definition at line 914 of file parser.c.

7.8.1.449 `#define USING 678`

Definition at line 915 of file parser.c.

**7.8.1.450 #define VALUE 679**

Definition at line 916 of file parser.c.

**7.8.1.451 #define VARYING 680**

Definition at line 917 of file parser.c.

**7.8.1.452 #define WAIT 681**

Definition at line 918 of file parser.c.

**7.8.1.453 #define WHEN 682**

Definition at line 919 of file parser.c.

**7.8.1.454 #define WHEN\_COMPILED\_FUNC 683**

Definition at line 920 of file parser.c.

**7.8.1.455 #define WITH 684**

Definition at line 921 of file parser.c.

**7.8.1.456 #define WORD 685**

Definition at line 922 of file parser.c.

**7.8.1.457 #define WORDS 686**

Definition at line 923 of file parser.c.

**7.8.1.458 #define WORKING\_STORAGE 687**

Definition at line 924 of file parser.c.

**7.8.1.459 #define WRITE 688**

Definition at line 925 of file parser.c.

**7.8.1.460 #define YY\_REDUCE\_PRINT( Rule )****Value:**

```
do {
    if (yydebug)
        yy_reduce_print (Rule);
} while (0)
```

Definition at line 4900 of file parser.c.

**7.8.1.461 #define YY\_STACK\_PRINT( Bottom, Top )****Value:**

```
do {
    if (yydebug)
        yy_stack_print ((Bottom), (Top));
} while (0)
```

Definition at line 4870 of file parser.c.

**7.8.1.462 #define YYABORT goto yyabortlab**

Definition at line 4768 of file parser.c.

**7.8.1.463 #define YYACCEPT goto yyacceptlab**

Definition at line 4767 of file parser.c.

**7.8.1.464 #define YYBACKUP( Token, Value )****Value:**

```
do
    if (yychar == YYEMPTY && yylen == 1)
    {
        yychar = (Token);
        yylval = (Value);
        yytoken = YYTRANSLATE (yychar);
        YYPOPSTACK;
        goto yybackup;
    }
    else
    {
        yyerror ("syntax error: cannot back up");\
        YYERROR;
    }
while (0)
```

Definition at line 4779 of file parser.c.



**7.8.1.465 #define YYBISON 1**

Definition at line 37 of file parser.c.

**7.8.1.466 #define yyclearin (yychar = YYEMPTY)**

Definition at line 4763 of file parser.c.

**7.8.1.467 #define YYCOPY( To, From, Count )****Value:**

```
do
    {
        register YYSIZE_T yyi;
        for (yyi = 0; yyi < (Count); yyi++)
            (To)[yyi] = (From)[yyi];
    }
    while (0)
```

Definition at line 1305 of file parser.c.

**7.8.1.468 #define YYDEBUG 1**

Definition at line 946 of file parser.c.

**7.8.1.469 #define YYDPRINTF( Args )****Value:**

```
do {
    if (yydebug)
        YYFPRINTF Args;
} while (0)
```

Definition at line 4826 of file parser.c.

**7.8.1.470 #define YYSYMPRINT( Args )****Value:**

```
do {
    if (yydebug)
        ysymprint Args;
} while (0)
```

Definition at line 4832 of file parser.c.

7.8.1.471 **#define YYDSYMPRINTF( Title, Token, Value, Location )**

**Value:**

```
do {
    if (yydebug)
    {
        YYFPRINTF (stderr, "%s ", Title);
        ysymprint (stderr,
                   Token, Value);
        YYFPRINTF (stderr, "\n");
    }
} while (0)
```

Definition at line 4838 of file parser.c.

7.8.1.472 **#define YYEMPTY (-2)**

Definition at line 4764 of file parser.c.

7.8.1.473 **#define YYEOF 0**

Definition at line 4765 of file parser.c.

7.8.1.474 **#define YYERRCODE 256**

Definition at line 4797 of file parser.c.

7.8.1.475 **#define yyerrok (yyerrstatus = 0)**

Definition at line 4762 of file parser.c.

7.8.1.476 **#define yyerror cb\_error**

Definition at line 945 of file parser.c.

7.8.1.477 **#define YYERROR goto yyerrlab1**

Definition at line 4769 of file parser.c.

7.8.1.478 **#define YYERROR\_VERBOSE 1**

Definition at line 1226 of file parser.c.

7.8.1.479 `#define YYERROR_VERBOSE 1`

Definition at line 1226 of file parser.c.

7.8.1.480 `#define YYFAIL goto yyerrlab`

Definition at line 4775 of file parser.c.

7.8.1.481 `#define YYFINAL 3`

Definition at line 1341 of file parser.c.

7.8.1.482 `#define YYFPRINTF fprintf`

Definition at line 4823 of file parser.c.

7.8.1.483 `#define YYINITDEPTH 200`

Definition at line 4920 of file parser.c.

7.8.1.484 `#define YYLAST 5462`

Definition at line 1343 of file parser.c.

7.8.1.485 `#define YYLEX yylex ()`

Definition at line 4815 of file parser.c.

7.8.1.486 `#define YYLLOC_DEFAULT( Current, Rhs, N )`

**Value:**

```
Current.first_line    = Rhs[1].first_line;      \
Current.first_column  = Rhs[1].first_column;    \
Current.last_line     = Rhs[N].last_line;       \
Current.last_column   = Rhs[N].last_column;
```

Definition at line 4803 of file parser.c.

7.8.1.487 `#define YYLSP_NEEDED 0`

Definition at line 46 of file parser.c.

7.8.1.488 `#define YYMAXDEPTH 10000`

Definition at line 4935 of file parser.c.

7.8.1.489 `#define YYMAXUTOK 692`

Definition at line 1356 of file parser.c.

7.8.1.490 `#define YYNNTS 690`

Definition at line 1348 of file parser.c.

7.8.1.491 `#define YYNRULES 1518`

Definition at line 1350 of file parser.c.

7.8.1.492 `#define YYNSTATES 2240`

Definition at line 1352 of file parser.c.

7.8.1.493 `#define YYNTOKENS 451`

Definition at line 1346 of file parser.c.

7.8.1.494 `#define YYPACT_NINF -1922`

Definition at line 3106 of file parser.c.

7.8.1.495 `#define YYPOPSTACK (yyvsp--, yyssp--)`

7.8.1.496 `#define YYPURE 0`

Definition at line 43 of file parser.c.

7.8.1.497 `#define YYRECOVERING( ) (!yyerrstatus)`

Definition at line 4777 of file parser.c.

7.8.1.498 `#define YYSIZE_T unsigned int`

Definition at line 4759 of file parser.c.

7.8.1.499 `#define YYSKELETON_NAME "yacc.c"`

Definition at line 40 of file parser.c.

7.8.1.500 `#define YYSTACK_ALLOC malloc`

Definition at line 1272 of file parser.c.

7.8.1.501 `#define YYSTACK_BYTES( N )`

**Value:**

```
((N) * (sizeof (short) + sizeof (YYSTYPE))
    + YYSTACK_GAP_MAXIMUM) \
```

Definition at line 1294 of file parser.c.

7.8.1.502 `#define YYSTACK_FREE free`

Definition at line 1273 of file parser.c.

7.8.1.503 `#define YYSTACK_GAP_MAXIMUM (sizeof (union yyallo) - 1)`

Definition at line 1290 of file parser.c.

7.8.1.504 `#define YYSTACK_RELOCATE( Stack )`

**Value:**

```
do \
{ \
    YYSIZE_T yynewbytes; \
    YYCOPY (&yyptr->Stack, Stack, yysize); \
    Stack = &yyptr->Stack; \
    yynewbytes = yyystacksize * sizeof (*Stack) + YYSTACK_GAP_MAXIMUM; \
    yyptr += yynewbytes / sizeof (*yyptr); \
} \
while (0)
```

Definition at line 1321 of file parser.c.

7.8.1.505 `#define YYTABLE_NINF -1518`

Definition at line 3413 of file parser.c.

7.8.1.506 `#define YYTERROR 1`

Definition at line 4796 of file parser.c.

**7.8.1.507 #define YYTOKENTYPE**

Definition at line 52 of file parser.c.

**7.8.1.508 #define YYTRANSLATE( YYX ) ((unsigned int) (YYX) <= YYMAXUTOK ?  
yytranslate[YYX] : YYUNDEFTOK)**

Definition at line 1358 of file parser.c.

**7.8.1.509 #define YYUNDEFTOK 2**

Definition at line 1355 of file parser.c.

**7.8.1.510 #define YYYYDDD 689**

Definition at line 926 of file parser.c.

**7.8.1.511 #define YYYYMMDD 690**

Definition at line 927 of file parser.c.

**7.8.1.512 #define ZERO 691**

Definition at line 928 of file parser.c.

**7.8.2 Typedef Documentation****7.8.2.1 typedef short yysigned\_char**

Definition at line 1337 of file parser.c.

**7.8.3 Enumeration Type Documentation****7.8.3.1 enum yytokentype**

**Enumerator:**

***TOKEN\_EOF***

***ACCEPT***

***ACCESS***

***ADD***

***ADDRESS***

***ADVANCING***

*AFTER*  
*ALL*  
*ALLOCATE*  
*ALPHABET*  
*ALPHABETIC*  
*ALPHABETIC\_LOWER*  
*ALPHABETIC\_UPPER*  
*ALPHANUMERIC*  
*ALPHANUMERIC\_EDITED*  
*ALSO*  
*ALTER*  
*ALTERNATE*  
*AND*  
*ANY*  
*ARE*  
*AREA*  
*ARGUMENT\_NUMBER*  
*ARGUMENT\_VALUE*  
*AS*  
*ASCENDING*  
*ASSIGN*  
*AT*  
*AUTO*  
*AUTOMATIC*  
*BACKGROUND\_COLOR*  
*BASED*  
*BEFORE*  
*BELL*  
*BINARY*  
*BINARY\_C\_LONG*  
*BINARY\_CHAR*  
*BINARY\_DOUBLE*  
*BINARY\_LONG*  
*BINARY\_SHORT*  
*BLANK*  
*BLANK\_LINE*  
*BLANK\_SCREEN*  
*BLINK*

**BLOCK**  
**BOTTOM**  
**BY**  
**BYTE\_LENGTH**  
**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**  
**COMMAND\_LINE**  
**COMMA\_DELIM**  
**COMMIT**  
**COMMON**  
**COMP**  
**COMPUTE**  
**COMP\_1**  
**COMP\_2**  
**COMP\_3**  
**COMP\_4**  
**COMP\_5**  
**COMP\_X**  
**CONCATENATE\_FUNC**  
**CONFIGURATION**  
**CONSTANT**  
**CONTAINS**  
**CONTENT**  
**CONTINUE**



*CONTROL*  
*CONTROLS*  
*CONTROL\_FOOTING*  
*CONTROL\_HEADING*  
*CONVERTING*  
*CORRESPONDING*  
*COUNT*  
*CRT*  
*CURRENCY*  
*CURRENT\_DATE\_FUNC*  
*CURSOR*  
*CYCLE*  
*DATA*  
*DATE*  
*DAY*  
*DAY\_OF\_WEEK*  
*DE*  
*DEBUGGING*  
*DECIMAL\_POINT*  
*DECLARATIVES*  
*DEFAULT*  
*DELETE*  
*DELIMITED*  
*DELIMITER*  
*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*

*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*

*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*FOREGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*  
*GROUP*  
*HEADING*  
*HIGHLIGHT*  
*HIGH\_VALUE*  
*IDENTIFICATION*  
*IF*  
*IGNORE*  
*IGNORING*  
*IN*  
*INDEX*  
*INDEXED*  
*INDICATE*  
*INITIALIZE*

*INITIALIZED*  
*INITIATE*  
*INPUT*  
*INPUT\_OUTPUT*  
*INSPECT*  
*INTO*  
*INTRINSIC*  
*INVALID*  
*INVALID\_KEY*  
*IS*  
*I\_O*  
*I\_O\_CONTROL*  
*JUSTIFIED*  
*KEY*  
*LABEL*  
*LAST*  
*LAST\_DETAIL*  
*LE*  
*LEADING*  
*LEFT*  
*LENGTH*  
*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*

*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*

**OR**  
**ORDER**  
**ORGANIZATION**  
**OTHER**  
**OUTPUT**  
**OVERFLOW**  
**OVERLINE**  
**PACKED\_DECIMAL**  
**PADDING**  
**PAGE**  
**PAGE\_FOOTING**  
**PAGE\_HEADING**  
**PARAGRAPH**  
**PERFORM**  
**PICTURE**  
**PLUS**  
**POINTER**  
**POSITION**  
**POSITIVE**  
**PRESENT**  
**PREVIOUS**  
**PRINTER**  
**PRINTING**  
**PROCEDURE**  
**PROCEDURES**  
**PROCEED**  
**PROGRAM**  
**PROGRAM\_ID**  
**PROGRAM\_NAME**  
**PROGRAM\_POINTER**  
**PROMPT**  
**QUOTE**  
**RANDOM**  
**RD**  
**READ**  
**RECORD**  
**RECORDING**  
**RECORDS**

*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*  
*REVERSE\_FUNC*  
*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*  
*RIGHT*  
*ROLLBACK*  
*ROUNDED*  
*RUN*  
*SAME*  
*SCREEN*  
*SCREEN\_CONTROL*  
*SCROLL*  
*SD*  
*SEARCH*  
*SECTION*  
*SECURE*  
*SEGMENT\_LIMIT*  
*SELECT*

**SEMI\_COLON**  
**SENTENCE**  
**SEPARATE**  
**SEQUENCE**  
**SEQUENTIAL**  
**SET**  
**SHARING**  
**SIGN**  
**SIGNED**  
**SIGNED\_INT**  
**SIGNED\_LONG**  
**SIGNED\_SHORT**  
**SIZE**  
**SIZE\_ERROR**  
**SORT**  
**SORT\_MERGE**  
**SOURCE**  
**SOURCE\_COMPUTER**  
**SPACE**  
**SPECIAL\_NAMES**  
**STANDARD**  
**STANDARD\_1**  
**STANDARD\_2**  
**START**  
**STATUS**  
**STOP**  
**STRING**  
**SUBSTITUTE\_FUNC**  
**SUBSTITUTE\_CASE\_FUNC**  
**SUBTRACT**  
**SUM**  
**SUPPRESS**  
**SYMBOLIC**  
**SYNCHRONIZED**  
**TALLYING**  
**TAPE**  
**TERMINATE**  
**TEST**



*THAN*  
*THEN*  
*THRU*  
*TIME*  
*TIMES*  
*TO*  
*TOK\_FALSE*  
*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*

**WAIT**  
**WHEN**  
**WHEN\_COMPILED\_FUNC**  
**WITH**  
**WORD**  
**WORDS**  
**WORKING\_STORAGE**  
**WRITE**  
**YYYYDDD**  
**YYYYMMDD**  
**ZERO**  
**UNARY\_SIGN**  
**TOKEN\_EOF**  
**ACCEPT**  
**ACCESS**  
**ADD**  
**ADDRESS**  
**ADVANCING**  
**AFTER**  
**ALL**  
**ALLOCATE**  
**ALPHABET**  
**ALPHABETIC**  
**ALPHABETIC\_LOWER**  
**ALPHABETIC\_UPPER**  
**ALPHANUMERIC**  
**ALPHANUMERIC\_EDITED**  
**ALSO**  
**ALTER**  
**ALTERNATE**  
**AND**  
**ANY**  
**ARE**  
**AREA**  
**ARGUMENT\_NUMBER**  
**ARGUMENT\_VALUE**  
**AS**  
**ASCENDING**

**ASSIGN**  
**AT**  
**AUTO**  
**AUTOMATIC**  
**BACKGROUND\_COLOR**  
**BASED**  
**BEFORE**  
**BELL**  
**BINARY**  
**BINARY\_C\_LONG**  
**BINARY\_CHAR**  
**BINARY\_DOUBLE**  
**BINARY\_LONG**  
**BINARY\_SHORT**  
**BLANK**  
**BLANK\_LINE**  
**BLANK\_SCREEN**  
**BLINK**  
**BLOCK**  
**BOTTOM**  
**BY**  
**BYTE\_LENGTH**  
**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**

**COMMAND\_LINE**  
**COMMA\_DELIM**  
**COMMIT**  
**COMMON**  
**COMP**  
**COMPUTE**  
**COMP\_1**  
**COMP\_2**  
**COMP\_3**  
**COMP\_4**  
**COMP\_5**  
**COMP\_X**  
**CONCATENATE\_FUNC**  
**CONFIGURATION**  
**CONSTANT**  
**CONTAINS**  
**CONTENT**  
**CONTINUE**  
**CONTROL**  
**CONTROLS**  
**CONTROL\_FOOTING**  
**CONTROL\_HEADING**  
**CONVERTING**  
**CORRESPONDING**  
**COUNT**  
**CRT**  
**CURRENCY**  
**CURRENT\_DATE\_FUNC**  
**CURSOR**  
**CYCLE**  
**DATA**  
**DATE**  
**DAY**  
**DAY\_OF\_WEEK**  
**DE**  
**DEBUGGING**  
**DECIMAL\_POINT**  
**DECLARATIVES**

*DEFAULT*  
*DELETE*  
*DELIMITED*  
*DELIMITER*  
*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*

*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*BACKGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*

*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*  
*GROUP*  
*HEADING*  
*HIGHLIGHT*  
*HIGH\_VALUE*  
*IDENTIFICATION*  
*IF*  
*IGNORE*  
*IGNORING*  
*IN*  
*INDEX*  
*INDEXED*  
*INDICATE*  
*INITIALIZE*  
*INITIALIZED*  
*INITIATE*  
*INPUT*  
*INPUT\_OUTPUT*  
*INSPECT*  
*INTO*  
*INTRINSIC*  
*INVALID*  
*INVALID\_KEY*  
*IS*  
*I\_O*  
*I\_O\_CONTROL*  
*JUSTIFIED*  
*KEY*  
*LABEL*  
*LAST*  
*LAST\_DETAIL*  
*LE*  
*LEADING*  
*LEFT*

*LENGTH*  
*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*



*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*  
*OR*  
*ORDER*  
*ORGANIZATION*  
*OTHER*  
*OUTPUT*  
*OVERFLOW*  
*OVERLINE*  
*PACKED\_DECIMAL*  
*PADDING*  
*PAGE*  
*PAGE\_FOOTING*  
*PAGE\_HEADING*  
*PARAGRAPH*  
*PERFORM*  
*PICTURE*  
*PLUS*  
*POINTER*  
*POSITION*  
*POSITIVE*  
*PRESENT*

*PREVIOUS*  
*PRINTER*  
*PRINTING*  
*PROCEDURE*  
*PROCEDURES*  
*PROCEED*  
*PROGRAM*  
*PROGRAM\_ID*  
*PROGRAM\_NAME*  
*PROGRAM\_POINTER*  
*PROMPT*  
*QUOTE*  
*RANDOM*  
*RD*  
*READ*  
*RECORD*  
*RECORDING*  
*RECORDS*  
*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*

*REVERSE\_FUNC*  
*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*  
*RIGHT*  
*ROLLBACK*  
*ROUNDED*  
*RUN*  
*SAME*  
*SCREEN*  
*SCREEN\_CONTROL*  
*SCROLL*  
*SD*  
*SEARCH*  
*SECTION*  
*SECURE*  
*SEGMENT\_LIMIT*  
*SELECT*  
*SEMI\_COLON*  
*SENTENCE*  
*SEPARATE*  
*SEQUENCE*  
*SEQUENTIAL*  
*SET*  
*SHARING*  
*SIGN*  
*SIGNED*  
*SIGNED\_INT*  
*SIGNED\_LONG*  
*SIGNED\_SHORT*  
*SIZE*  
*SIZE\_ERROR*  
*SORT*  
*SORT\_MERGE*  
*SOURCE*  
*SOURCE\_COMPUTER*  
*SPACE*  
*SPECIAL\_NAMES*

**STANDARD**  
**STANDARD\_1**  
**STANDARD\_2**  
**START**  
**STATUS**  
**STOP**  
**STRING**  
**SUBSTITUTE\_FUNC**  
**SUBSTITUTE\_CASE\_FUNC**  
**SUBTRACT**  
**SUM**  
**SUPPRESS**  
**SYMBOLIC**  
**SYNCHRONIZED**  
**TALLYING**  
**TAPE**  
**TERMINATE**  
**TEST**  
**THAN**  
**THEN**  
**THRU**  
**TIME**  
**TIMES**  
**TO**  
**TOK\_FALSE**  
**TOK\_FILE**  
**TOK\_INITIAL**  
**TOK\_NULL**  
**TOK\_TRUE**  
**TOP**  
**TRAILING**  
**TRANSFORM**  
**TRIM\_FUNCTION**  
**TYPE**  
**UNDERLINE**  
**UNIT**  
**UNLOCK**  
**UNSIGNED**

*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*  
*WORD*  
*WORDS*  
*WORKING\_STORAGE*  
*WRITE*  
*YYYYDDD*  
*YYYYMMDD*  
*ZERO*  
*UNARY\_SIGN*  
*TOKEN\_EOF*  
*COPY*  
*REPLACE*  
*SUPPRESS*  
*PRINTING*  
*REPLACING*  
*OFF*  
*IN*

**OF**  
**BY**  
**EQEQ**  
**TOKEN**  
**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**

Definition at line 55 of file parser.c.

```
{  
    TOKEN_EOF = 0,  
    ACCEPT = 258,  
    ACCESS = 259,  
    ADD = 260,  
    ADDRESS = 261,  
    ADVANCING = 262,  
    AFTER = 263,  
    ALL = 264,  
    ALLOCATE = 265,  
    ALPHABET = 266,  
    ALPHABETIC = 267,  
    ALPHABETIC_LOWER = 268,  
    ALPHABETIC_UPPER = 269,  
    ALPHANUMERIC = 270,  
    ALPHANUMERIC_EDITED = 271,  
    ALSO = 272,  
    ALTER = 273,  
    ALTERNATE = 274,  
    AND = 275,  
    ANY = 276,  
    ARE = 277,  
    AREA = 278,  
    ARGUMENT_NUMBER = 279,  
    ARGUMENT_VALUE = 280,  
    AS = 281,  
    ASCENDING = 282,  
    ASSIGN = 283,  
    AT = 284,  
    AUTO = 285,  
    AUTOMATIC = 286,  
    BACKGROUND_COLOR = 287,
```

```
BASED = 288,  
BEFORE = 289,  
BELL = 290,  
BINARY = 291,  
BINARY_C_LONG = 292,  
BINARY_CHAR = 293,  
BINARY_DOUBLE = 294,  
BINARY_LONG = 295,  
BINARY_SHORT = 296,  
BLANK = 297,  
BLANK_LINE = 298,  
BLANK_SCREEN = 299,  
BLINK = 300,  
BLOCK = 301,  
BOTTOM = 302,  
BY = 303,  
BYTE_LENGTH = 304,  
CALL = 305,  
CANCEL = 306,  
CH = 307,  
CHAINING = 308,  
CHARACTER = 309,  
CHARACTERS = 310,  
CLASS = 311,  
CLOSE = 312,  
CODE = 313,  
CODE_SET = 314,  
COLLATING = 315,  
COL = 316,  
COLS = 317,  
COLUMN = 318,  
COLUMNS = 319,  
COMMA = 320,  
COMMAND_LINE = 321,  
COMMA_DELIM = 322,  
COMMIT = 323,  
COMMON = 324,  
COMP = 325,  
COMPUTE = 326,  
COMP_1 = 327,  
COMP_2 = 328,  
COMP_3 = 329,  
COMP_4 = 330,  
COMP_5 = 331,  
COMP_X = 332,  
CONCATENATE_FUNC = 333,  
CONFIGURATION = 334,  
CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337,  
CONTINUE = 338,  
CONTROL = 339,  
CONTROLS = 340,  
CONTROL_FOOTING = 341,  
CONTROL_HEADING = 342,  
CONVERTING = 343,  
CORRESPONDING = 344,  
COUNT = 345,  
CRT = 346,  
CURRENCY = 347,  
CURRENT_DATE_FUNC = 348,  
CURSOR = 349,
```

```
CYCLE = 350,  
DATA = 351,  
DATE = 352,  
DAY = 353,  
DAY_OF_WEEK = 354,  
DE = 355,  
DEBUGGING = 356,  
DECIMAL_POINT = 357,  
DECLARATIVES = 358,  
DEFAULT = 359,  
DELETE = 360,  
DELIMITED = 361,  
DELIMITER = 362,  
DEPENDING = 363,  
DESCENDING = 364,  
DETAIL = 365,  
DISK = 366,  
DISPLAY = 367,  
DIVIDE = 368,  
DIVISION = 369,  
DOWN = 370,  
DUPLICATES = 371,  
DYNAMIC = 372,  
EBCDIC = 373,  
ELSE = 374,  
END = 375,  
END_ACCEPT = 376,  
END_ADD = 377,  
END_CALL = 378,  
END_COMPUTE = 379,  
END_DELETE = 380,  
END_DISPLAY = 381,  
END_DIVIDE = 382,  
END_EVALUATE = 383,  
END_FUNCTION = 384,  
END_IF = 385,  
END_MULTIPLY = 386,  
END_PERFORM = 387,  
END_PROGRAM = 388,  
END_READ = 389,  
END_RETURN = 390,  
END_REWRITE = 391,  
END_SEARCH = 392,  
END_START = 393,  
END_STRING = 394,  
END_SUBTRACT = 395,  
END_UNSTRING = 396,  
END_WRITE = 397,  
ENTRY = 398,  
ENVIRONMENT = 399,  
ENVIRONMENT_NAME = 400,  
ENVIRONMENT_VALUE = 401,  
EOL = 402,  
EOP = 403,  
EOS = 404,  
EQUAL = 405,  
EQUALS = 406,  
ERASE = 407,  
ERROR = 408,  
ESCAPE = 409,  
EVALUATE = 410,  
EVENT_STATUS = 411,
```



```
EXCEPTION = 412,  
EXCLUSIVE = 413,  
EXIT = 414,  
EXTEND = 415,  
EXTERNAL = 416,  
FD = 417,  
FILE_CONTROL = 418,  
FILE_ID = 419,  
FILLER = 420,  
FINAL = 421,  
FIRST = 422,  
FOOTING = 423,  
FOR = 424,  
FOREGROUND_COLOR = 425,  
FOREVER = 426,  
FREE = 427,  
FROM = 428,  
FULL = 429,  
FUNCTION = 430,  
FUNCTION_ID = 431,  
FUNCTION_NAME = 432,  
GE = 433,  
GENERATE = 434,  
GIVING = 435,  
GLOBAL = 436,  
GO = 437,  
GOBACK = 438,  
GREATER = 439,  
GROUP = 440,  
HEADING = 441,  
HIGHLIGHT = 442,  
HIGH_VALUE = 443,  
IDENTIFICATION = 444,  
IF = 445,  
IGNORE = 446,  
IGNORING = 447,  
IN = 448,  
INDEX = 449,  
INDEXED = 450,  
INDICATE = 451,  
INITIALIZE = 452,  
INITIALIZED = 453,  
INITIATE = 454,  
INPUT = 455,  
INPUT_OUTPUT = 456,  
INSPECT = 457,  
INTO = 458,  
INTRINSIC = 459,  
INVALID = 460,  
INVALID_KEY = 461,  
IS = 462,  
I_O = 463,  
I_O_CONTROL = 464,  
JUSTIFIED = 465,  
KEY = 466,  
LABEL = 467,  
LAST = 468,  
LAST_DETAIL = 469,  
LE = 470,  
LEADING = 471,  
LEFT = 472,  
LENGTH = 473,
```

```
LESS = 474,  
LIMIT = 475,  
LIMITS = 476,  
LINAGE = 477,  
LINAGE_COUNTER = 478,  
LINE = 479,  
LINES = 480,  
LINKAGE = 481,  
LITERAL = 482,  
LOCALE = 483,  
LOCALE_DT_FUNC = 484,  
LOCAL_STORAGE = 485,  
LOCK = 486,  
LOWER_CASE_FUNC = 487,  
LOWLIGHT = 488,  
LOW_VALUE = 489,  
MANUAL = 490,  
MEMORY = 491,  
MERGE = 492,  
MINUS = 493,  
MNEMONIC_NAME = 494,  
MODE = 495,  
MOVE = 496,  
MULTIPLE = 497,  
MULTIPLY = 498,  
NATIONAL = 499,  
NATIONAL_EDITED = 500,  
NATIVE = 501,  
NE = 502,  
NEGATIVE = 503,  
NEXT = 504,  
NEXT_SENTENCE = 505,  
NO = 506,  
NOT = 507,  
NOT_END = 508,  
NOT_EOP = 509,  
NOT_EXCEPTION = 510,  
NOT_INVALID_KEY = 511,  
NOT_OVERFLOW = 512,  
NOT_SIZE_ERROR = 513,  
NO_ADVANCING = 514,  
NUMBER = 515,  
NUMBERS = 516,  
NUMERIC = 517,  
NUMERIC_EDITED = 518,  
NUMVALC_FUNC = 519,  
OBJECT_COMPUTER = 520,  
OCCURS = 521,  
OF = 522,  
OFF = 523,  
OMITTED = 524,  
ON = 525,  
ONLY = 526,  
OPEN = 527,  
OPTIONAL = 528,  
OR = 529,  
ORDER = 530,  
ORGANIZATION = 531,  
OTHER = 532,  
OUTPUT = 533,  
OVERFLOW = 534,  
OVERLINE = 535,
```

PACKED\_DECIMAL = 536,  
PADDING = 537,  
PAGE = 538,  
PAGE\_FOOTING = 539,  
PAGE\_HEADING = 540,  
PARAGRAPH = 541,  
PERFORM = 542,  
PICTURE = 543,  
PLUS = 544,  
POINTER = 545,  
POSITION = 546,  
POSITIVE = 547,  
PRESENT = 548,  
PREVIOUS = 549,  
PRINTER = 550,  
PRINTING = 551,  
PROCEDURE = 552,  
PROCEDURES = 553,  
PROCEED = 554,  
PROGRAM = 555,  
PROGRAM\_ID = 556,  
PROGRAM\_NAME = 557,  
PROGRAM\_POINTER = 558,  
PROMPT = 559,  
QUOTE = 560,  
RANDOM = 561,  
RD = 562,  
READ = 563,  
RECORD = 564,  
RECORDING = 565,  
RECORDS = 566,  
RECURSIVE = 567,  
REDEFINES = 568,  
REEL = 569,  
REFERENCE = 570,  
RELATIVE = 571,  
RELEASE = 572,  
REMAINDER = 573,  
REMOVAL = 574,  
RENAMES = 575,  
REPLACING = 576,  
REPORT = 577,  
REPORTING = 578,  
REPORTS = 579,  
REPORT\_FOOTING = 580,  
REPORT\_HEADING = 581,  
REPOSITORY = 582,  
REQUIRED = 583,  
RESERVE = 584,  
RETURN = 585,  
RETURNING = 586,  
REVERSE\_FUNC = 587,  
REVERSE\_VIDEO = 588,  
REWIND = 589,  
REWRITE = 590,  
RIGHT = 591,  
ROLLBACK = 592,  
ROUNDED = 593,  
RUN = 594,  
SAME = 595,  
SCREEN = 596,  
SCREEN\_CONTROL = 597,

```
SCROLL = 598,  
SD = 599,  
SEARCH = 600,  
SECTION = 601,  
SECURE = 602,  
SEGMENT_LIMIT = 603,  
SELECT = 604,  
SEMI_COLON = 605,  
SENTENCE = 606,  
SEPARATE = 607,  
SEQUENCE = 608,  
SEQUENTIAL = 609,  
SET = 610,  
SHARING = 611,  
SIGN = 612,  
SIGNED = 613,  
SIGNED_INT = 614,  
SIGNED_LONG = 615,  
SIGNED_SHORT = 616,  
SIZE = 617,  
SIZE_ERROR = 618,  
SORT = 619,  
SORT_MERGE = 620,  
SOURCE = 621,  
SOURCE_COMPUTER = 622,  
SPACE = 623,  
SPECIAL_NAMES = 624,  
STANDARD = 625,  
STANDARD_1 = 626,  
STANDARD_2 = 627,  
START = 628,  
STATUS = 629,  
STOP = 630,  
STRING = 631,  
SUBSTITUTE_FUNC = 632,  
SUBSTITUTE_CASE_FUNC = 633,  
SUBTRACT = 634,  
SUM = 635,  
SUPPRESS = 636,  
SYMBOLIC = 637,  
SYNCHRONIZED = 638,  
TALLYING = 639,  
TAPE = 640,  
TERMINATE = 641,  
TEST = 642,  
THAN = 643,  
THEN = 644,  
THRU = 645,  
TIME = 646,  
TIMES = 647,  
TO = 648,  
TOK_FALSE = 649,  
TOK_FILE = 650,  
TOK_INITIAL = 651,  
TOK_NULL = 652,  
TOK_TRUE = 653,  
TOP = 654,  
TRAILING = 655,  
TRANSFORM = 656,  
TRIM_FUNCTION = 657,  
TYPE = 658,  
UNDERLINE = 659,
```

```
UNIT = 660,  
UNLOCK = 661,  
UNSIGNED = 662,  
UNSIGNED_INT = 663,  
UNSIGNED_LONG = 664,  
UNSIGNED_SHORT = 665,  
UNSTRING = 666,  
UNTIL = 667,  
UP = 668,  
UPDATE = 669,  
UPON = 670,  
UPON_ARGUMENT_NUMBER = 671,  
UPON_COMMAND_LINE = 672,  
UPON_ENVIRONMENT_NAME = 673,  
UPON_ENVIRONMENT_VALUE = 674,  
UPPER_CASE_FUNC = 675,  
USAGE = 676,  
USE = 677,  
USING = 678,  
VALUE = 679,  
VARYING = 680,  
WAIT = 681,  
WHEN = 682,  
WHEN_COMPILED_FUNC = 683,  
WITH = 684,  
WORD = 685,  
WORDS = 686,  
WORKING_STORAGE = 687,  
WRITE = 688,  
YYYYDDD = 689,  
YYYYMMDD = 690,  
ZERO = 691,  
UNARY_SIGN = 692  
};
```

## 7.8.4 Function Documentation

### 7.8.4.1 int yyparse ( void )

## 7.8.5 Variable Documentation

### 7.8.5.1 struct cb\_label\* current\_paragraph = NULL

Definition at line 986 of file parser.c.

### 7.8.5.2 struct cb\_program\* current\_program = NULL

Definition at line 983 of file parser.c.

### 7.8.5.3 struct cb\_label\* current\_section = NULL

Definition at line 985 of file parser.c.

**7.8.5.4** `struct cb_statement* current_statement = NULL`

Definition at line 984 of file parser.c.

**7.8.5.5** `size_t functions_are_all = 0`

Definition at line 987 of file parser.c.

**7.8.5.6** `int non_const_word = 0`

Definition at line 988 of file parser.c.

**7.8.5.7** `int yychar`

Definition at line 5078 of file parser.c.

**7.8.5.8** `int yydebug`

Definition at line 4908 of file parser.c.

**7.8.5.9** `YYSTYPE yylval`

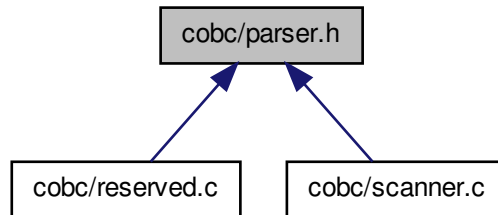
Definition at line 5081 of file parser.c.

**7.8.5.10** `int yynerrs`

Definition at line 5084 of file parser.c.

## 7.9 cobc/parser.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define [TOKEN\\_EOF](#) 0
- #define [ACCEPT](#) 258
- #define [ACCESS](#) 259
- #define [ADD](#) 260
- #define [ADDRESS](#) 261
- #define [ADVANCING](#) 262
- #define [AFTER](#) 263
- #define [ALL](#) 264
- #define [ALLOCATE](#) 265
- #define [ALPHABET](#) 266
- #define [ALPHABETIC](#) 267
- #define [ALPHABETIC\\_LOWER](#) 268
- #define [ALPHABETIC\\_UPPER](#) 269
- #define [ALPHANUMERIC](#) 270
- #define [ALPHANUMERIC\\_EDITED](#) 271
- #define [ALSO](#) 272
- #define [ALTER](#) 273
- #define [ALTERNATE](#) 274
- #define [AND](#) 275
- #define [ANY](#) 276
- #define [ARE](#) 277
- #define [AREA](#) 278
- #define [ARGUMENT\\_NUMBER](#) 279
- #define [ARGUMENT\\_VALUE](#) 280
- #define [AS](#) 281

- #define [ASCENDING](#) 282
- #define [ASSIGN](#) 283
- #define [AT](#) 284
- #define [AUTO](#) 285
- #define [AUTOMATIC](#) 286
- #define [BACKGROUND\\_COLOR](#) 287
- #define [BASED](#) 288
- #define [BEFORE](#) 289
- #define [BELL](#) 290
- #define [BINARY](#) 291
- #define [BINARY\\_C\\_LONG](#) 292
- #define [BINARY\\_CHAR](#) 293
- #define [BINARY\\_DOUBLE](#) 294
- #define [BINARY\\_LONG](#) 295
- #define [BINARY\\_SHORT](#) 296
- #define [BLANK](#) 297
- #define [BLANK\\_LINE](#) 298
- #define [BLANK\\_SCREEN](#) 299
- #define [BLINK](#) 300
- #define [BLOCK](#) 301
- #define [BOTTOM](#) 302
- #define [BY](#) 303
- #define [BYTE\\_LENGTH](#) 304
- #define [CALL](#) 305
- #define [CANCEL](#) 306
- #define [CH](#) 307
- #define [CHAINING](#) 308
- #define [CHARACTER](#) 309
- #define [CHARACTERS](#) 310
- #define [CLASS](#) 311
- #define [CLOSE](#) 312
- #define [CODE](#) 313
- #define [CODE\\_SET](#) 314
- #define [COLLATING](#) 315
- #define [COL](#) 316
- #define [COLS](#) 317
- #define [COLUMN](#) 318
- #define [COLUMNS](#) 319
- #define [COMMA](#) 320
- #define [COMMAND\\_LINE](#) 321
- #define [COMMA\\_DELIM](#) 322
- #define [COMMIT](#) 323
- #define [COMMON](#) 324
- #define [COMP](#) 325
- #define [COMPUTE](#) 326
- #define [COMP\\_1](#) 327



- #define [COMP\\_2](#) 328
- #define [COMP\\_3](#) 329
- #define [COMP\\_4](#) 330
- #define [COMP\\_5](#) 331
- #define [COMP\\_X](#) 332
- #define [CONCATENATE\\_FUNC](#) 333
- #define [CONFIGURATION](#) 334
- #define [CONSTANT](#) 335
- #define [CONTAINS](#) 336
- #define [CONTENT](#) 337
- #define [CONTINUE](#) 338
- #define [CONTROL](#) 339
- #define [CONTROLS](#) 340
- #define [CONTROL\\_FOOTING](#) 341
- #define [CONTROL\\_HEADING](#) 342
- #define [CONVERTING](#) 343
- #define [CORRESPONDING](#) 344
- #define [COUNT](#) 345
- #define [CRT](#) 346
- #define [CURRENCY](#) 347
- #define [CURRENT\\_DATE\\_FUNC](#) 348
- #define [CURSOR](#) 349
- #define [CYCLE](#) 350
- #define [DATA](#) 351
- #define [DATE](#) 352
- #define [DAY](#) 353
- #define [DAY\\_OF\\_WEEK](#) 354
- #define [DE](#) 355
- #define [DEBUGGING](#) 356
- #define [DECIMAL\\_POINT](#) 357
- #define [DECLARATIVES](#) 358
- #define [DEFAULT](#) 359
- #define [DELETE](#) 360
- #define [DELIMITED](#) 361
- #define [DELIMITER](#) 362
- #define [DEPENDING](#) 363
- #define [DESCENDING](#) 364
- #define [DETAIL](#) 365
- #define [DISK](#) 366
- #define [DISPLAY](#) 367
- #define [DIVIDE](#) 368
- #define [DIVISION](#) 369
- #define [DOWN](#) 370
- #define [DUPLICATES](#) 371
- #define [DYNAMIC](#) 372
- #define [EBCDIC](#) 373

- #define [ELSE](#) 374
- #define [END](#) 375
- #define [END\\_ACCEPT](#) 376
- #define [END\\_ADD](#) 377
- #define [END\\_CALL](#) 378
- #define [END\\_COMPUTE](#) 379
- #define [END\\_DELETE](#) 380
- #define [END\\_DISPLAY](#) 381
- #define [END\\_DIVIDE](#) 382
- #define [END\\_EVALUATE](#) 383
- #define [END\\_FUNCTION](#) 384
- #define [END\\_IF](#) 385
- #define [END\\_MULTIPLY](#) 386
- #define [END\\_PERFORM](#) 387
- #define [END\\_PROGRAM](#) 388
- #define [END\\_READ](#) 389
- #define [END\\_RETURN](#) 390
- #define [END\\_REWRITE](#) 391
- #define [END\\_SEARCH](#) 392
- #define [END\\_START](#) 393
- #define [END\\_STRING](#) 394
- #define [END\\_SUBTRACT](#) 395
- #define [END\\_UNSTRING](#) 396
- #define [END\\_WRITE](#) 397
- #define [ENTRY](#) 398
- #define [ENVIRONMENT](#) 399
- #define [ENVIRONMENT\\_NAME](#) 400
- #define [ENVIRONMENT\\_VALUE](#) 401
- #define [EOL](#) 402
- #define [EOP](#) 403
- #define [EOS](#) 404
- #define [EQUAL](#) 405
- #define [EQUALS](#) 406
- #define [ERASE](#) 407
- #define [ERROR](#) 408
- #define [ESCAPE](#) 409
- #define [EVALUATE](#) 410
- #define [EVENT\\_STATUS](#) 411
- #define [EXCEPTION](#) 412
- #define [EXCLUSIVE](#) 413
- #define [EXIT](#) 414
- #define [EXTEND](#) 415
- #define [EXTERNAL](#) 416
- #define [FD](#) 417
- #define [FILE\\_CONTROL](#) 418
- #define [FILE\\_ID](#) 419

- #define [FILLER](#) 420
- #define [FINAL](#) 421
- #define [FIRST](#) 422
- #define [FOOTING](#) 423
- #define [FOR](#) 424
- #define [FOREGROUND\\_COLOR](#) 425
- #define [FOREVER](#) 426
- #define [FREE](#) 427
- #define [FROM](#) 428
- #define [FULL](#) 429
- #define [FUNCTION](#) 430
- #define [FUNCTION\\_ID](#) 431
- #define [FUNCTION\\_NAME](#) 432
- #define [GE](#) 433
- #define [GENERATE](#) 434
- #define [GIVING](#) 435
- #define [GLOBAL](#) 436
- #define [GO](#) 437
- #define [GOBACK](#) 438
- #define [GREATER](#) 439
- #define [GROUP](#) 440
- #define [HEADING](#) 441
- #define [HIGHLIGHT](#) 442
- #define [HIGH\\_VALUE](#) 443
- #define [IDENTIFICATION](#) 444
- #define [IF](#) 445
- #define [IGNORE](#) 446
- #define [IGNORING](#) 447
- #define [IN](#) 448
- #define [INDEX](#) 449
- #define [INDEXED](#) 450
- #define [INDICATE](#) 451
- #define [INITIALIZE](#) 452
- #define [INITIALIZED](#) 453
- #define [INITIATE](#) 454
- #define [INPUT](#) 455
- #define [INPUT\\_OUTPUT](#) 456
- #define [INSPECT](#) 457
- #define [INTO](#) 458
- #define [INTRINSIC](#) 459
- #define [INVALID](#) 460
- #define [INVALID\\_KEY](#) 461
- #define [IS](#) 462
- #define [I\\_O](#) 463
- #define [I\\_O\\_CONTROL](#) 464
- #define [JUSTIFIED](#) 465

- #define [KEY](#) 466
- #define [LABEL](#) 467
- #define [LAST](#) 468
- #define [LAST\\_DETAIL](#) 469
- #define [LE](#) 470
- #define [LEADING](#) 471
- #define [LEFT](#) 472
- #define [LENGTH](#) 473
- #define [LESS](#) 474
- #define [LIMIT](#) 475
- #define [LIMITS](#) 476
- #define [LINAGE](#) 477
- #define [LINAGE\\_COUNTER](#) 478
- #define [LINE](#) 479
- #define [LINES](#) 480
- #define [LINKAGE](#) 481
- #define [LITERAL](#) 482
- #define [LOCALE](#) 483
- #define [LOCALE\\_DT\\_FUNC](#) 484
- #define [LOCAL\\_STORAGE](#) 485
- #define [LOCK](#) 486
- #define [LOWER\\_CASE\\_FUNC](#) 487
- #define [LOWLIGHT](#) 488
- #define [LOW\\_VALUE](#) 489
- #define [MANUAL](#) 490
- #define [MEMORY](#) 491
- #define [MERGE](#) 492
- #define [MINUS](#) 493
- #define [MNEMONIC\\_NAME](#) 494
- #define [MODE](#) 495
- #define [MOVE](#) 496
- #define [MULTIPLE](#) 497
- #define [MULTIPLY](#) 498
- #define [NATIONAL](#) 499
- #define [NATIONAL\\_EDITED](#) 500
- #define [NATIVE](#) 501
- #define [NE](#) 502
- #define [NEGATIVE](#) 503
- #define [NEXT](#) 504
- #define [NEXT\\_SENTENCE](#) 505
- #define [NO](#) 506
- #define [NOT](#) 507
- #define [NOT\\_END](#) 508
- #define [NOT\\_EOP](#) 509
- #define [NOT\\_EXCEPTION](#) 510
- #define [NOT\\_INVALID\\_KEY](#) 511

- #define [NOT\\_OVERFLOW](#) 512
- #define [NOT\\_SIZE\\_ERROR](#) 513
- #define [NO\\_ADVANCING](#) 514
- #define [NUMBER](#) 515
- #define [NUMBERS](#) 516
- #define [NUMERIC](#) 517
- #define [NUMERIC\\_EDITED](#) 518
- #define [NUMVALC\\_FUNC](#) 519
- #define [OBJECT\\_COMPUTER](#) 520
- #define [OCCURS](#) 521
- #define [OF](#) 522
- #define [OFF](#) 523
- #define [OMITTED](#) 524
- #define [ON](#) 525
- #define [ONLY](#) 526
- #define [OPEN](#) 527
- #define [OPTIONAL](#) 528
- #define [OR](#) 529
- #define [ORDER](#) 530
- #define [ORGANIZATION](#) 531
- #define [OTHER](#) 532
- #define [OUTPUT](#) 533
- #define [OVERFLOW](#) 534
- #define [OVERLINE](#) 535
- #define [PACKED\\_DECIMAL](#) 536
- #define [PADDING](#) 537
- #define [PAGE](#) 538
- #define [PAGE\\_FOOTING](#) 539
- #define [PAGE\\_HEADING](#) 540
- #define [PARAGRAPH](#) 541
- #define [PERFORM](#) 542
- #define [PICTURE](#) 543
- #define [PLUS](#) 544
- #define [POINTER](#) 545
- #define [POSITION](#) 546
- #define [POSITIVE](#) 547
- #define [PRESENT](#) 548
- #define [PREVIOUS](#) 549
- #define [PRINTER](#) 550
- #define [PRINTING](#) 551
- #define [PROCEDURE](#) 552
- #define [PROCEDURES](#) 553
- #define [PROCEED](#) 554
- #define [PROGRAM](#) 555
- #define [PROGRAM\\_ID](#) 556
- #define [PROGRAM\\_NAME](#) 557

- #define [PROGRAM\\_POINTER](#) 558
- #define [PROMPT](#) 559
- #define [QUOTE](#) 560
- #define [RANDOM](#) 561
- #define [RD](#) 562
- #define [READ](#) 563
- #define [RECORD](#) 564
- #define [RECORDING](#) 565
- #define [RECORDS](#) 566
- #define [RECURSIVE](#) 567
- #define [REDEFINES](#) 568
- #define [REEL](#) 569
- #define [REFERENCE](#) 570
- #define [RELATIVE](#) 571
- #define [RELEASE](#) 572
- #define [REMAINDER](#) 573
- #define [REMOVAL](#) 574
- #define [RENAMES](#) 575
- #define [REPLACING](#) 576
- #define [REPORT](#) 577
- #define [REPORTING](#) 578
- #define [REPORTS](#) 579
- #define [REPORT\\_FOOTING](#) 580
- #define [REPORT\\_HEADING](#) 581
- #define [REPOSITORY](#) 582
- #define [REQUIRED](#) 583
- #define [RESERVE](#) 584
- #define [RETURN](#) 585
- #define [RETURNING](#) 586
- #define [REVERSE\\_FUNC](#) 587
- #define [REVERSE\\_VIDEO](#) 588
- #define [REWIND](#) 589
- #define [REWRITE](#) 590
- #define [RIGHT](#) 591
- #define [ROLLBACK](#) 592
- #define [ROUNDED](#) 593
- #define [RUN](#) 594
- #define [SAME](#) 595
- #define [SCREEN](#) 596
- #define [SCREEN\\_CONTROL](#) 597
- #define [SCROLL](#) 598
- #define [SD](#) 599
- #define [SEARCH](#) 600
- #define [SECTION](#) 601
- #define [SECURE](#) 602
- #define [SEGMENT\\_LIMIT](#) 603

- #define [SELECT](#) 604
- #define [SEMI\\_COLON](#) 605
- #define [SENTENCE](#) 606
- #define [SEPARATE](#) 607
- #define [SEQUENCE](#) 608
- #define [SEQUENTIAL](#) 609
- #define [SET](#) 610
- #define [SHARING](#) 611
- #define [SIGN](#) 612
- #define [SIGNED](#) 613
- #define [SIGNED\\_INT](#) 614
- #define [SIGNED\\_LONG](#) 615
- #define [SIGNED\\_SHORT](#) 616
- #define [SIZE](#) 617
- #define [SIZE\\_ERROR](#) 618
- #define [SORT](#) 619
- #define [SORT\\_MERGE](#) 620
- #define [SOURCE](#) 621
- #define [SOURCE\\_COMPUTER](#) 622
- #define [SPACE](#) 623
- #define [SPECIAL\\_NAMES](#) 624
- #define [STANDARD](#) 625
- #define [STANDARD\\_1](#) 626
- #define [STANDARD\\_2](#) 627
- #define [START](#) 628
- #define [STATUS](#) 629
- #define [STOP](#) 630
- #define [STRING](#) 631
- #define [SUBSTITUTE\\_FUNC](#) 632
- #define [SUBSTITUTE\\_CASE\\_FUNC](#) 633
- #define [SUBTRACT](#) 634
- #define [SUM](#) 635
- #define [SUPPRESS](#) 636
- #define [SYMBOLIC](#) 637
- #define [SYNCHRONIZED](#) 638
- #define [TALLYING](#) 639
- #define [TAPE](#) 640
- #define [TERMINATE](#) 641
- #define [TEST](#) 642
- #define [THAN](#) 643
- #define [THEN](#) 644
- #define [THRU](#) 645
- #define [TIME](#) 646
- #define [TIMES](#) 647
- #define [TO](#) 648
- #define [TOK\\_FALSE](#) 649

- #define [TOK\\_FILE](#) 650
- #define [TOK\\_INITIAL](#) 651
- #define [TOK\\_NULL](#) 652
- #define [TOK\\_TRUE](#) 653
- #define [TOP](#) 654
- #define [TRAILING](#) 655
- #define [TRANSFORM](#) 656
- #define [TRIM\\_FUNCTION](#) 657
- #define [TYPE](#) 658
- #define [UNDERLINE](#) 659
- #define [UNIT](#) 660
- #define [UNLOCK](#) 661
- #define [UNSIGNED](#) 662
- #define [UNSIGNED\\_INT](#) 663
- #define [UNSIGNED\\_LONG](#) 664
- #define [UNSIGNED\\_SHORT](#) 665
- #define [UNSTRING](#) 666
- #define [UNTIL](#) 667
- #define [UP](#) 668
- #define [UPDATE](#) 669
- #define [UPON](#) 670
- #define [UPON\\_ARGUMENT\\_NUMBER](#) 671
- #define [UPON\\_COMMAND\\_LINE](#) 672
- #define [UPON\\_ENVIRONMENT\\_NAME](#) 673
- #define [UPON\\_ENVIRONMENT\\_VALUE](#) 674
- #define [UPPER\\_CASE\\_FUNC](#) 675
- #define [USAGE](#) 676
- #define [USE](#) 677
- #define [USING](#) 678
- #define [VALUE](#) 679
- #define [VARYING](#) 680
- #define [WAIT](#) 681
- #define [WHEN](#) 682
- #define [WHEN\\_COMPILED\\_FUNC](#) 683
- #define [WITH](#) 684
- #define [WORD](#) 685
- #define [WORDS](#) 686
- #define [WORKING\\_STORAGE](#) 687
- #define [WRITE](#) 688
- #define [YYYYDDD](#) 689
- #define [YYYYMMDD](#) 690
- #define [ZERO](#) 691
- #define [UNARY\\_SIGN](#) 692
- #define [ystype](#) YYSTYPE
- #define [YYSTYPE\\_IS\\_DECLARED](#) 1
- #define [YYSTYPE\\_IS\\_TRIVIAL](#) 1



## Typedefs

- typedef int [YYSTYPE](#)

## Enumerations

- enum [yytokentype](#) {  
[TOKEN\\_EOF](#) = 0, [ACCEPT](#) = 258, [ACCESS](#) = 259, [ADD](#) = 260,  
[ADDRESS](#) = 261, [ADVANCING](#) = 262, [AFTER](#) = 263, [ALL](#) = 264,  
[ALLOCATE](#) = 265, [ALPHABET](#) = 266, [ALPHABETIC](#) = 267, [ALPHABETIC\\_](#)  
[LOWER](#) = 268,  
[ALPHABETIC\\_UPPER](#) = 269, [ALPHANUMERIC](#) = 270, [ALPHANUMERIC\\_EDITED](#)  
= 271, [ALSO](#) = 272,  
[ALTER](#) = 273, [ALTERNATE](#) = 274, [AND](#) = 275, [ANY](#) = 276,  
[ARE](#) = 277, [AREA](#) = 278, [ARGUMENT\\_NUMBER](#) = 279, [ARGUMENT\\_VALUE](#)  
= 280,  
[AS](#) = 281, [ASCENDING](#) = 282, [ASSIGN](#) = 283, [AT](#) = 284,  
[AUTO](#) = 285, [AUTOMATIC](#) = 286, [BACKGROUND\\_COLOR](#) = 287, [BASED](#) = 288,  
[BEFORE](#) = 289, [BELL](#) = 290, [BINARY](#) = 291, [BINARY\\_C\\_LONG](#) = 292,  
[BINARY\\_CHAR](#) = 293, [BINARY\\_DOUBLE](#) = 294, [BINARY\\_LONG](#) = 295, [BINARY\\_](#)  
[SHORT](#) = 296,  
[BLANK](#) = 297, [BLANK\\_LINE](#) = 298, [BLANK\\_SCREEN](#) = 299, [BLINK](#) = 300,  
[BLOCK](#) = 301, [BOTTOM](#) = 302, [BY](#) = 303, [BYTE\\_LENGTH](#) = 304,  
[CALL](#) = 305, [CANCEL](#) = 306, [CH](#) = 307, [CHAINING](#) = 308,  
[CHARACTER](#) = 309, [CHARACTERS](#) = 310, [CLASS](#) = 311, [CLOSE](#) = 312,  
[CODE](#) = 313, [CODE\\_SET](#) = 314, [COLLATING](#) = 315, [COL](#) = 316,  
[COLS](#) = 317, [COLUMN](#) = 318, [COLUMNS](#) = 319, [COMMA](#) = 320,  
[COMMAND\\_LINE](#) = 321, [COMMA\\_DELIM](#) = 322, [COMMIT](#) = 323, [COMMON](#) =  
324,  
[COMP](#) = 325, [COMPUTE](#) = 326, [COMP\\_1](#) = 327, [COMP\\_2](#) = 328,  
[COMP\\_3](#) = 329, [COMP\\_4](#) = 330, [COMP\\_5](#) = 331, [COMP\\_X](#) = 332,  
[CONCATENATE\\_FUNC](#) = 333, [CONFIGURATION](#) = 334, [CONSTANT](#) = 335,  
[CONTAINS](#) = 336,  
[CONTENT](#) = 337, [CONTINUE](#) = 338, [CONTROL](#) = 339, [CONTROLS](#) = 340,  
[CONTROL\\_FOOTING](#) = 341, [CONTROL\\_HEADING](#) = 342, [CONVERTING](#) =  
343, [CORRESPONDING](#) = 344,  
[COUNT](#) = 345, [CRT](#) = 346, [CURRENCY](#) = 347, [CURRENT\\_DATE\\_FUNC](#) = 348,  
[CURSOR](#) = 349, [CYCLE](#) = 350, [DATA](#) = 351, [DATE](#) = 352,  
[DAY](#) = 353, [DAY\\_OF\\_WEEK](#) = 354, [DE](#) = 355, [DEBUGGING](#) = 356,  
[DECIMAL\\_POINT](#) = 357, [DECLARATIVES](#) = 358, [DEFAULT](#) = 359, [DELETE](#) =  
360,

DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING = 364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE = 380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_-  
FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM  
= 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH  
= 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING  
= 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_-  
NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION =  
444,  
IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,

LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOWLIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED = 500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUMBERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL = 536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID = 556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE = 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,  
REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING = 580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE = 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,

SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT  
= 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
COMMAND\_LINE = 672,  
UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
UPPER\_CASE\_FUNC = 675, USAGE = 676,  
USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
TOKEN\_EOF = 0, ACCEPT = 258, ACCESS = 259, ADD = 260,  
ADDRESS = 261, ADVANCING = 262, AFTER = 263, ALL = 264,  
ALLOCATE = 265, ALPHABET = 266, ALPHABETIC = 267, ALPHABETIC\_  
LOWER = 268,  
ALPHABETIC\_UPPER = 269, ALPHANUMERIC = 270, ALPHANUMERIC\_EDITED  
= 271, ALSO = 272,  
ALTER = 273, ALTERNATE = 274, AND = 275, ANY = 276,  
ARE = 277, AREA = 278, ARGUMENT\_NUMBER = 279, ARGUMENT\_VALUE  
= 280,  
AS = 281, ASCENDING = 282, ASSIGN = 283, AT = 284,  
AUTO = 285, AUTOMATIC = 286, BACKGROUND\_COLOR = 287, BASED = 288,

BEFORE = 289, BELL = 290, BINARY = 291, BINARY\_C\_LONG = 292,  
BINARY\_CHAR = 293, BINARY\_DOUBLE = 294, BINARY\_LONG = 295, BINARY\_SHORT = 296,  
BLANK = 297, BLANK\_LINE = 298, BLANK\_SCREEN = 299, BLINK = 300,  
BLOCK = 301, BOTTOM = 302, BY = 303, BYTE\_LENGTH = 304,  
CALL = 305, CANCEL = 306, CH = 307, CHAINING = 308,  
CHARACTER = 309, CHARACTERS = 310, CLASS = 311, CLOSE = 312,  
CODE = 313, CODE\_SET = 314, COLLATING = 315, COL = 316,  
COLS = 317, COLUMN = 318, COLUMNS = 319, COMMA = 320,  
COMMAND\_LINE = 321, COMMA\_DELIM = 322, COMMIT = 323, COMMON = 324,  
COMP = 325, COMPUTE = 326, COMP\_1 = 327, COMP\_2 = 328,  
COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING = 343,  
CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE = 360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING = 364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE = 380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM = 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH = 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING = 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,

EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION =  
444,  
IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOW-  
LIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED =  
500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_-  
OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUM-  
BERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_-  
COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL =  
536,

PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID =  
556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE  
= 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,  
REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING =  
580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE  
= 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_  
VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT  
= 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,

UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
 = 664,  
 UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
 UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
 COMMAND\_LINE = 672,  
 UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
 UPPER\_CASE\_FUNC = 675, USAGE = 676,  
 USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
 WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
 WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
 YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
 TOKEN\_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,  
 PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,  
 OF = 265, BY = 266, EQEQ = 267, TOKEN = 268,  
 TOKEN\_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,  
 PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,  
 OF = 265, BY = 266, EQEQ = 267, TOKEN = 268 }

## Variables

- [YYSTYPE yylval](#)

### 7.9.1 Define Documentation

#### 7.9.1.1 #define ACCEPT 258

Definition at line 471 of file parser.h.

#### 7.9.1.2 #define ACCESS 259

Definition at line 472 of file parser.h.

#### 7.9.1.3 #define ADD 260

Definition at line 473 of file parser.h.

#### 7.9.1.4 #define ADDRESS 261

Definition at line 474 of file parser.h.



**7.9.1.5 #define ADVANCING 262**

Definition at line 475 of file parser.h.

**7.9.1.6 #define AFTER 263**

Definition at line 476 of file parser.h.

**7.9.1.7 #define ALL 264**

Definition at line 477 of file parser.h.

**7.9.1.8 #define ALLOCATE 265**

Definition at line 478 of file parser.h.

**7.9.1.9 #define ALPHABET 266**

Definition at line 479 of file parser.h.

**7.9.1.10 #define ALPHABETIC 267**

Definition at line 480 of file parser.h.

**7.9.1.11 #define ALPHABETIC\_LOWER 268**

Definition at line 481 of file parser.h.

**7.9.1.12 #define ALPHABETIC\_UPPER 269**

Definition at line 482 of file parser.h.

**7.9.1.13 #define ALPHANUMERIC 270**

Definition at line 483 of file parser.h.

**7.9.1.14 #define ALPHANUMERIC\_EDITED 271**

Definition at line 484 of file parser.h.

**7.9.1.15 #define ALSO 272**

Definition at line 485 of file parser.h.

**7.9.1.16 #define ALTER 273**

Definition at line 486 of file parser.h.

**7.9.1.17 #define ALTERNATE 274**

Definition at line 487 of file parser.h.

**7.9.1.18 #define AND 275**

Definition at line 488 of file parser.h.

**7.9.1.19 #define ANY 276**

Definition at line 489 of file parser.h.

**7.9.1.20 #define ARE 277**

Definition at line 490 of file parser.h.

**7.9.1.21 #define AREA 278**

Definition at line 491 of file parser.h.

**7.9.1.22 #define ARGUMENT\_NUMBER 279**

Definition at line 492 of file parser.h.

**7.9.1.23 #define ARGUMENT\_VALUE 280**

Definition at line 493 of file parser.h.

**7.9.1.24 #define AS 281**

Definition at line 494 of file parser.h.

**7.9.1.25 #define ASCENDING 282**

Definition at line 495 of file parser.h.

**7.9.1.26 #define ASSIGN 283**

Definition at line 496 of file parser.h.

**7.9.1.27 #define AT 284**

Definition at line 497 of file parser.h.

**7.9.1.28 #define AUTO 285**

Definition at line 498 of file parser.h.

**7.9.1.29 #define AUTOMATIC 286**

Definition at line 499 of file parser.h.

**7.9.1.30 #define BACKGROUND\_COLOR 287**

Definition at line 500 of file parser.h.

**7.9.1.31 #define BASED 288**

Definition at line 501 of file parser.h.

**7.9.1.32 #define BEFORE 289**

Definition at line 502 of file parser.h.

**7.9.1.33 #define BELL 290**

Definition at line 503 of file parser.h.

**7.9.1.34 #define BINARY 291**

Definition at line 504 of file parser.h.

7.9.1.35 `#define BINARY_C_LONG` 292

Definition at line 505 of file parser.h.

7.9.1.36 `#define BINARY_CHAR` 293

Definition at line 506 of file parser.h.

7.9.1.37 `#define BINARY_DOUBLE` 294

Definition at line 507 of file parser.h.

7.9.1.38 `#define BINARY_LONG` 295

Definition at line 508 of file parser.h.

7.9.1.39 `#define BINARY_SHORT` 296

Definition at line 509 of file parser.h.

7.9.1.40 `#define BLANK` 297

Definition at line 510 of file parser.h.

7.9.1.41 `#define BLANK_LINE` 298

Definition at line 511 of file parser.h.

7.9.1.42 `#define BLANK_SCREEN` 299

Definition at line 512 of file parser.h.

7.9.1.43 `#define BLINK` 300

Definition at line 513 of file parser.h.

7.9.1.44 `#define BLOCK` 301

Definition at line 514 of file parser.h.

**7.9.1.45 #define BOTTOM 302**

Definition at line 515 of file parser.h.

**7.9.1.46 #define BY 303**

Definition at line 516 of file parser.h.

**7.9.1.47 #define BYTE\_LENGTH 304**

Definition at line 517 of file parser.h.

**7.9.1.48 #define CALL 305**

Definition at line 518 of file parser.h.

**7.9.1.49 #define CANCEL 306**

Definition at line 519 of file parser.h.

**7.9.1.50 #define CH 307**

Definition at line 520 of file parser.h.

**7.9.1.51 #define CHAINING 308**

Definition at line 521 of file parser.h.

**7.9.1.52 #define CHARACTER 309**

Definition at line 522 of file parser.h.

**7.9.1.53 #define CHARACTERS 310**

Definition at line 523 of file parser.h.

**7.9.1.54 #define CLASS 311**

Definition at line 524 of file parser.h.

7.9.1.55 `#define CLOSE 312`

Definition at line 525 of file parser.h.

7.9.1.56 `#define CODE 313`

Definition at line 526 of file parser.h.

7.9.1.57 `#define CODE_SET 314`

Definition at line 527 of file parser.h.

7.9.1.58 `#define COL 316`

Definition at line 529 of file parser.h.

7.9.1.59 `#define COLLATING 315`

Definition at line 528 of file parser.h.

7.9.1.60 `#define COLS 317`

Definition at line 530 of file parser.h.

7.9.1.61 `#define COLUMN 318`

Definition at line 531 of file parser.h.

7.9.1.62 `#define COLUMNS 319`

Definition at line 532 of file parser.h.

7.9.1.63 `#define COMMA 320`

Definition at line 533 of file parser.h.

7.9.1.64 `#define COMMA_DELIM 322`

Definition at line 535 of file parser.h.

**7.9.1.65 #define COMMAND\_LINE 321**

Definition at line 534 of file parser.h.

**7.9.1.66 #define COMMIT 323**

Definition at line 536 of file parser.h.

**7.9.1.67 #define COMMON 324**

Definition at line 537 of file parser.h.

**7.9.1.68 #define COMP 325**

Definition at line 538 of file parser.h.

**7.9.1.69 #define COMP\_1 327**

Definition at line 540 of file parser.h.

**7.9.1.70 #define COMP\_2 328**

Definition at line 541 of file parser.h.

**7.9.1.71 #define COMP\_3 329**

Definition at line 542 of file parser.h.

**7.9.1.72 #define COMP\_4 330**

Definition at line 543 of file parser.h.

**7.9.1.73 #define COMP\_5 331**

Definition at line 544 of file parser.h.

**7.9.1.74 #define COMP\_X 332**

Definition at line 545 of file parser.h.

7.9.1.75 `#define COMPUTE 326`

Definition at line 539 of file parser.h.

7.9.1.76 `#define CONCATENATE_FUNC 333`

Definition at line 546 of file parser.h.

7.9.1.77 `#define CONFIGURATION 334`

Definition at line 547 of file parser.h.

7.9.1.78 `#define CONSTANT 335`

Definition at line 548 of file parser.h.

7.9.1.79 `#define CONTAINS 336`

Definition at line 549 of file parser.h.

7.9.1.80 `#define CONTENT 337`

Definition at line 550 of file parser.h.

7.9.1.81 `#define CONTINUE 338`

Definition at line 551 of file parser.h.

7.9.1.82 `#define CONTROL 339`

Definition at line 552 of file parser.h.

7.9.1.83 `#define CONTROL_FOOTING 341`

Definition at line 554 of file parser.h.

7.9.1.84 `#define CONTROL_HEADING 342`

Definition at line 555 of file parser.h.



**7.9.1.85 #define CONTROLS 340**

Definition at line 553 of file parser.h.

**7.9.1.86 #define CONVERTING 343**

Definition at line 556 of file parser.h.

**7.9.1.87 #define CORRESPONDING 344**

Definition at line 557 of file parser.h.

**7.9.1.88 #define COUNT 345**

Definition at line 558 of file parser.h.

**7.9.1.89 #define CRT 346**

Definition at line 559 of file parser.h.

**7.9.1.90 #define CURRENCY 347**

Definition at line 560 of file parser.h.

**7.9.1.91 #define CURRENT\_DATE\_FUNC 348**

Definition at line 561 of file parser.h.

**7.9.1.92 #define CURSOR 349**

Definition at line 562 of file parser.h.

**7.9.1.93 #define CYCLE 350**

Definition at line 563 of file parser.h.

**7.9.1.94 #define DATA 351**

Definition at line 564 of file parser.h.

7.9.1.95 `#define DATE 352`

Definition at line 565 of file parser.h.

7.9.1.96 `#define DAY 353`

Definition at line 566 of file parser.h.

7.9.1.97 `#define DAY_OF_WEEK 354`

Definition at line 567 of file parser.h.

7.9.1.98 `#define DE 355`

Definition at line 568 of file parser.h.

7.9.1.99 `#define DEBUGGING 356`

Definition at line 569 of file parser.h.

7.9.1.100 `#define DECIMAL_POINT 357`

Definition at line 570 of file parser.h.

7.9.1.101 `#define DECLARATIVES 358`

Definition at line 571 of file parser.h.

7.9.1.102 `#define DEFAULT 359`

Definition at line 572 of file parser.h.

7.9.1.103 `#define DELETE 360`

Definition at line 573 of file parser.h.

7.9.1.104 `#define DELIMITED 361`

Definition at line 574 of file parser.h.

7.9.1.105 `#define DELIMITER 362`

Definition at line 575 of file parser.h.

7.9.1.106 `#define DEPENDING 363`

Definition at line 576 of file parser.h.

7.9.1.107 `#define DESCENDING 364`

Definition at line 577 of file parser.h.

7.9.1.108 `#define DETAIL 365`

Definition at line 578 of file parser.h.

7.9.1.109 `#define DISK 366`

Definition at line 579 of file parser.h.

7.9.1.110 `#define DISPLAY 367`

Definition at line 580 of file parser.h.

7.9.1.111 `#define DIVIDE 368`

Definition at line 581 of file parser.h.

7.9.1.112 `#define DIVISION 369`

Definition at line 582 of file parser.h.

7.9.1.113 `#define DOWN 370`

Definition at line 583 of file parser.h.

7.9.1.114 `#define DUPLICATES 371`

Definition at line 584 of file parser.h.

7.9.1.115 `#define DYNAMIC 372`

Definition at line 585 of file parser.h.

7.9.1.116 `#define EBCDIC 373`

Definition at line 586 of file parser.h.

7.9.1.117 `#define ELSE 374`

Definition at line 587 of file parser.h.

7.9.1.118 `#define END 375`

Definition at line 588 of file parser.h.

7.9.1.119 `#define END_ACCEPT 376`

Definition at line 589 of file parser.h.

7.9.1.120 `#define END_ADD 377`

Definition at line 590 of file parser.h.

7.9.1.121 `#define END_CALL 378`

Definition at line 591 of file parser.h.

7.9.1.122 `#define END_COMPUTE 379`

Definition at line 592 of file parser.h.

7.9.1.123 `#define END_DELETE 380`

Definition at line 593 of file parser.h.

7.9.1.124 `#define END_DISPLAY 381`

Definition at line 594 of file parser.h.

7.9.1.125 `#define END_DIVIDE 382`

Definition at line 595 of file parser.h.

7.9.1.126 `#define END_EVALUATE 383`

Definition at line 596 of file parser.h.

7.9.1.127 `#define END_FUNCTION 384`

Definition at line 597 of file parser.h.

7.9.1.128 `#define END_IF 385`

Definition at line 598 of file parser.h.

7.9.1.129 `#define END_MULTIPLY 386`

Definition at line 599 of file parser.h.

7.9.1.130 `#define END_PERFORM 387`

Definition at line 600 of file parser.h.

7.9.1.131 `#define END_PROGRAM 388`

Definition at line 601 of file parser.h.

7.9.1.132 `#define END_READ 389`

Definition at line 602 of file parser.h.

7.9.1.133 `#define END_RETURN 390`

Definition at line 603 of file parser.h.

7.9.1.134 `#define END_REWRITE 391`

Definition at line 604 of file parser.h.

7.9.1.135 `#define END_SEARCH 392`

Definition at line 605 of file parser.h.

7.9.1.136 `#define END_START 393`

Definition at line 606 of file parser.h.

7.9.1.137 `#define END_STRING 394`

Definition at line 607 of file parser.h.

7.9.1.138 `#define END_SUBTRACT 395`

Definition at line 608 of file parser.h.

7.9.1.139 `#define END_UNSTRING 396`

Definition at line 609 of file parser.h.

7.9.1.140 `#define END_WRITE 397`

Definition at line 610 of file parser.h.

7.9.1.141 `#define ENTRY 398`

Definition at line 611 of file parser.h.

7.9.1.142 `#define ENVIRONMENT 399`

Definition at line 612 of file parser.h.

7.9.1.143 `#define ENVIRONMENT_NAME 400`

Definition at line 613 of file parser.h.

7.9.1.144 `#define ENVIRONMENT_VALUE 401`

Definition at line 614 of file parser.h.

7.9.1.145 **#define EOL 402**

Definition at line 615 of file parser.h.

7.9.1.146 **#define EOP 403**

Definition at line 616 of file parser.h.

7.9.1.147 **#define EOS 404**

Definition at line 617 of file parser.h.

7.9.1.148 **#define EQUAL 405**

Definition at line 618 of file parser.h.

7.9.1.149 **#define EQUALS 406**

Definition at line 619 of file parser.h.

7.9.1.150 **#define ERASE 407**

Definition at line 620 of file parser.h.

7.9.1.151 **#define ERROR 408**

Definition at line 621 of file parser.h.

7.9.1.152 **#define ESCAPE 409**

Definition at line 622 of file parser.h.

7.9.1.153 **#define EVALUATE 410**

Definition at line 623 of file parser.h.

7.9.1.154 **#define EVENT\_STATUS 411**

Definition at line 624 of file parser.h.

7.9.1.155 `#define EXCEPTION 412`

Definition at line 625 of file parser.h.

7.9.1.156 `#define EXCLUSIVE 413`

Definition at line 626 of file parser.h.

7.9.1.157 `#define EXIT 414`

Definition at line 627 of file parser.h.

7.9.1.158 `#define EXTEND 415`

Definition at line 628 of file parser.h.

7.9.1.159 `#define EXTERNAL 416`

Definition at line 629 of file parser.h.

7.9.1.160 `#define FD 417`

Definition at line 630 of file parser.h.

7.9.1.161 `#define FILE_CONTROL 418`

Definition at line 631 of file parser.h.

7.9.1.162 `#define FILE_ID 419`

Definition at line 632 of file parser.h.

7.9.1.163 `#define FILLER 420`

Definition at line 633 of file parser.h.

7.9.1.164 `#define FINAL 421`

Definition at line 634 of file parser.h.



7.9.1.165 `#define FIRST 422`

Definition at line 635 of file parser.h.

7.9.1.166 `#define FOOTING 423`

Definition at line 636 of file parser.h.

7.9.1.167 `#define FOR 424`

Definition at line 637 of file parser.h.

7.9.1.168 `#define FOREGROUND_COLOR 425`

Definition at line 638 of file parser.h.

7.9.1.169 `#define FOREVER 426`

Definition at line 639 of file parser.h.

7.9.1.170 `#define FREE 427`

Definition at line 640 of file parser.h.

7.9.1.171 `#define FROM 428`

Definition at line 641 of file parser.h.

7.9.1.172 `#define FULL 429`

Definition at line 642 of file parser.h.

7.9.1.173 `#define FUNCTION 430`

Definition at line 643 of file parser.h.

7.9.1.174 `#define FUNCTION_ID 431`

Definition at line 644 of file parser.h.

7.9.1.175 `#define FUNCTION_NAME` 432

Definition at line 645 of file parser.h.

7.9.1.176 `#define GE` 433

Definition at line 646 of file parser.h.

7.9.1.177 `#define GENERATE` 434

Definition at line 647 of file parser.h.

7.9.1.178 `#define GIVING` 435

Definition at line 648 of file parser.h.

7.9.1.179 `#define GLOBAL` 436

Definition at line 649 of file parser.h.

7.9.1.180 `#define GO` 437

Definition at line 650 of file parser.h.

7.9.1.181 `#define GOBACK` 438

Definition at line 651 of file parser.h.

7.9.1.182 `#define GREATER` 439

Definition at line 652 of file parser.h.

7.9.1.183 `#define GROUP` 440

Definition at line 653 of file parser.h.

7.9.1.184 `#define HEADING` 441

Definition at line 654 of file parser.h.

7.9.1.185 `#define HIGH_VALUE 443`

Definition at line 656 of file parser.h.

7.9.1.186 `#define HIGHLIGHT 442`

Definition at line 655 of file parser.h.

7.9.1.187 `#define I_O 463`

Definition at line 676 of file parser.h.

7.9.1.188 `#define I_O_CONTROL 464`

Definition at line 677 of file parser.h.

7.9.1.189 `#define IDENTIFICATION 444`

Definition at line 657 of file parser.h.

7.9.1.190 `#define IF 445`

Definition at line 658 of file parser.h.

7.9.1.191 `#define IGNORE 446`

Definition at line 659 of file parser.h.

7.9.1.192 `#define IGNORING 447`

Definition at line 660 of file parser.h.

7.9.1.193 `#define IN 448`

Definition at line 661 of file parser.h.

7.9.1.194 `#define INDEX 449`

Definition at line 662 of file parser.h.

7.9.1.195 `#define INDEXED` 450

Definition at line 663 of file parser.h.

7.9.1.196 `#define INDICATE` 451

Definition at line 664 of file parser.h.

7.9.1.197 `#define INITIALIZE` 452

Definition at line 665 of file parser.h.

7.9.1.198 `#define INITIALIZED` 453

Definition at line 666 of file parser.h.

7.9.1.199 `#define INITIATE` 454

Definition at line 667 of file parser.h.

7.9.1.200 `#define INPUT` 455

Definition at line 668 of file parser.h.

7.9.1.201 `#define INPUT_OUTPUT` 456

Definition at line 669 of file parser.h.

7.9.1.202 `#define INSPECT` 457

Definition at line 670 of file parser.h.

7.9.1.203 `#define INTO` 458

Definition at line 671 of file parser.h.

7.9.1.204 `#define INTRINSIC` 459

Definition at line 672 of file parser.h.

7.9.1.205 `#define INVALID 460`

Definition at line 673 of file parser.h.

7.9.1.206 `#define INVALID_KEY 461`

Definition at line 674 of file parser.h.

7.9.1.207 `#define IS 462`

Definition at line 675 of file parser.h.

7.9.1.208 `#define JUSTIFIED 465`

Definition at line 678 of file parser.h.

7.9.1.209 `#define KEY 466`

Definition at line 679 of file parser.h.

7.9.1.210 `#define LABEL 467`

Definition at line 680 of file parser.h.

7.9.1.211 `#define LAST 468`

Definition at line 681 of file parser.h.

7.9.1.212 `#define LAST_DETAIL 469`

Definition at line 682 of file parser.h.

7.9.1.213 `#define LE 470`

Definition at line 683 of file parser.h.

7.9.1.214 `#define LEADING 471`

Definition at line 684 of file parser.h.

7.9.1.215 `#define LEFT 472`

Definition at line 685 of file parser.h.

7.9.1.216 `#define LENGTH 473`

Definition at line 686 of file parser.h.

7.9.1.217 `#define LESS 474`

Definition at line 687 of file parser.h.

7.9.1.218 `#define LIMIT 475`

Definition at line 688 of file parser.h.

7.9.1.219 `#define LIMITS 476`

Definition at line 689 of file parser.h.

7.9.1.220 `#define LINAGE 477`

Definition at line 690 of file parser.h.

7.9.1.221 `#define LINAGE_COUNTER 478`

Definition at line 691 of file parser.h.

7.9.1.222 `#define LINE 479`

Definition at line 692 of file parser.h.

7.9.1.223 `#define LINES 480`

Definition at line 693 of file parser.h.

7.9.1.224 `#define LINKAGE 481`

Definition at line 694 of file parser.h.

7.9.1.225 `#define LITERAL` 482

Definition at line 695 of file parser.h.

7.9.1.226 `#define LOCAL_STORAGE` 485

Definition at line 698 of file parser.h.

7.9.1.227 `#define LOCALE` 483

Definition at line 696 of file parser.h.

7.9.1.228 `#define LOCALE_DT_FUNC` 484

Definition at line 697 of file parser.h.

7.9.1.229 `#define LOCK` 486

Definition at line 699 of file parser.h.

7.9.1.230 `#define LOW_VALUE` 489

Definition at line 702 of file parser.h.

7.9.1.231 `#define LOWER_CASE_FUNC` 487

Definition at line 700 of file parser.h.

7.9.1.232 `#define LOWLIGHT` 488

Definition at line 701 of file parser.h.

7.9.1.233 `#define MANUAL` 490

Definition at line 703 of file parser.h.

7.9.1.234 `#define MEMORY` 491

Definition at line 704 of file parser.h.

7.9.1.235 `#define MERGE 492`

Definition at line 705 of file parser.h.

7.9.1.236 `#define MINUS 493`

Definition at line 706 of file parser.h.

7.9.1.237 `#define MNEMONIC_NAME 494`

Definition at line 707 of file parser.h.

7.9.1.238 `#define MODE 495`

Definition at line 708 of file parser.h.

7.9.1.239 `#define MOVE 496`

Definition at line 709 of file parser.h.

7.9.1.240 `#define MULTIPLE 497`

Definition at line 710 of file parser.h.

7.9.1.241 `#define MULTIPLY 498`

Definition at line 711 of file parser.h.

7.9.1.242 `#define NATIONAL 499`

Definition at line 712 of file parser.h.

7.9.1.243 `#define NATIONAL_EDITED 500`

Definition at line 713 of file parser.h.

7.9.1.244 `#define NATIVE 501`

Definition at line 714 of file parser.h.



7.9.1.245 `#define NE 502`

Definition at line 715 of file parser.h.

7.9.1.246 `#define NEGATIVE 503`

Definition at line 716 of file parser.h.

7.9.1.247 `#define NEXT 504`

Definition at line 717 of file parser.h.

7.9.1.248 `#define NEXT_SENTENCE 505`

Definition at line 718 of file parser.h.

7.9.1.249 `#define NO 506`

Definition at line 719 of file parser.h.

7.9.1.250 `#define NO_ADVANCING 514`

Definition at line 727 of file parser.h.

7.9.1.251 `#define NOT 507`

Definition at line 720 of file parser.h.

7.9.1.252 `#define NOT_END 508`

Definition at line 721 of file parser.h.

7.9.1.253 `#define NOT_EOP 509`

Definition at line 722 of file parser.h.

7.9.1.254 `#define NOT_EXCEPTION 510`

Definition at line 723 of file parser.h.

7.9.1.255 `#define NOT_INVALID_KEY 511`

Definition at line 724 of file parser.h.

7.9.1.256 `#define NOT_OVERFLOW 512`

Definition at line 725 of file parser.h.

7.9.1.257 `#define NOT_SIZE_ERROR 513`

Definition at line 726 of file parser.h.

7.9.1.258 `#define NUMBER 515`

Definition at line 728 of file parser.h.

7.9.1.259 `#define NUMBERS 516`

Definition at line 729 of file parser.h.

7.9.1.260 `#define NUMERIC 517`

Definition at line 730 of file parser.h.

7.9.1.261 `#define NUMERIC_EDITED 518`

Definition at line 731 of file parser.h.

7.9.1.262 `#define NUMVALC_FUNC 519`

Definition at line 732 of file parser.h.

7.9.1.263 `#define OBJECT_COMPUTER 520`

Definition at line 733 of file parser.h.

7.9.1.264 `#define OCCURS 521`

Definition at line 734 of file parser.h.

7.9.1.265 `#define OF 522`

Definition at line 735 of file parser.h.

7.9.1.266 `#define OFF 523`

Definition at line 736 of file parser.h.

7.9.1.267 `#define OMITTED 524`

Definition at line 737 of file parser.h.

7.9.1.268 `#define ON 525`

Definition at line 738 of file parser.h.

7.9.1.269 `#define ONLY 526`

Definition at line 739 of file parser.h.

7.9.1.270 `#define OPEN 527`

Definition at line 740 of file parser.h.

7.9.1.271 `#define OPTIONAL 528`

Definition at line 741 of file parser.h.

7.9.1.272 `#define OR 529`

Definition at line 742 of file parser.h.

7.9.1.273 `#define ORDER 530`

Definition at line 743 of file parser.h.

7.9.1.274 `#define ORGANIZATION 531`

Definition at line 744 of file parser.h.

7.9.1.275 `#define OTHER` 532

Definition at line 745 of file parser.h.

7.9.1.276 `#define OUTPUT` 533

Definition at line 746 of file parser.h.

7.9.1.277 `#define OVERFLOW` 534

Definition at line 747 of file parser.h.

7.9.1.278 `#define OVERLINE` 535

Definition at line 748 of file parser.h.

7.9.1.279 `#define PACKED_DECIMAL` 536

Definition at line 749 of file parser.h.

7.9.1.280 `#define PADDING` 537

Definition at line 750 of file parser.h.

7.9.1.281 `#define PAGE` 538

Definition at line 751 of file parser.h.

7.9.1.282 `#define PAGE_FOOTING` 539

Definition at line 752 of file parser.h.

7.9.1.283 `#define PAGE_HEADING` 540

Definition at line 753 of file parser.h.

7.9.1.284 `#define PARAGRAPH` 541

Definition at line 754 of file parser.h.

7.9.1.285 `#define PERFORM 542`

Definition at line 755 of file parser.h.

7.9.1.286 `#define PICTURE 543`

Definition at line 756 of file parser.h.

7.9.1.287 `#define PLUS 544`

Definition at line 757 of file parser.h.

7.9.1.288 `#define POINTER 545`

Definition at line 758 of file parser.h.

7.9.1.289 `#define POSITION 546`

Definition at line 759 of file parser.h.

7.9.1.290 `#define POSITIVE 547`

Definition at line 760 of file parser.h.

7.9.1.291 `#define PRESENT 548`

Definition at line 761 of file parser.h.

7.9.1.292 `#define PREVIOUS 549`

Definition at line 762 of file parser.h.

7.9.1.293 `#define PRINTER 550`

Definition at line 763 of file parser.h.

7.9.1.294 `#define PRINTING 551`

Definition at line 764 of file parser.h.

7.9.1.295 `#define PROCEDURE 552`

Definition at line 765 of file parser.h.

7.9.1.296 `#define PROCEDURES 553`

Definition at line 766 of file parser.h.

7.9.1.297 `#define PROCEED 554`

Definition at line 767 of file parser.h.

7.9.1.298 `#define PROGRAM 555`

Definition at line 768 of file parser.h.

7.9.1.299 `#define PROGRAM_ID 556`

Definition at line 769 of file parser.h.

7.9.1.300 `#define PROGRAM_NAME 557`

Definition at line 770 of file parser.h.

7.9.1.301 `#define PROGRAM_POINTER 558`

Definition at line 771 of file parser.h.

7.9.1.302 `#define PROMPT 559`

Definition at line 772 of file parser.h.

7.9.1.303 `#define QUOTE 560`

Definition at line 773 of file parser.h.

7.9.1.304 `#define RANDOM 561`

Definition at line 774 of file parser.h.

7.9.1.305 `#define RD 562`

Definition at line 775 of file parser.h.

7.9.1.306 `#define READ 563`

Definition at line 776 of file parser.h.

7.9.1.307 `#define RECORD 564`

Definition at line 777 of file parser.h.

7.9.1.308 `#define RECORDING 565`

Definition at line 778 of file parser.h.

7.9.1.309 `#define RECORDS 566`

Definition at line 779 of file parser.h.

7.9.1.310 `#define RECURSIVE 567`

Definition at line 780 of file parser.h.

7.9.1.311 `#define REDEFINES 568`

Definition at line 781 of file parser.h.

7.9.1.312 `#define REEL 569`

Definition at line 782 of file parser.h.

7.9.1.313 `#define REFERENCE 570`

Definition at line 783 of file parser.h.

7.9.1.314 `#define RELATIVE 571`

Definition at line 784 of file parser.h.

7.9.1.315 **#define RELEASE 572**

Definition at line 785 of file parser.h.

7.9.1.316 **#define REMAINDER 573**

Definition at line 786 of file parser.h.

7.9.1.317 **#define REMOVAL 574**

Definition at line 787 of file parser.h.

7.9.1.318 **#define RENAMES 575**

Definition at line 788 of file parser.h.

7.9.1.319 **#define REPLACING 576**

Definition at line 789 of file parser.h.

7.9.1.320 **#define REPORT 577**

Definition at line 790 of file parser.h.

7.9.1.321 **#define REPORT\_FOOTING 580**

Definition at line 793 of file parser.h.

7.9.1.322 **#define REPORT\_HEADING 581**

Definition at line 794 of file parser.h.

7.9.1.323 **#define REPORTING 578**

Definition at line 791 of file parser.h.

7.9.1.324 **#define REPORTS 579**

Definition at line 792 of file parser.h.



7.9.1.325 `#define REPOSITORY 582`

Definition at line 795 of file parser.h.

7.9.1.326 `#define REQUIRED 583`

Definition at line 796 of file parser.h.

7.9.1.327 `#define RESERVE 584`

Definition at line 797 of file parser.h.

7.9.1.328 `#define RETURN 585`

Definition at line 798 of file parser.h.

7.9.1.329 `#define RETURNING 586`

Definition at line 799 of file parser.h.

7.9.1.330 `#define REVERSE_FUNC 587`

Definition at line 800 of file parser.h.

7.9.1.331 `#define REVERSE_VIDEO 588`

Definition at line 801 of file parser.h.

7.9.1.332 `#define REWIND 589`

Definition at line 802 of file parser.h.

7.9.1.333 `#define REWRITE 590`

Definition at line 803 of file parser.h.

7.9.1.334 `#define RIGHT 591`

Definition at line 804 of file parser.h.

7.9.1.335 `#define ROLLBACK 592`

Definition at line 805 of file parser.h.

7.9.1.336 `#define ROUNDED 593`

Definition at line 806 of file parser.h.

7.9.1.337 `#define RUN 594`

Definition at line 807 of file parser.h.

7.9.1.338 `#define SAME 595`

Definition at line 808 of file parser.h.

7.9.1.339 `#define SCREEN 596`

Definition at line 809 of file parser.h.

7.9.1.340 `#define SCREEN_CONTROL 597`

Definition at line 810 of file parser.h.

7.9.1.341 `#define SCROLL 598`

Definition at line 811 of file parser.h.

7.9.1.342 `#define SD 599`

Definition at line 812 of file parser.h.

7.9.1.343 `#define SEARCH 600`

Definition at line 813 of file parser.h.

7.9.1.344 `#define SECTION 601`

Definition at line 814 of file parser.h.

7.9.1.345 `#define SECURE 602`

Definition at line 815 of file parser.h.

7.9.1.346 `#define SEGMENT_LIMIT 603`

Definition at line 816 of file parser.h.

7.9.1.347 `#define SELECT 604`

Definition at line 817 of file parser.h.

7.9.1.348 `#define SEMI_COLON 605`

Definition at line 818 of file parser.h.

7.9.1.349 `#define SENTENCE 606`

Definition at line 819 of file parser.h.

7.9.1.350 `#define SEPARATE 607`

Definition at line 820 of file parser.h.

7.9.1.351 `#define SEQUENCE 608`

Definition at line 821 of file parser.h.

7.9.1.352 `#define SEQUENTIAL 609`

Definition at line 822 of file parser.h.

7.9.1.353 `#define SET 610`

Definition at line 823 of file parser.h.

7.9.1.354 `#define SHARING 611`

Definition at line 824 of file parser.h.

7.9.1.355 `#define SIGN 612`

Definition at line 825 of file parser.h.

7.9.1.356 `#define SIGNED 613`

Definition at line 826 of file parser.h.

7.9.1.357 `#define SIGNED_INT 614`

Definition at line 827 of file parser.h.

7.9.1.358 `#define SIGNED_LONG 615`

Definition at line 828 of file parser.h.

7.9.1.359 `#define SIGNED_SHORT 616`

Definition at line 829 of file parser.h.

7.9.1.360 `#define SIZE 617`

Definition at line 830 of file parser.h.

7.9.1.361 `#define SIZE_ERROR 618`

Definition at line 831 of file parser.h.

7.9.1.362 `#define SORT 619`

Definition at line 832 of file parser.h.

7.9.1.363 `#define SORT_MERGE 620`

Definition at line 833 of file parser.h.

7.9.1.364 `#define SOURCE 621`

Definition at line 834 of file parser.h.

7.9.1.365 **#define SOURCE\_COMPUTER 622**

Definition at line 835 of file parser.h.

7.9.1.366 **#define SPACE 623**

Definition at line 836 of file parser.h.

7.9.1.367 **#define SPECIAL\_NAMES 624**

Definition at line 837 of file parser.h.

7.9.1.368 **#define STANDARD 625**

Definition at line 838 of file parser.h.

7.9.1.369 **#define STANDARD\_1 626**

Definition at line 839 of file parser.h.

7.9.1.370 **#define STANDARD\_2 627**

Definition at line 840 of file parser.h.

7.9.1.371 **#define START 628**

Definition at line 841 of file parser.h.

7.9.1.372 **#define STATUS 629**

Definition at line 842 of file parser.h.

7.9.1.373 **#define STOP 630**

Definition at line 843 of file parser.h.

7.9.1.374 **#define STRING 631**

Definition at line 844 of file parser.h.

7.9.1.375 `#define SUBSTITUTE_CASE_FUNC 633`

Definition at line 846 of file parser.h.

7.9.1.376 `#define SUBSTITUTE_FUNC 632`

Definition at line 845 of file parser.h.

7.9.1.377 `#define SUBTRACT 634`

Definition at line 847 of file parser.h.

7.9.1.378 `#define SUM 635`

Definition at line 848 of file parser.h.

7.9.1.379 `#define SUPPRESS 636`

Definition at line 849 of file parser.h.

7.9.1.380 `#define SYMBOLIC 637`

Definition at line 850 of file parser.h.

7.9.1.381 `#define SYNCHRONIZED 638`

Definition at line 851 of file parser.h.

7.9.1.382 `#define TALLYING 639`

Definition at line 852 of file parser.h.

7.9.1.383 `#define TAPE 640`

Definition at line 853 of file parser.h.

7.9.1.384 `#define TERMINATE 641`

Definition at line 854 of file parser.h.

7.9.1.385 `#define TEST 642`

Definition at line 855 of file parser.h.

7.9.1.386 `#define THAN 643`

Definition at line 856 of file parser.h.

7.9.1.387 `#define THEN 644`

Definition at line 857 of file parser.h.

7.9.1.388 `#define THRU 645`

Definition at line 858 of file parser.h.

7.9.1.389 `#define TIME 646`

Definition at line 859 of file parser.h.

7.9.1.390 `#define TIMES 647`

Definition at line 860 of file parser.h.

7.9.1.391 `#define TO 648`

Definition at line 861 of file parser.h.

7.9.1.392 `#define TOK_FALSE 649`

Definition at line 862 of file parser.h.

7.9.1.393 `#define TOK_FILE 650`

Definition at line 863 of file parser.h.

7.9.1.394 `#define TOK_INITIAL 651`

Definition at line 864 of file parser.h.

7.9.1.395 `#define TOK_NULL 652`

Definition at line 865 of file parser.h.

7.9.1.396 `#define TOK_TRUE 653`

Definition at line 866 of file parser.h.

7.9.1.397 `#define TOKEN_EOF 0`

Definition at line 470 of file parser.h.

7.9.1.398 `#define TOP 654`

Definition at line 867 of file parser.h.

7.9.1.399 `#define TRAILING 655`

Definition at line 868 of file parser.h.

7.9.1.400 `#define TRANSFORM 656`

Definition at line 869 of file parser.h.

7.9.1.401 `#define TRIM_FUNCTION 657`

Definition at line 870 of file parser.h.

7.9.1.402 `#define TYPE 658`

Definition at line 871 of file parser.h.

7.9.1.403 `#define UNARY_SIGN 692`

Definition at line 905 of file parser.h.

7.9.1.404 `#define UNDERLINE 659`

Definition at line 872 of file parser.h.



7.9.1.405 `#define UNIT 660`

Definition at line 873 of file parser.h.

7.9.1.406 `#define UNLOCK 661`

Definition at line 874 of file parser.h.

7.9.1.407 `#define UNSIGNED 662`

Definition at line 875 of file parser.h.

7.9.1.408 `#define UNSIGNED_INT 663`

Definition at line 876 of file parser.h.

7.9.1.409 `#define UNSIGNED_LONG 664`

Definition at line 877 of file parser.h.

7.9.1.410 `#define UNSIGNED_SHORT 665`

Definition at line 878 of file parser.h.

7.9.1.411 `#define UNSTRING 666`

Definition at line 879 of file parser.h.

7.9.1.412 `#define UNTIL 667`

Definition at line 880 of file parser.h.

7.9.1.413 `#define UP 668`

Definition at line 881 of file parser.h.

7.9.1.414 `#define UPDATE 669`

Definition at line 882 of file parser.h.

7.9.1.415 `#define UPON` 670

Definition at line 883 of file parser.h.

7.9.1.416 `#define UPON_ARGUMENT_NUMBER` 671

Definition at line 884 of file parser.h.

7.9.1.417 `#define UPON_COMMAND_LINE` 672

Definition at line 885 of file parser.h.

7.9.1.418 `#define UPON_ENVIRONMENT_NAME` 673

Definition at line 886 of file parser.h.

7.9.1.419 `#define UPON_ENVIRONMENT_VALUE` 674

Definition at line 887 of file parser.h.

7.9.1.420 `#define UPPER_CASE_FUNC` 675

Definition at line 888 of file parser.h.

7.9.1.421 `#define USAGE` 676

Definition at line 889 of file parser.h.

7.9.1.422 `#define USE` 677

Definition at line 890 of file parser.h.

7.9.1.423 `#define USING` 678

Definition at line 891 of file parser.h.

7.9.1.424 `#define VALUE` 679

Definition at line 892 of file parser.h.

7.9.1.425 `#define VARYING 680`

Definition at line 893 of file parser.h.

7.9.1.426 `#define WAIT 681`

Definition at line 894 of file parser.h.

7.9.1.427 `#define WHEN 682`

Definition at line 895 of file parser.h.

7.9.1.428 `#define WHEN_COMPILED_FUNC 683`

Definition at line 896 of file parser.h.

7.9.1.429 `#define WITH 684`

Definition at line 897 of file parser.h.

7.9.1.430 `#define WORD 685`

Definition at line 898 of file parser.h.

7.9.1.431 `#define WORDS 686`

Definition at line 899 of file parser.h.

7.9.1.432 `#define WORKING_STORAGE 687`

Definition at line 900 of file parser.h.

7.9.1.433 `#define WRITE 688`

Definition at line 901 of file parser.h.

7.9.1.434 `#define yystype YYSTYPE`

Definition at line 912 of file parser.h.

7.9.1.435 `#define YYSTYPE_IS_DECLARED 1`

Definition at line 913 of file parser.h.

7.9.1.436 `#define YYSTYPE_IS_TRIVIAL 1`

Definition at line 914 of file parser.h.

7.9.1.437 `#define YYYYDDD 689`

Definition at line 902 of file parser.h.

7.9.1.438 `#define YYYYMMDD 690`

Definition at line 903 of file parser.h.

7.9.1.439 `#define ZERO 691`

Definition at line 904 of file parser.h.

## 7.9.2 Typedef Documentation

7.9.2.1 `typedef int YYSTYPE`

Definition at line 911 of file parser.h.

## 7.9.3 Enumeration Type Documentation

7.9.3.1 `enum yytokentype`

**Enumerator:**

***TOKEN\_EOF***

***ACCEPT***

***ACCESS***

***ADD***

***ADDRESS***

***ADVANCING***

***AFTER***

***ALL***

***ALLOCATE***

***ALPHABET***

**ALPHABETIC**  
**ALPHABETIC\_LOWER**  
**ALPHABETIC\_UPPER**  
**ALPHANUMERIC**  
**ALPHANUMERIC\_EDITED**  
**ALSO**  
**ALTER**  
**ALTERNATE**  
**AND**  
**ANY**  
**ARE**  
**AREA**  
**ARGUMENT\_NUMBER**  
**ARGUMENT\_VALUE**  
**AS**  
**ASCENDING**  
**ASSIGN**  
**AT**  
**AUTO**  
**AUTOMATIC**  
**BACKGROUND\_COLOR**  
**BASED**  
**BEFORE**  
**BELL**  
**BINARY**  
**BINARY\_C\_LONG**  
**BINARY\_CHAR**  
**BINARY\_DOUBLE**  
**BINARY\_LONG**  
**BINARY\_SHORT**  
**BLANK**  
**BLANK\_LINE**  
**BLANK\_SCREEN**  
**BLINK**  
**BLOCK**  
**BOTTOM**  
**BY**  
**BYTE\_LENGTH**

**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**  
**COMMAND\_LINE**  
**COMMA\_DELIM**  
**COMMIT**  
**COMMON**  
**COMP**  
**COMPUTE**  
**COMP\_1**  
**COMP\_2**  
**COMP\_3**  
**COMP\_4**  
**COMP\_5**  
**COMP\_X**  
**CONCATENATE\_FUNC**  
**CONFIGURATION**  
**CONSTANT**  
**CONTAINS**  
**CONTENT**  
**CONTINUE**  
**CONTROL**  
**CONTROLS**  
**CONTROL\_FOOTING**  
**CONTROL\_HEADING**

*CONVERTING*  
*CORRESPONDING*  
*COUNT*  
*CRT*  
*CURRENCY*  
*CURRENT\_DATE\_FUNC*  
*CURSOR*  
*CYCLE*  
*DATA*  
*DATE*  
*DAY*  
*DAY\_OF\_WEEK*  
*DE*  
*DEBUGGING*  
*DECIMAL\_POINT*  
*DECLARATIVES*  
*DEFAULT*  
*DELETE*  
*DELIMITED*  
*DELIMITER*  
*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*

*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*



*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*FOREGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*  
*GROUP*  
*HEADING*  
*HIGHLIGHT*  
*HIGH\_VALUE*  
*IDENTIFICATION*  
*IF*  
*IGNORE*  
*IGNORING*  
*IN*  
*INDEX*  
*INDEXED*  
*INDICATE*  
*INITIALIZE*  
*INITIALIZED*  
*INITIATE*  
*INPUT*  
*INPUT\_OUTPUT*

*INSPECT*  
*INTO*  
*INTRINSIC*  
*INVALID*  
*INVALID\_KEY*  
*IS*  
*I\_O*  
*I\_O\_CONTROL*  
*JUSTIFIED*  
*KEY*  
*LABEL*  
*LAST*  
*LAST\_DETAIL*  
*LE*  
*LEADING*  
*LEFT*  
*LENGTH*  
*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*

*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*  
*OR*  
*ORDER*  
*ORGANIZATION*  
*OTHER*

**OUTPUT**  
**OVERFLOW**  
**OVERLINE**  
**PACKED\_DECIMAL**  
**PADDING**  
**PAGE**  
**PAGE\_FOOTING**  
**PAGE\_HEADING**  
**PARAGRAPH**  
**PERFORM**  
**PICTURE**  
**PLUS**  
**POINTER**  
**POSITION**  
**POSITIVE**  
**PRESENT**  
**PREVIOUS**  
**PRINTER**  
**PRINTING**  
**PROCEDURE**  
**PROCEDURES**  
**PROCEED**  
**PROGRAM**  
**PROGRAM\_ID**  
**PROGRAM\_NAME**  
**PROGRAM\_POINTER**  
**PROMPT**  
**QUOTE**  
**RANDOM**  
**RD**  
**READ**  
**RECORD**  
**RECORDING**  
**RECORDS**  
**RECURSIVE**  
**REDEFINES**  
**REEL**  
**REFERENCE**

*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*  
*REVERSE\_FUNC*  
*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*  
*RIGHT*  
*ROLLBACK*  
*ROUNDED*  
*RUN*  
*SAME*  
*SCREEN*  
*SCREEN\_CONTROL*  
*SCROLL*  
*SD*  
*SEARCH*  
*SECTION*  
*SECURE*  
*SEGMENT\_LIMIT*  
*SELECT*  
*SEMI\_COLON*  
*SENTENCE*  
*SEPARATE*  
*SEQUENCE*

**SEQUENTIAL**  
**SET**  
**SHARING**  
**SIGN**  
**SIGNED**  
**SIGNED\_INT**  
**SIGNED\_LONG**  
**SIGNED\_SHORT**  
**SIZE**  
**SIZE\_ERROR**  
**SORT**  
**SORT\_MERGE**  
**SOURCE**  
**SOURCE\_COMPUTER**  
**SPACE**  
**SPECIAL\_NAMES**  
**STANDARD**  
**STANDARD\_1**  
**STANDARD\_2**  
**START**  
**STATUS**  
**STOP**  
**STRING**  
**SUBSTITUTE\_FUNC**  
**SUBSTITUTE\_CASE\_FUNC**  
**SUBTRACT**  
**SUM**  
**SUPPRESS**  
**SYMBOLIC**  
**SYNCHRONIZED**  
**TALLYING**  
**TAPE**  
**TERMINATE**  
**TEST**  
**THAN**  
**THEN**  
**THRU**  
**TIME**

*TIMES*  
*TO*  
*TOK\_FALSE*  
*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*

*WORD*  
*WORDS*  
*WORKING\_STORAGE*  
*WRITE*  
*YYYYDDD*  
*YYYYMMDD*  
*ZERO*  
*UNARY\_SIGN*  
*TOKEN\_EOF*  
*ACCEPT*  
*ACCESS*  
*ADD*  
*ADDRESS*  
*ADVANCING*  
*AFTER*  
*ALL*  
*ALLOCATE*  
*ALPHABET*  
*ALPHABETIC*  
*ALPHABETIC\_LOWER*  
*ALPHABETIC\_UPPER*  
*ALPHANUMERIC*  
*ALPHANUMERIC\_EDITED*  
*ALSO*  
*ALTER*  
*ALTERNATE*  
*AND*  
*ANY*  
*ARE*  
*AREA*  
*ARGUMENT\_NUMBER*  
*ARGUMENT\_VALUE*  
*AS*  
*ASCENDING*  
*ASSIGN*  
*AT*  
*AUTO*  
*AUTOMATIC*



***BACKGROUND\_COLOR***  
***BASED***  
***BEFORE***  
***BELL***  
***BINARY***  
***BINARY\_C\_LONG***  
***BINARY\_CHAR***  
***BINARY\_DOUBLE***  
***BINARY\_LONG***  
***BINARY\_SHORT***  
***BLANK***  
***BLANK\_LINE***  
***BLANK\_SCREEN***  
***BLINK***  
***BLOCK***  
***BOTTOM***  
***BY***  
***BYTE\_LENGTH***  
***CALL***  
***CANCEL***  
***CH***  
***CHAINING***  
***CHARACTER***  
***CHARACTERS***  
***CLASS***  
***CLOSE***  
***CODE***  
***CODE\_SET***  
***COLLATING***  
***COL***  
***COLS***  
***COLUMN***  
***COLUMNS***  
***COMMA***  
***COMMAND\_LINE***  
***COMMA\_DELIM***  
***COMMIT***  
***COMMON***

*COMP*  
*COMPUTE*  
*COMP\_1*  
*COMP\_2*  
*COMP\_3*  
*COMP\_4*  
*COMP\_5*  
*COMP\_X*  
*CONCATENATE\_FUNC*  
*CONFIGURATION*  
*CONSTANT*  
*CONTAINS*  
*CONTENT*  
*CONTINUE*  
*CONTROL*  
*CONTROLS*  
*CONTROL\_FOOTING*  
*CONTROL\_HEADING*  
*CONVERTING*  
*CORRESPONDING*  
*COUNT*  
*CRT*  
*CURRENCY*  
*CURRENT\_DATE\_FUNC*  
*CURSOR*  
*CYCLE*  
*DATA*  
*DATE*  
*DAY*  
*DAY\_OF\_WEEK*  
*DE*  
*DEBUGGING*  
*DECIMAL\_POINT*  
*DECLARATIVES*  
*DEFAULT*  
*DELETE*  
*DELIMITED*  
*DELIMITER*

*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*

*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*BACKGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*

**GREATER**  
**GROUP**  
**HEADING**  
**HIGHLIGHT**  
**HIGH\_VALUE**  
**IDENTIFICATION**  
**IF**  
**IGNORE**  
**IGNORING**  
**IN**  
**INDEX**  
**INDEXED**  
**INDICATE**  
**INITIALIZE**  
**INITIALIZED**  
**INITIATE**  
**INPUT**  
**INPUT\_OUTPUT**  
**INSPECT**  
**INTO**  
**INTRINSIC**  
**INVALID**  
**INVALID\_KEY**  
**IS**  
**I\_O**  
**I\_O\_CONTROL**  
**JUSTIFIED**  
**KEY**  
**LABEL**  
**LAST**  
**LAST\_DETAIL**  
**LE**  
**LEADING**  
**LEFT**  
**LENGTH**  
**LESS**  
**LIMIT**  
**LIMITS**

*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*

**NUMBER**  
**NUMBERS**  
**NUMERIC**  
**NUMERIC\_EDITED**  
**NUMVALC\_FUNC**  
**OBJECT\_COMPUTER**  
**OCCURS**  
**OF**  
**OFF**  
**OMITTED**  
**ON**  
**ONLY**  
**OPEN**  
**OPTIONAL**  
**OR**  
**ORDER**  
**ORGANIZATION**  
**OTHER**  
**OUTPUT**  
**OVERFLOW**  
**OVERLINE**  
**PACKED\_DECIMAL**  
**PADDING**  
**PAGE**  
**PAGE\_FOOTING**  
**PAGE\_HEADING**  
**PARAGRAPH**  
**PERFORM**  
**PICTURE**  
**PLUS**  
**POINTER**  
**POSITION**  
**POSITIVE**  
**PRESENT**  
**PREVIOUS**  
**PRINTER**  
**PRINTING**  
**PROCEDURE**

*PROCEDURES*  
*PROCEED*  
*PROGRAM*  
*PROGRAM\_ID*  
*PROGRAM\_NAME*  
*PROGRAM\_POINTER*  
*PROMPT*  
*QUOTE*  
*RANDOM*  
*RD*  
*READ*  
*RECORD*  
*RECORDING*  
*RECORDS*  
*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*  
*REVERSE\_FUNC*  
*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*



*RIGHT*  
*ROLLBACK*  
*ROUNDED*  
*RUN*  
*SAME*  
*SCREEN*  
*SCREEN\_CONTROL*  
*SCROLL*  
*SD*  
*SEARCH*  
*SECTION*  
*SECURE*  
*SEGMENT\_LIMIT*  
*SELECT*  
*SEMI\_COLON*  
*SENTENCE*  
*SEPARATE*  
*SEQUENCE*  
*SEQUENTIAL*  
*SET*  
*SHARING*  
*SIGN*  
*SIGNED*  
*SIGNED\_INT*  
*SIGNED\_LONG*  
*SIGNED\_SHORT*  
*SIZE*  
*SIZE\_ERROR*  
*SORT*  
*SORT\_MERGE*  
*SOURCE*  
*SOURCE\_COMPUTER*  
*SPACE*  
*SPECIAL\_NAMES*  
*STANDARD*  
*STANDARD\_1*  
*STANDARD\_2*  
*START*

*STATUS*  
*STOP*  
*STRING*  
*SUBSTITUTE\_FUNC*  
*SUBSTITUTE\_CASE\_FUNC*  
*SUBTRACT*  
*SUM*  
*SUPPRESS*  
*SYMBOLIC*  
*SYNCHRONIZED*  
*TALLYING*  
*TAPE*  
*TERMINATE*  
*TEST*  
*THAN*  
*THEN*  
*THRU*  
*TIME*  
*TIMES*  
*TO*  
*TOK\_FALSE*  
*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*

*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*  
*WORD*  
*WORDS*  
*WORKING\_STORAGE*  
*WRITE*  
*YYYYDDD*  
*YYYYMMDD*  
*ZERO*  
*UNARY\_SIGN*  
*TOKEN\_EOF*  
*COPY*  
*REPLACE*  
*SUPPRESS*  
*PRINTING*  
*REPLACING*  
*OFF*  
*IN*  
*OF*  
*BY*  
*EQEQ*  
*TOKEN*

**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**

Definition at line 31 of file parser.h.

```
{  
    TOKEN_EOF = 0,  
    ACCEPT = 258,  
    ACCESS = 259,  
    ADD = 260,  
    ADDRESS = 261,  
    ADVANCING = 262,  
    AFTER = 263,  
    ALL = 264,  
    ALLOCATE = 265,  
    ALPHABET = 266,  
    ALPHABETIC = 267,  
    ALPHABETIC_LOWER = 268,  
    ALPHABETIC_UPPER = 269,  
    ALPHANUMERIC = 270,  
    ALPHANUMERIC_EDITED = 271,  
    ALSO = 272,  
    ALTER = 273,  
    ALTERNATE = 274,  
    AND = 275,  
    ANY = 276,  
    ARE = 277,  
    AREA = 278,  
    ARGUMENT_NUMBER = 279,  
    ARGUMENT_VALUE = 280,  
    AS = 281,  
    ASCENDING = 282,  
    ASSIGN = 283,  
    AT = 284,  
    AUTO = 285,  
    AUTOMATIC = 286,  
    BACKGROUND_COLOR = 287,  
    BASED = 288,  
    BEFORE = 289,  
    BELL = 290,  
    BINARY = 291,  
    BINARY_C_LONG = 292,  
    BINARY_CHAR = 293,  
    BINARY_DOUBLE = 294,
```

```
BINARY_LONG = 295,  
BINARY_SHORT = 296,  
BLANK = 297,  
BLANK_LINE = 298,  
BLANK_SCREEN = 299,  
BLINK = 300,  
BLOCK = 301,  
BOTTOM = 302,  
BY = 303,  
BYTE_LENGTH = 304,  
CALL = 305,  
CANCEL = 306,  
CH = 307,  
CHAINING = 308,  
CHARACTER = 309,  
CHARACTERS = 310,  
CLASS = 311,  
CLOSE = 312,  
CODE = 313,  
CODE_SET = 314,  
COLLATING = 315,  
COL = 316,  
COLS = 317,  
COLUMN = 318,  
COLUMNS = 319,  
COMMA = 320,  
COMMAND_LINE = 321,  
COMMA_DELIM = 322,  
COMMIT = 323,  
COMMON = 324,  
COMP = 325,  
COMPUTE = 326,  
COMP_1 = 327,  
COMP_2 = 328,  
COMP_3 = 329,  
COMP_4 = 330,  
COMP_5 = 331,  
COMP_X = 332,  
CONCATENATE_FUNC = 333,  
CONFIGURATION = 334,  
CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337,  
CONTINUE = 338,  
CONTROL = 339,  
CONTROLS = 340,  
CONTROL_FOOTING = 341,  
CONTROL_HEADING = 342,  
CONVERTING = 343,  
CORRESPONDING = 344,  
COUNT = 345,  
CRT = 346,  
CURRENCY = 347,  
CURRENT_DATE_FUNC = 348,  
CURSOR = 349,  
CYCLE = 350,  
DATA = 351,  
DATE = 352,  
DAY = 353,  
DAY_OF_WEEK = 354,  
DE = 355,  
DEBUGGING = 356,
```

```
DECIMAL_POINT = 357,  
DECLARATIVES = 358,  
DEFAULT = 359,  
DELETE = 360,  
DELIMITED = 361,  
DELIMITER = 362,  
DEPENDING = 363,  
DESCENDING = 364,  
DETAIL = 365,  
DISK = 366,  
DISPLAY = 367,  
DIVIDE = 368,  
DIVISION = 369,  
DOWN = 370,  
DUPLICATES = 371,  
DYNAMIC = 372,  
EBCDIC = 373,  
ELSE = 374,  
END = 375,  
END_ACCEPT = 376,  
END_ADD = 377,  
END_CALL = 378,  
END_COMPUTE = 379,  
END_DELETE = 380,  
END_DISPLAY = 381,  
END_DIVIDE = 382,  
END_EVALUATE = 383,  
END_FUNCTION = 384,  
END_IF = 385,  
END_MULTIPLY = 386,  
END_PERFORM = 387,  
END_PROGRAM = 388,  
END_READ = 389,  
END_RETURN = 390,  
END_REWRITE = 391,  
END_SEARCH = 392,  
END_START = 393,  
END_STRING = 394,  
END_SUBTRACT = 395,  
END_UNSTRING = 396,  
END_WRITE = 397,  
ENTRY = 398,  
ENVIRONMENT = 399,  
ENVIRONMENT_NAME = 400,  
ENVIRONMENT_VALUE = 401,  
EOL = 402,  
EOP = 403,  
EOS = 404,  
EQUAL = 405,  
EQUALS = 406,  
ERASE = 407,  
ERROR = 408,  
ESCAPE = 409,  
EVALUATE = 410,  
EVENT_STATUS = 411,  
EXCEPTION = 412,  
EXCLUSIVE = 413,  
EXIT = 414,  
EXTEND = 415,  
EXTERNAL = 416,  
FD = 417,  
FILE_CONTROL = 418,
```

```
FILE_ID = 419,  
FILLER = 420,  
FINAL = 421,  
FIRST = 422,  
FOOTING = 423,  
FOR = 424,  
FOREGROUND_COLOR = 425,  
FOREVER = 426,  
FREE = 427,  
FROM = 428,  
FULL = 429,  
FUNCTION = 430,  
FUNCTION_ID = 431,  
FUNCTION_NAME = 432,  
GE = 433,  
GENERATE = 434,  
GIVING = 435,  
GLOBAL = 436,  
GO = 437,  
GOBACK = 438,  
GREATER = 439,  
GROUP = 440,  
HEADING = 441,  
HIGHLIGHT = 442,  
HIGH_VALUE = 443,  
IDENTIFICATION = 444,  
IF = 445,  
IGNORE = 446,  
IGNORING = 447,  
IN = 448,  
INDEX = 449,  
INDEXED = 450,  
INDICATE = 451,  
INITIALIZE = 452,  
INITIALIZED = 453,  
INITIATE = 454,  
INPUT = 455,  
INPUT_OUTPUT = 456,  
INSPECT = 457,  
INTO = 458,  
INTRINSIC = 459,  
INVALID = 460,  
INVALID_KEY = 461,  
IS = 462,  
I_O = 463,  
I_O_CONTROL = 464,  
JUSTIFIED = 465,  
KEY = 466,  
LABEL = 467,  
LAST = 468,  
LAST_DETAIL = 469,  
LE = 470,  
LEADING = 471,  
LEFT = 472,  
LENGTH = 473,  
LESS = 474,  
LIMIT = 475,  
LIMITS = 476,  
LINAGE = 477,  
LINAGE_COUNTER = 478,  
LINE = 479,  
LINES = 480,
```

LINKAGE = 481,  
LITERAL = 482,  
LOCALE = 483,  
LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485,  
LOCK = 486,  
LOWER\_CASE\_FUNC = 487,  
LOWLIGHT = 488,  
LOW\_VALUE = 489,  
MANUAL = 490,  
MEMORY = 491,  
MERGE = 492,  
MINUS = 493,  
MNEMONIC\_NAME = 494,  
MODE = 495,  
MOVE = 496,  
MULTIPLE = 497,  
MULTIPLY = 498,  
NATIONAL = 499,  
NATIONAL\_EDITED = 500,  
NATIVE = 501,  
NE = 502,  
NEGATIVE = 503,  
NEXT = 504,  
NEXT\_SENTENCE = 505,  
NO = 506,  
NOT = 507,  
NOT\_END = 508,  
NOT\_EOP = 509,  
NOT\_EXCEPTION = 510,  
NOT\_INVALID\_KEY = 511,  
NOT\_OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513,  
NO\_ADVANCING = 514,  
NUMBER = 515,  
NUMBERS = 516,  
NUMERIC = 517,  
NUMERIC\_EDITED = 518,  
NUMVALC\_FUNC = 519,  
OBJECT\_COMPUTER = 520,  
OCCURS = 521,  
OF = 522,  
OFF = 523,  
OMITTED = 524,  
ON = 525,  
ONLY = 526,  
OPEN = 527,  
OPTIONAL = 528,  
OR = 529,  
ORDER = 530,  
ORGANIZATION = 531,  
OTHER = 532,  
OUTPUT = 533,  
OVERFLOW = 534,  
OVERLINE = 535,  
PACKED\_DECIMAL = 536,  
PADDING = 537,  
PAGE = 538,  
PAGE\_FOOTING = 539,  
PAGE\_HEADING = 540,  
PARAGRAPH = 541,  
PERFORM = 542,



```
PICTURE = 543,  
PLUS = 544,  
POINTER = 545,  
POSITION = 546,  
POSITIVE = 547,  
PRESENT = 548,  
PREVIOUS = 549,  
PRINTER = 550,  
PRINTING = 551,  
PROCEDURE = 552,  
PROCEDURES = 553,  
PROCEED = 554,  
PROGRAM = 555,  
PROGRAM_ID = 556,  
PROGRAM_NAME = 557,  
PROGRAM_POINTER = 558,  
PROMPT = 559,  
QUOTE = 560,  
RANDOM = 561,  
RD = 562,  
READ = 563,  
RECORD = 564,  
RECORDING = 565,  
RECORDS = 566,  
RECURSIVE = 567,  
REDEFINES = 568,  
REEL = 569,  
REFERENCE = 570,  
RELATIVE = 571,  
RELEASE = 572,  
REMAINDER = 573,  
REMOVAL = 574,  
RENAMES = 575,  
REPLACING = 576,  
REPORT = 577,  
REPORTING = 578,  
REPORTS = 579,  
REPORT_FOOTING = 580,  
REPORT_HEADING = 581,  
REPOSITORY = 582,  
REQUIRED = 583,  
RESERVE = 584,  
RETURN = 585,  
RETURNING = 586,  
REVERSE_FUNC = 587,  
REVERSE_VIDEO = 588,  
REWIND = 589,  
REWRITE = 590,  
RIGHT = 591,  
ROLLBACK = 592,  
ROUNDED = 593,  
RUN = 594,  
SAME = 595,  
SCREEN = 596,  
SCREEN_CONTROL = 597,  
SCROLL = 598,  
SD = 599,  
SEARCH = 600,  
SECTION = 601,  
SECURE = 602,  
SEGMENT_LIMIT = 603,  
SELECT = 604,
```

```
SEMI_COLON = 605,  
SENTENCE = 606,  
SEPARATE = 607,  
SEQUENCE = 608,  
SEQUENTIAL = 609,  
SET = 610,  
SHARING = 611,  
SIGN = 612,  
SIGNED = 613,  
SIGNED_INT = 614,  
SIGNED_LONG = 615,  
SIGNED_SHORT = 616,  
SIZE = 617,  
SIZE_ERROR = 618,  
SORT = 619,  
SORT_MERGE = 620,  
SOURCE = 621,  
SOURCE_COMPUTER = 622,  
SPACE = 623,  
SPECIAL_NAMES = 624,  
STANDARD = 625,  
STANDARD_1 = 626,  
STANDARD_2 = 627,  
START = 628,  
STATUS = 629,  
STOP = 630,  
STRING = 631,  
SUBSTITUTE_FUNC = 632,  
SUBSTITUTE_CASE_FUNC = 633,  
SUBTRACT = 634,  
SUM = 635,  
SUPPRESS = 636,  
SYMBOLIC = 637,  
SYNCHRONIZED = 638,  
TALLYING = 639,  
TAPE = 640,  
TERMINATE = 641,  
TEST = 642,  
THAN = 643,  
THEN = 644,  
THRU = 645,  
TIME = 646,  
TIMES = 647,  
TO = 648,  
TOK_FALSE = 649,  
TOK_FILE = 650,  
TOK_INITIAL = 651,  
TOK_NULL = 652,  
TOK_TRUE = 653,  
TOP = 654,  
TRAILING = 655,  
TRANSFORM = 656,  
TRIM_FUNCTION = 657,  
TYPE = 658,  
UNDERLINE = 659,  
UNIT = 660,  
UNLOCK = 661,  
UNSIGNED = 662,  
UNSIGNED_INT = 663,  
UNSIGNED_LONG = 664,  
UNSIGNED_SHORT = 665,  
UNSTRING = 666,
```

```
UNTIL = 667,  
UP = 668,  
UPDATE = 669,  
UPON = 670,  
UPON_ARGUMENT_NUMBER = 671,  
UPON_COMMAND_LINE = 672,  
UPON_ENVIRONMENT_NAME = 673,  
UPON_ENVIRONMENT_VALUE = 674,  
UPPER_CASE_FUNC = 675,  
USAGE = 676,  
USE = 677,  
USING = 678,  
VALUE = 679,  
VARYING = 680,  
WAIT = 681,  
WHEN = 682,  
WHEN_COMPILED_FUNC = 683,  
WITH = 684,  
WORD = 685,  
WORDS = 686,  
WORKING_STORAGE = 687,  
WRITE = 688,  
YYYYDDD = 689,  
YYYYMMDD = 690,  
ZERO = 691,  
UNARY_SIGN = 692  
};
```

## 7.9.4 Variable Documentation

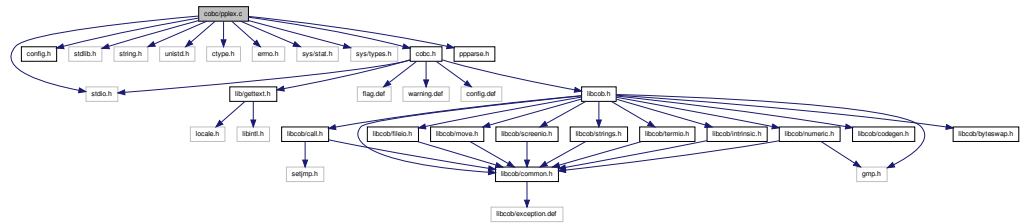
### 7.9.4.1 YYSTYPE yylval

Definition at line 5081 of file parser.c.

## 7.10 cobc/pplex.c File Reference

```
#include <stdio.h>  
#include "config.h"  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <ctype.h>  
#include <errno.h>  
#include <sys/stat.h>  
#include <sys/types.h>  
#include "cobc.h"  
#include "ppparse.h"
```

Include dependency graph for pplex.c:



## Classes

- struct [yy\\_buffer\\_state](#)
- struct **copy\_info**

## Defines

- #define [yy\\_create\\_buffer](#) pp\_create\_buffer
- #define [yy\\_delete\\_buffer](#) pp\_delete\_buffer
- #define [yy\\_scan\\_buffer](#) pp\_scan\_buffer
- #define [yy\\_scan\\_string](#) pp\_scan\_string
- #define [yy\\_scan\\_bytes](#) pp\_scan\_bytes
- #define [yy\\_flex\\_debug](#) pp\_flex\_debug
- #define [yy\\_init\\_buffer](#) pp\_init\_buffer
- #define [yy\\_flush\\_buffer](#) pp\_flush\_buffer
- #define [yy\\_load\\_buffer\\_state](#) pp\_load\_buffer\_state
- #define [yy\\_switch\\_to\\_buffer](#) pp\_switch\_to\_buffer
- #define [yyin](#) ppin
- #define [yyleng](#) ppleng
- #define [yylex](#) pplex
- #define [yyout](#) ppout
- #define [yyrestart](#) pprestart
- #define [yytext](#) pptext
- #define [FLEX\\_SCANNER](#)
- #define [YY\\_FLEX\\_MAJOR\\_VERSION](#) 2
- #define [YY\\_FLEX\\_MINOR\\_VERSION](#) 5
- #define [yyconst](#)
- #define [YY\\_PROTO](#)(proto) ()
- #define [YY\\_NULL](#) 0
- #define [YY\\_SC\\_TO\\_UI](#)(c) ((unsigned int) (unsigned char) c)
- #define [BEGIN](#) yy\_start = 1 + 2 \*
- #define [YY\\_START](#) ((yy\_start - 1) / 2)
- #define [YYSTATE](#) YY\_START

- #define `YY_STATE_EOF`(state) (`YY_END_OF_BUFFER + state + 1`)
- #define `YY_NEW_FILE` `yyrestart( yyin )`
- #define `YY_END_OF_BUFFER_CHAR` 0
- #define `YY_BUF_SIZE` 16384
- #define `EOB_ACT_CONTINUE_SCAN` 0
- #define `EOB_ACT_END_OF_FILE` 1
- #define `EOB_ACT_LAST_MATCH` 2
- #define `yylless`(n)
- #define `unput`(c) `yyunput( c, yytext_ptr )`
- #define `YY_BUFFER_NEW` 0
- #define `YY_BUFFER_NORMAL` 1
- #define `YY_BUFFER_EOF_PENDING` 2
- #define `YY_CURRENT_BUFFER` `yy_current_buffer`
- #define `YY_FLUSH_BUFFER` `yy_flush_buffer( yy_current_buffer )`
- #define `yy_new_buffer` `yy_create_buffer`
- #define `yy_set_interactive`(is\_interactive)
- #define `yy_set_bol`(at\_bol)
- #define `YY_AT_BOL`() (`yy_current_buffer->yy_at_bol`)
- #define `yywrap`() 1
- #define `YY_SKIP_YYWRAP`
- #define `yytext_ptr` `yytext`
- #define `YY_DO_BEFORE_ACTION`
- #define `YY_NUM_RULES` 54
- #define `YY_END_OF_BUFFER` 55
- #define `REJECT` `reject_used_but_not_detected`
- #define `yymore`() `yymore_used_but_not_detected`
- #define `YY_MORE_ADJ` 0
- #define `YY_RESTORE_YY_MORE_OFFSET`
- #define `INITIAL` 0
- #define `YY_NEVER_INTERACTIVE` 1
- #define `YY_INPUT`(buf, result, max\_size) `result = ppinput (buf, max_size);`
- #define `PROCESS_STATE` 1
- #define `COPY_STATE` 2
- #define `PSEUDO_STATE` 3
- #define `YY_NO_PUSH_STATE` 1
- #define `YY_NO_POP_STATE` 1
- #define `YY_NO_TOP_STATE` 1
- #define `YY_READ_BUF_SIZE` 8192
- #define `ECHO` (void) `fwrite( yytext, yyleng, 1, yyout )`
- #define `yyterminate`() `return YY_NULL`
- #define `YY_START_STACK_INCR` 25
- #define `YY_FATAL_ERROR`(msg) `yy_fatal_error( msg )`
- #define `YY_DECL` `int yylex YY_PROTO(( void ))`
- #define `YY_USER_ACTION`
- #define `YY_BREAK` `break;`
- #define `YY_RULE_SETUP`
- #define `YY_EXIT_FAILURE` 2
- #define `yylless`(n)

## Typedefs

- typedef struct [yy\\_buffer\\_state](#) \* [YY\\_BUFFER\\_STATE](#)
- typedef unsigned int [yy\\_size\\_t](#)
- typedef unsigned char [YY\\_CHAR](#)
- typedef int [yy\\_state\\_type](#)

## Functions

- void [yyrestart](#) [YY\\_PROTO](#) ((FILE \*input\_file))
- void [yy\\_switch\\_to\\_buffer](#) [YY\\_PROTO](#) (([YY\\_BUFFER\\_STATE](#) new\_buffer))
- void [yy\\_load\\_buffer\\_state](#) [YY\\_PROTO](#) ((void))
- [YY\\_BUFFER\\_STATE](#) [yy\\_create\\_buffer](#) [YY\\_PROTO](#) ((FILE \*file, int size))
- void [yy\\_delete\\_buffer](#) [YY\\_PROTO](#) (([YY\\_BUFFER\\_STATE](#) b))
- void [yy\\_init\\_buffer](#) [YY\\_PROTO](#) (([YY\\_BUFFER\\_STATE](#) b, FILE \*file))
- [YY\\_BUFFER\\_STATE](#) [yy\\_scan\\_buffer](#) [YY\\_PROTO](#) ((char \*base, [yy\\_size\\_t](#) size))
- [YY\\_BUFFER\\_STATE](#) [yy\\_scan\\_string](#) [YY\\_PROTO](#) ((yyconst char \*yy\_str))
- [YY\\_BUFFER\\_STATE](#) [yy\\_scan\\_bytes](#) [YY\\_PROTO](#) ((yyconst char \*bytes, int len))

## Variables

- int [yyleng](#)
- FILE \* [yyin](#) = (FILE \*) 0
- FILE \* [yyout](#) = (FILE \*) 0
- char \* [yytext](#)
- register char \* [yy\\_bp](#)
- int [size](#)
- FILE \* [file](#)
- int [len](#)

### 7.10.1 Define Documentation

#### 7.10.1.1 #define BEGIN yy\_start = 1 + 2 \*

Definition at line 97 of file pplex.c.

#### 7.10.1.2 #define COPY\_STATE 2

Definition at line 710 of file pplex.c.

#### 7.10.1.3 #define ECHO (void) fwrite( yytext, yyleng, 1, yyout )

Definition at line 793 of file pplex.c.

7.10.1.4 **#define EOB\_ACT\_CONTINUE\_SCAN 0**

Definition at line 122 of file pplex.c.

7.10.1.5 **#define EOB\_ACT\_END\_OF\_FILE 1**

Definition at line 123 of file pplex.c.

7.10.1.6 **#define EOB\_ACT\_LAST\_MATCH 2**

Definition at line 124 of file pplex.c.

7.10.1.7 **#define FLEX\_SCANNER**

Definition at line 25 of file pplex.c.

7.10.1.8 **#define INITIAL 0**

Definition at line 639 of file pplex.c.

7.10.1.9 **#define PROCESS\_STATE 1**

Definition at line 709 of file pplex.c.

7.10.1.10 **#define PSEUDO\_STATE 3**

Definition at line 711 of file pplex.c.

7.10.1.11 **#define REJECT reject\_used\_but\_not\_detected**

Definition at line 633 of file pplex.c.

7.10.1.12 **#define unput( c ) yyunput( c, yytext\_ptr )**

Definition at line 153 of file pplex.c.

7.10.1.13 **#define YY\_AT\_BOL( ) (yy\_current\_buffer->yy\_at\_bol)**

Definition at line 280 of file pplex.c.

7.10.1.14 `#define YY_BREAK break;`

Definition at line 852 of file pplex.c.

7.10.1.15 `#define YY_BUF_SIZE 16384`

Definition at line 115 of file pplex.c.

7.10.1.16 `#define YY_BUFFER_EOF_PENDING 2`

Definition at line 216 of file pplex.c.

7.10.1.17 `#define YY_BUFFER_NEW 0`

Definition at line 204 of file pplex.c.

7.10.1.18 `#define YY_BUFFER_NORMAL 1`

Definition at line 205 of file pplex.c.

7.10.1.19 `#define yy_create_buffer pp_create_buffer`

Definition at line 1 of file pplex.c.

7.10.1.20 `#define YY_CURRENT_BUFFER yy_current_buffer`

Definition at line 225 of file pplex.c.

7.10.1.21 `#define YY_DECL int yylex YY_PROTO(( void ))`

Definition at line 840 of file pplex.c.

7.10.1.22 `#define yy_delete_buffer pp_delete_buffer`

Definition at line 2 of file pplex.c.

7.10.1.23 `#define YY_DO_BEFORE_ACTION`

**Value:**

```
yytext_ptr = yy_bp; \  
  yy_leng = (int) (yy_cp - yy_bp); \  
  yy_hold_char = *yy_cp; \  
  
```



```
*yy_cp = '\0'; \  
yy_c_buf_p = yy_cp;
```

Definition at line 299 of file pplex.c.

7.10.1.24 **#define YY\_END\_OF\_BUFFER 55**

Definition at line 307 of file pplex.c.

7.10.1.25 **#define YY\_END\_OF\_BUFFER\_CHAR 0**

Definition at line 112 of file pplex.c.

7.10.1.26 **#define YY\_EXIT\_FAILURE 2**

7.10.1.27 **#define YY\_FATAL\_ERROR( msg ) yy\_fatal\_error( msg )**

Definition at line 833 of file pplex.c.

7.10.1.28 **#define yy\_flex\_debug pp\_flex\_debug**

Definition at line 6 of file pplex.c.

7.10.1.29 **#define YY\_FLEX\_MAJOR\_VERSION 2**

Definition at line 26 of file pplex.c.

7.10.1.30 **#define YY\_FLEX\_MINOR\_VERSION 5**

Definition at line 27 of file pplex.c.

7.10.1.31 **#define yy\_flush\_buffer pp\_flush\_buffer**

Definition at line 8 of file pplex.c.

7.10.1.32 **#define YY\_FLUSH\_BUFFER yy\_flush\_buffer( yy\_current\_buffer )**

Definition at line 254 of file pplex.c.

7.10.1.33 **#define yy\_init\_buffer pp\_init\_buffer**

Definition at line 7 of file pplex.c.

7.10.1.34 `#define YY_INPUT( buf, result, max_size ) result = ppinput (buf, max_size);`

Definition at line 701 of file pplex.c.

7.10.1.35 `#define yy_load_buffer_state pp_load_buffer_state`

Definition at line 9 of file pplex.c.

7.10.1.36 `#define YY_MORE_ADJ 0`

Definition at line 635 of file pplex.c.

7.10.1.37 `#define YY_NEVER_INTERACTIVE 1`

Definition at line 659 of file pplex.c.

7.10.1.38 `#define yy_new_buffer yy_create_buffer`

Definition at line 264 of file pplex.c.

7.10.1.39 `#define YY_NEW_FILE yyrestart( yyin )`

Definition at line 110 of file pplex.c.

7.10.1.40 `#define YY_NO_POP_STATE 1`

Definition at line 763 of file pplex.c.

7.10.1.41 `#define YY_NO_PUSH_STATE 1`

Definition at line 762 of file pplex.c.

7.10.1.42 `#define YY_NO_TOP_STATE 1`

Definition at line 764 of file pplex.c.

7.10.1.43 `#define YY_NULL 0`

Definition at line 84 of file pplex.c.

**7.10.1.44 #define YY\_NUM\_RULES 54**

Definition at line 306 of file pplex.c.

**7.10.1.45 #define YY\_PROTO( proto )()**

Definition at line 80 of file pplex.c.

**7.10.1.46 #define YY\_READ\_BUF\_SIZE 8192**

Definition at line 784 of file pplex.c.

**7.10.1.47 #define YY\_RESTORE\_YY\_MORE\_OFFSET**

Definition at line 636 of file pplex.c.

**7.10.1.48 #define YY\_RULE\_SETUP****Value:**

```
if ( yyleng > 0 ) \  
    yy_current_buffer->yy_at_bol = \  
        (yytext[yyleng - 1] == '\n'); \  
    YY_USER_ACTION
```

Definition at line 855 of file pplex.c.

**7.10.1.49 #define YY\_SC\_TO\_UI( c )((unsigned int) (unsigned char) c)**

Definition at line 91 of file pplex.c.

**7.10.1.50 #define yy\_scan\_buffer pp\_scan\_buffer**

Definition at line 3 of file pplex.c.

**7.10.1.51 #define yy\_scan\_bytes pp\_scan\_bytes**

Definition at line 5 of file pplex.c.

**7.10.1.52 #define yy\_scan\_string pp\_scan\_string**

Definition at line 4 of file pplex.c.

**7.10.1.53 #define yy\_set\_bol( *at\_bol* )****Value:**

```
{ \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_at_bol = at_bol; \
}
```

Definition at line 273 of file pplex.c.

**7.10.1.54 #define yy\_set\_interactive( *is\_interactive* )****Value:**

```
{ \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_is_interactive = is_interactive; \
}
```

Definition at line 266 of file pplex.c.

**7.10.1.55 #define YY\_SKIP\_YYWRAP**

Definition at line 284 of file pplex.c.

**7.10.1.56 #define YY\_START ((yy\_start - 1) / 2)**

Definition at line 103 of file pplex.c.

**7.10.1.57 #define YY\_START\_STACK\_INCR 25**

Definition at line 828 of file pplex.c.

**7.10.1.58 #define YY\_STATE\_EOF( *state* ) (YY\_END\_OF\_BUFFER + state + 1)**

Definition at line 107 of file pplex.c.

**7.10.1.59 #define yy\_switch\_to\_buffer pp\_switch\_to\_buffer**

Definition at line 10 of file pplex.c.

**7.10.1.60 #define YY\_USER\_ACTION**

Definition at line 847 of file pplex.c.

**7.10.1.61 #define yyconst**

Definition at line 73 of file pplex.c.

**7.10.1.62 FILE \* yyin ppin**

Definition at line 11 of file pplex.c.

**7.10.1.63 int yyleng ppleng**

Definition at line 12 of file pplex.c.

**7.10.1.64 #define yyless( n )**

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        yytext[yyleng] = yy_hold_char; \
        yy_c_buf_p = yytext + n; \
        yy_hold_char = *yy_c_buf_p; \
        *yy_c_buf_p = '\0'; \
        yyleng = n; \
    } \
while ( 0 )
```

Definition at line 142 of file pplex.c.

**7.10.1.65 #define yyless( n )**

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        *yy_cp = yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yy_c_buf_p = yy_cp = yy_bp + n - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
while ( 0 )
```

Definition at line 142 of file pplex.c.

**7.10.1.66 #define yylex pplex**

Definition at line 13 of file pplex.c.

7.10.1.67 `#define yymore( ) yymore_used_but_not_detected`

Definition at line 634 of file pplex.c.

7.10.1.68 `FILE * yyout ppout`

Definition at line 14 of file pplex.c.

7.10.1.69 `#define yyrestart pprestart`

Definition at line 15 of file pplex.c.

7.10.1.70 `#define YYSTATE YY_START`

Definition at line 104 of file pplex.c.

7.10.1.71 `#define yyterminate( ) return YY_NULL`

Definition at line 823 of file pplex.c.

7.10.1.72 `char * yytext pptext`

Definition at line 16 of file pplex.c.

7.10.1.73 `#define yytext_ptr yytext`

Definition at line 289 of file pplex.c.

7.10.1.74 `#define yywrap( ) 1`

Definition at line 283 of file pplex.c.

## 7.10.2 Typedef Documentation

7.10.2.1 `typedef struct yy_buffer_state* YY_BUFFER_STATE`

Definition at line 117 of file pplex.c.

7.10.2.2 `typedef unsigned char YY_CHAR`

Definition at line 285 of file pplex.c.

### 7.10.2.3 typedef unsigned int yy\_size\_t

Definition at line 159 of file pplex.c.

### 7.10.2.4 typedef int yy\_state\_type

Definition at line 287 of file pplex.c.

## 7.10.3 Function Documentation

### 7.10.3.1 YY\_BUFFER\_STATE yy\_scan\_string YY\_PROTO ( (yyconst char \*yy\_str) )

### 7.10.3.2 YY\_BUFFER\_STATE yy\_scan\_buffer YY\_PROTO ( (char \*base, yy\_size\_t size) )

### 7.10.3.3 YY\_BUFFER\_STATE yy\_scan\_bytes YY\_PROTO ( (yyconst char \*bytes, int len) )

### 7.10.3.4 void yy\_switch\_to\_buffer YY\_PROTO ( (YY\_BUFFER\_STATE new\_buffer) )

### 7.10.3.5 static int input YY\_PROTO ( (void) )

### 7.10.3.6 void yy\_init\_buffer YY\_PROTO ( (YY\_BUFFER\_STATE b, FILE \*file) )

### 7.10.3.7 void yy\_flush\_buffer YY\_PROTO ( (YY\_BUFFER\_STATE b) )

### 7.10.3.8 void yyrestart YY\_PROTO ( (FILE \*input\_file) )

### 7.10.3.9 YY\_BUFFER\_STATE yy\_create\_buffer YY\_PROTO ( (FILE \*file, int size) )

## 7.10.4 Variable Documentation

### 7.10.4.1 FILE\* file

Definition at line 1807 of file pplex.c.

### 7.10.4.2 int len

Definition at line 1929 of file pplex.c.

### 7.10.4.3 yy\_size\_t size

Definition at line 1718 of file pplex.c.

### 7.10.4.4 register char\* yy\_bp

Definition at line 1570 of file pplex.c.

#### 7.10.4.5 FILE\* yyin = (FILE \*) 0

Definition at line 286 of file pplex.c.

#### 7.10.4.6 int yyleng

Definition at line 234 of file pplex.c.

#### 7.10.4.7 FILE \* yyout = (FILE \*) 0

Definition at line 286 of file pplex.c.

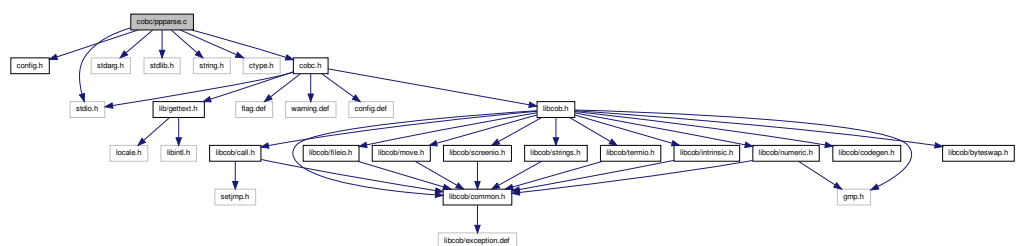
#### 7.10.4.8 char\* yytext

Definition at line 637 of file pplex.c.

## 7.11 cobc/ppparse.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobc.h"
```

Include dependency graph for ppparse.c:



## Classes

- union [YYSTYPE](#)



- union [yyalloc](#)

## Defines

- #define [YBISON](#) 1
- #define [YYSKELETON\\_NAME](#) "yacc.c"
- #define [YYPURE](#) 0
- #define [YYLSP\\_NEEDED](#) 0
- #define [yyparse](#) ppparse
- #define [yylex](#) pplex
- #define [yyerror](#) pperror
- #define [yylval](#) pplval
- #define [yychar](#) ppchar
- #define [yydebug](#) ppdebug
- #define [yynerrs](#) ppnerrs
- #define [YYTOKENTYPE](#)
- #define [TOKEN\\_EOF](#) 0
- #define [COPY](#) 258
- #define [REPLACE](#) 259
- #define [SUPPRESS](#) 260
- #define [PRINTING](#) 261
- #define [REPLACING](#) 262
- #define [OFF](#) 263
- #define [IN](#) 264
- #define [OF](#) 265
- #define [BY](#) 266
- #define [EQEQ](#) 267
- #define [TOKEN](#) 268
- #define [YYDEBUG](#) 1
- #define [YYERROR\\_VERBOSE](#) 1
- #define [pperror](#) cb\_error
- #define [YYERROR\\_VERBOSE](#) 1
- #define [yystype](#) YYSTYPE
- #define [YYSTYPE\\_IS\\_DECLARED](#) 1
- #define [YYSTYPE\\_IS\\_TRIVIAL](#) 1
- #define [YYSTACK\\_ALLOC](#) malloc
- #define [YYSTACK\\_FREE](#) free
- #define [YYSTACK\\_GAP\\_MAXIMUM](#) (sizeof (union [yyalloc](#)) - 1)
- #define [YYSTACK\\_BYTES](#)(N)
- #define [YYCOPY](#)(To, From, Count)
- #define [YYSTACK\\_RELOCATE](#)(Stack)
- #define [YYFINAL](#) 2
- #define [YYLAST](#) 41
- #define [YYNTOKENS](#) 17
- #define [YYNNTS](#) 15
- #define [YYNRULES](#) 31

- #define `YYNSTATES` 50
- #define `YYUNDEFTOK` 2
- #define `YYMAXUTOK` 268
- #define `YYTRANSLATE`(YYX) ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)
- #define `YYPACT_NINF` -13
- #define `YYTABLE_NINF` -1
- #define `YYSIZE_T` unsigned int
- #define `yyerrok` (yyerrstatus = 0)
- #define `yyclearin` (yychar = YYEMPTY)
- #define `YYEMPTY` (-2)
- #define `YYEOF` 0
- #define `YYACCEPT` goto yyacceptlab
- #define `YYABORT` goto yyabortlab
- #define `YYERROR` goto yyerrlab1
- #define `YYFAIL` goto yyerrlab
- #define `YYRECOVERING`() (!yyerrstatus)
- #define `YYBACKUP`(Token, Value)
- #define `YYTERROR` 1
- #define `YYERRCODE` 256
- #define `YYLLOC_DEFAULT`(Current, Rhs, N)
- #define `YYLEX` yylex ()
- #define `YYFPRINTF` fprintf
- #define `YYDPRINTF`(Args)
- #define `YYDSYMPRINT`(Args)
- #define `YYDSYMPRINTF`(Title, Token, Value, Location)
- #define `YY_STACK_PRINT`(Bottom, Top)
- #define `YY_REDUCE_PRINT`(Rule)
- #define `YYINITDEPTH` 200
- #define `YYMAXDEPTH` 10000
- #define `YYPOPSTACK` (yyvsp--, yyssp--)

## Typedefs

- typedef union `YYSTYPE` `YYSTYPE`
- typedef short `yysigned_char`

## Enumerations

- enum `yytokentype` {  
`TOKEN_EOF` = 0, `ACCEPT` = 258, `ACCESS` = 259, `ADD` = 260,  
`ADDRESS` = 261, `ADVANCING` = 262, `AFTER` = 263, `ALL` = 264,  
`ALLOCATE` = 265, `ALPHABET` = 266, `ALPHABETIC` = 267, `ALPHABETIC_`  
`LOWER` = 268,

ALPHABETIC\_UPPER = 269, ALPHANUMERIC = 270, ALPHANUMERIC\_EDITED = 271, ALSO = 272,  
ALTER = 273, ALTERNATE = 274, AND = 275, ANY = 276,  
ARE = 277, AREA = 278, ARGUMENT\_NUMBER = 279, ARGUMENT\_VALUE = 280,  
AS = 281, ASCENDING = 282, ASSIGN = 283, AT = 284,  
AUTO = 285, AUTOMATIC = 286, BACKGROUND\_COLOR = 287, BASED = 288,  
BEFORE = 289, BELL = 290, BINARY = 291, BINARY\_C\_LONG = 292,  
BINARY\_CHAR = 293, BINARY\_DOUBLE = 294, BINARY\_LONG = 295, BINARY\_SHORT = 296,  
BLANK = 297, BLANK\_LINE = 298, BLANK\_SCREEN = 299, BLINK = 300,  
BLOCK = 301, BOTTOM = 302, BY = 303, BYTE\_LENGTH = 304,  
CALL = 305, CANCEL = 306, CH = 307, CHAINING = 308,  
CHARACTER = 309, CHARACTERS = 310, CLASS = 311, CLOSE = 312,  
CODE = 313, CODE\_SET = 314, COLLATING = 315, COL = 316,  
COLS = 317, COLUMN = 318, COLUMNS = 319, COMMA = 320,  
COMMAND\_LINE = 321, COMMA\_DELIM = 322, COMMIT = 323, COMMON = 324,  
COMP = 325, COMPUTE = 326, COMP\_1 = 327, COMP\_2 = 328,  
COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335, CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING = 343, CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE = 360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING = 364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE = 380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM = 388,

END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH = 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING = 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION = 444,  
IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOWLIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED = 500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUMBERS = 516,

NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL = 536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID = 556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE = 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,  
REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING = 580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE = 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT = 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES = 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS = 636,

SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
COMMAND\_LINE = 672,  
UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
UPPER\_CASE\_FUNC = 675, USAGE = 676,  
USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
TOKEN\_EOF = 0, ACCEPT = 258, ACCESS = 259, ADD = 260,  
ADDRESS = 261, ADVANCING = 262, AFTER = 263, ALL = 264,  
ALLOCATE = 265, ALPHABET = 266, ALPHABETIC = 267, ALPHABETIC\_  
LOWER = 268,  
ALPHABETIC\_UPPER = 269, ALPHANUMERIC = 270, ALPHANUMERIC\_EDITED  
= 271, ALSO = 272,  
ALTER = 273, ALTERNATE = 274, AND = 275, ANY = 276,  
ARE = 277, AREA = 278, ARGUMENT\_NUMBER = 279, ARGUMENT\_VALUE  
= 280,  
AS = 281, ASCENDING = 282, ASSIGN = 283, AT = 284,  
AUTO = 285, AUTOMATIC = 286, BACKGROUND\_COLOR = 287, BASED = 288,  
BEFORE = 289, BELL = 290, BINARY = 291, BINARY\_C\_LONG = 292,  
BINARY\_CHAR = 293, BINARY\_DOUBLE = 294, BINARY\_LONG = 295, BINARY\_  
SHORT = 296,  
BLANK = 297, BLANK\_LINE = 298, BLANK\_SCREEN = 299, BLINK = 300,  
BLOCK = 301, BOTTOM = 302, BY = 303, BYTE\_LENGTH = 304,  
CALL = 305, CANCEL = 306, CH = 307, CHAINING = 308,  
CHARACTER = 309, CHARACTERS = 310, CLASS = 311, CLOSE = 312,  
CODE = 313, CODE\_SET = 314, COLLATING = 315, COL = 316,  
COLS = 317, COLUMN = 318, COLUMNS = 319, COMMA = 320,  
COMMAND\_LINE = 321, COMMA\_DELIM = 322, COMMIT = 323, COMMON =  
324,  
COMP = 325, COMPUTE = 326, COMP\_1 = 327, COMP\_2 = 328,

COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING =  
343, CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE =  
360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING =  
364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE =  
380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_-  
FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM  
= 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH  
= 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING  
= 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_-  
NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,  
FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION =  
444,

IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOW-  
LIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED =  
500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_-  
OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUM-  
BERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_-  
COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL =  
536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,  
PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID =  
556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE  
= 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,



REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING =  
580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE  
= 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_  
VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT  
= 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
COMMAND\_LINE = 672,  
UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
UPPER\_CASE\_FUNC = 675, USAGE = 676,  
USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
TOKEN\_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,

```
PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,  
OF = 265, BY = 266, EQEQ = 267, TOKEN = 268,  
TOKEN_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,  
PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,  
OF = 265, BY = 266, EQEQ = 267, TOKEN = 268 }
```

## Functions

- int `yyparse` ()

## Variables

- int `yydebug`
- int `yychar`
- YYSTYPE `yylval`
- int `yynerres`

### 7.11.1 Define Documentation

#### 7.11.1.1 #define BY 266

Definition at line 88 of file `ppparse.c`.

#### 7.11.1.2 #define COPY 258

Definition at line 80 of file `ppparse.c`.

#### 7.11.1.3 #define EQEQ 267

Definition at line 89 of file `ppparse.c`.

#### 7.11.1.4 #define IN 264

Definition at line 86 of file `ppparse.c`.

#### 7.11.1.5 #define OF 265

Definition at line 87 of file `ppparse.c`.

#### 7.11.1.6 #define OFF 263

Definition at line 85 of file `ppparse.c`.

**7.11.1.7 #define perror cb\_error**

Definition at line 110 of file ppparse.c.

**7.11.1.8 #define PRINTING 261**

Definition at line 83 of file ppparse.c.

**7.11.1.9 #define REPLACE 259**

Definition at line 81 of file ppparse.c.

**7.11.1.10 #define REPLACING 262**

Definition at line 84 of file ppparse.c.

**7.11.1.11 #define SUPPRESS 260**

Definition at line 82 of file ppparse.c.

**7.11.1.12 #define TOKEN 268**

Definition at line 90 of file ppparse.c.

**7.11.1.13 #define TOKEN\_EOF 0**

Definition at line 79 of file ppparse.c.

**7.11.1.14 #define YY\_REDUCE\_PRINT( *Rule* )****Value:**

```
do {
    if (yydebug)
        yy_reduce_print (Rule);
} while (0)
```

Definition at line 604 of file ppparse.c.

**7.11.1.15 #define YY\_STACK\_PRINT( *Bottom*, *Top* )****Value:**

```
do {
    if (yydebug)
        yy_stack_print ((Bottom), (Top));
} while (0)
```

Definition at line 574 of file ppparse.c.

#### 7.11.1.16 #define YYABORT goto yyabortlab

Definition at line 472 of file ppparse.c.

#### 7.11.1.17 #define YYACCEPT goto yyacceptlab

Definition at line 471 of file ppparse.c.

#### 7.11.1.18 #define YYBACKUP( Token, Value )

##### Value:

```
do
    if (yychar == YYEMPTY && yylen == 1)
    {
        yychar = (Token);
        yylval = (Value);
        yytoken = YYTRANSLATE (yychar);
        YYPOPSTACK;
        goto yybackup;
    }
    else
    {
        yyerror ("syntax error: cannot back up");\
        YYERROR;
    }
while (0)
```

Definition at line 483 of file ppparse.c.

#### 7.11.1.19 #define YYBISON 1

Definition at line 37 of file ppparse.c.

#### 7.11.1.20 #define yychar ppchar

Definition at line 54 of file ppparse.c.

#### 7.11.1.21 #define yyclearin (yychar = YYEMPTY)

Definition at line 467 of file ppparse.c.

**7.11.1.22 #define YYCOPY( To, From, Count )****Value:**

```
do
    {
        register YYSIZE_T yyi;
        for (yyi = 0; yyi < (Count); yyi++)
            (To)[yyi] = (From)[yyi];
    }
    while (0)
```

Definition at line 213 of file ppparse.c.

**7.11.1.23 #define yydebug ppdebug**

Definition at line 55 of file ppparse.c.

**7.11.1.24 #define YYDEBUG 1**

Definition at line 108 of file ppparse.c.

**7.11.1.25 #define YYDPRINTF( Args )****Value:**

```
do {
    if (yydebug)
        YYFPRINTF Args;
} while (0)
```

Definition at line 530 of file ppparse.c.

**7.11.1.26 #define YYDSYMPRINT( Args )****Value:**

```
do {
    if (yydebug)
        ysymprint Args;
} while (0)
```

Definition at line 536 of file ppparse.c.

**7.11.1.27 #define YYDSYMPRINTF( Title, Token, Value, Location )****Value:**

```

do {
    if (yydebug)
    {
        YYFPRINTF (stderr, "%s ", Title);
        ysymprint (stderr,
                   Token, Value);
        YYFPRINTF (stderr, "\n");
    }
} while (0)

```

Definition at line 542 of file ppparse.c.

#### 7.11.1.28 #define YYEMPTY (-2)

Definition at line 468 of file ppparse.c.

#### 7.11.1.29 #define YYEOF 0

Definition at line 469 of file ppparse.c.

#### 7.11.1.30 #define YYERRCODE 256

Definition at line 501 of file ppparse.c.

#### 7.11.1.31 #define yyerrok (yyerrstatus = 0)

Definition at line 466 of file ppparse.c.

#### 7.11.1.32 #define yyerror perror

Definition at line 52 of file ppparse.c.

#### 7.11.1.33 #define YYERROR goto yyerrlab1

Definition at line 473 of file ppparse.c.

#### 7.11.1.34 #define YYERROR\_VERBOSE 1

Definition at line 127 of file ppparse.c.

#### 7.11.1.35 #define YYERROR\_VERBOSE 1

Definition at line 127 of file ppparse.c.

7.11.1.36 `#define YYFAIL goto yyerrlab`

Definition at line 479 of file ppparse.c.

7.11.1.37 `#define YYFINAL 2`

Definition at line 249 of file ppparse.c.

7.11.1.38 `#define YYFPRINTF fprintf`

Definition at line 527 of file ppparse.c.

7.11.1.39 `#define YYINITDEPTH 200`

Definition at line 624 of file ppparse.c.

7.11.1.40 `#define YYLAST 41`

Definition at line 251 of file ppparse.c.

7.11.1.41 `#define yylex pplex`

Definition at line 51 of file ppparse.c.

7.11.1.42 `#define YYLEX yylex ()`

Definition at line 519 of file ppparse.c.

7.11.1.43 `#define YYLLOC_DEFAULT( Current, Rhs, N )`**Value:**

```
Current.first_line   = Rhs[1].first_line;      \
Current.first_column = Rhs[1].first_column;    \
Current.last_line    = Rhs[N].last_line;       \
Current.last_column  = Rhs[N].last_column;
```

Definition at line 507 of file ppparse.c.

7.11.1.44 `#define YYLSP_NEEDED 0`

Definition at line 46 of file ppparse.c.

**7.11.1.45 #define yylval pplval**

Definition at line 53 of file ppparse.c.

**7.11.1.46 #define YYMAXDEPTH 10000**

Definition at line 639 of file ppparse.c.

**7.11.1.47 #define YYMAXUTOK 268**

Definition at line 264 of file ppparse.c.

**7.11.1.48 #define yynerrs ppnerrs**

Definition at line 56 of file ppparse.c.

**7.11.1.49 #define YYNNTS 15**

Definition at line 256 of file ppparse.c.

**7.11.1.50 #define YYNRULES 31**

Definition at line 258 of file ppparse.c.

**7.11.1.51 #define YYNSTATES 50**

Definition at line 260 of file ppparse.c.

**7.11.1.52 #define YYNTOKENS 17**

Definition at line 254 of file ppparse.c.

**7.11.1.53 #define YYPACT\_NINF -13**

Definition at line 399 of file ppparse.c.

**7.11.1.54 int yyparse ppparse**

Definition at line 50 of file ppparse.c.



7.11.1.55 `#define YYPOPSTACK (yyvsp--, yysp--)`

7.11.1.56 `#define YYPURE 0`

Definition at line 43 of file ppparse.c.

7.11.1.57 `#define YYRECOVERING( ) (!yyerrstatus)`

Definition at line 481 of file ppparse.c.

7.11.1.58 `#define YYSIZE_T unsigned int`

Definition at line 463 of file ppparse.c.

7.11.1.59 `#define YYSKELETON_NAME "yacc.c"`

Definition at line 40 of file ppparse.c.

7.11.1.60 `#define YYSTACK_ALLOC malloc`

Definition at line 180 of file ppparse.c.

7.11.1.61 `#define YYSTACK_BYTES( N )`

**Value:**

```
((N) * (sizeof (short) + sizeof (YYSTYPE))
+ YYSTACK_GAP_MAXIMUM)
```

Definition at line 202 of file ppparse.c.

7.11.1.62 `#define YYSTACK_FREE free`

Definition at line 181 of file ppparse.c.

7.11.1.63 `#define YYSTACK_GAP_MAXIMUM (sizeof (union yyalloc) - 1)`

Definition at line 198 of file ppparse.c.

7.11.1.64 `#define YYSTACK_RELOCATE( Stack )`

**Value:**

```

do
    {
        YYSIZE_T yynewbytes;
        YCOPY (&yyptr->Stack, Stack, yysize);
        Stack = &yyptr->Stack;
        yynewbytes = yystacksize * sizeof (*Stack) + YYSTACK_GAP_MAXIMUM;
        yyptr += yynewbytes / sizeof (*yyptr);
    }
while (0)

```

Definition at line 229 of file ppparse.c.

#### 7.11.1.65 #define YYSTYPE YYSTYPE

Definition at line 141 of file ppparse.c.

#### 7.11.1.66 #define YYSTYPE\_IS\_DECLARED 1

Definition at line 142 of file ppparse.c.

#### 7.11.1.67 #define YYSTYPE\_IS\_TRIVIAL 1

Definition at line 143 of file ppparse.c.

#### 7.11.1.68 #define YYTABLE\_NINF -1

Definition at line 420 of file ppparse.c.

#### 7.11.1.69 #define YYTERROR 1

Definition at line 500 of file ppparse.c.

#### 7.11.1.70 #define YYTOKENTYPE

Definition at line 61 of file ppparse.c.

#### 7.11.1.71 #define YYTRANSLATE( YYX ) ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)

Definition at line 266 of file ppparse.c.

#### 7.11.1.72 #define YYUNDEFTOK 2

Definition at line 263 of file ppparse.c.

## 7.11.2 Typedef Documentation

### 7.11.2.1 typedef short ysigned\_char

Definition at line 245 of file ppparse.c.

### 7.11.2.2 typedef union YYSTYPE YYSTYPE

## 7.11.3 Enumeration Type Documentation

### 7.11.3.1 enum yytokentype

Enumerator:

*TOKEN\_EOF*  
*ACCEPT*  
*ACCESS*  
*ADD*  
*ADDRESS*  
*ADVANCING*  
*AFTER*  
*ALL*  
*ALLOCATE*  
*ALPHABET*  
*ALPHABETIC*  
*ALPHABETIC\_LOWER*  
*ALPHABETIC\_UPPER*  
*ALPHANUMERIC*  
*ALPHANUMERIC\_EDITED*  
*ALSO*  
*ALTER*  
*ALTERNATE*  
*AND*  
*ANY*  
*ARE*  
*AREA*  
*ARGUMENT\_NUMBER*  
*ARGUMENT\_VALUE*  
*AS*  
*ASCENDING*  
*ASSIGN*

**AT**  
**AUTO**  
**AUTOMATIC**  
**BACKGROUND\_COLOR**  
**BASED**  
**BEFORE**  
**BELL**  
**BINARY**  
**BINARY\_C\_LONG**  
**BINARY\_CHAR**  
**BINARY\_DOUBLE**  
**BINARY\_LONG**  
**BINARY\_SHORT**  
**BLANK**  
**BLANK\_LINE**  
**BLANK\_SCREEN**  
**BLINK**  
**BLOCK**  
**BOTTOM**  
**BY**  
**BYTE\_LENGTH**  
**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**  
**COMMAND\_LINE**

*COMMA\_DELIM*  
*COMMIT*  
*COMMON*  
*COMP*  
*COMPUTE*  
*COMP\_1*  
*COMP\_2*  
*COMP\_3*  
*COMP\_4*  
*COMP\_5*  
*COMP\_X*  
*CONCATENATE\_FUNC*  
*CONFIGURATION*  
*CONSTANT*  
*CONTAINS*  
*CONTENT*  
*CONTINUE*  
*CONTROL*  
*CONTROLS*  
*CONTROL\_FOOTING*  
*CONTROL\_HEADING*  
*CONVERTING*  
*CORRESPONDING*  
*COUNT*  
*CRT*  
*CURRENCY*  
*CURRENT\_DATE\_FUNC*  
*CURSOR*  
*CYCLE*  
*DATA*  
*DATE*  
*DAY*  
*DAY\_OF\_WEEK*  
*DE*  
*DEBUGGING*  
*DECIMAL\_POINT*  
*DECLARATIVES*  
*DEFAULT*

*DELETE*  
*DELIMITED*  
*DELIMITER*  
*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*

**ENTRY**  
**ENVIRONMENT**  
**ENVIRONMENT\_NAME**  
**ENVIRONMENT\_VALUE**  
**EOL**  
**EOP**  
**EOS**  
**EQUAL**  
**EQUALS**  
**ERASE**  
**ERROR**  
**ESCAPE**  
**EVALUATE**  
**EVENT\_STATUS**  
**EXCEPTION**  
**EXCLUSIVE**  
**EXIT**  
**EXTEND**  
**EXTERNAL**  
**FD**  
**FILE\_CONTROL**  
**FILE\_ID**  
**FILLER**  
**FINAL**  
**FIRST**  
**FOOTING**  
**FOR**  
**BACKGROUND\_COLOR**  
**FOREVER**  
**FREE**  
**FROM**  
**FULL**  
**FUNCTION**  
**FUNCTION\_ID**  
**FUNCTION\_NAME**  
**GE**  
**GENERATE**  
**GIVING**

**GLOBAL**  
**GO**  
**GOBACK**  
**GREATER**  
**GROUP**  
**HEADING**  
**HIGHLIGHT**  
**HIGH\_VALUE**  
**IDENTIFICATION**  
**IF**  
**IGNORE**  
**IGNORING**  
**IN**  
**INDEX**  
**INDEXED**  
**INDICATE**  
**INITIALIZE**  
**INITIALIZED**  
**INITIATE**  
**INPUT**  
**INPUT\_OUTPUT**  
**INSPECT**  
**INTO**  
**INTRINSIC**  
**INVALID**  
**INVALID\_KEY**  
**IS**  
**I\_O**  
**I\_O\_CONTROL**  
**JUSTIFIED**  
**KEY**  
**LABEL**  
**LAST**  
**LAST\_DETAIL**  
**LE**  
**LEADING**  
**LEFT**  
**LENGTH**



*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*

*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*  
*OR*  
*ORDER*  
*ORGANIZATION*  
*OTHER*  
*OUTPUT*  
*OVERFLOW*  
*OVERLINE*  
*PACKED\_DECIMAL*  
*PADDING*  
*PAGE*  
*PAGE\_FOOTING*  
*PAGE\_HEADING*  
*PARAGRAPH*  
*PERFORM*  
*PICTURE*  
*PLUS*  
*POINTER*  
*POSITION*  
*POSITIVE*  
*PRESENT*  
*PREVIOUS*

*PRINTER*  
*PRINTING*  
*PROCEDURE*  
*PROCEDURES*  
*PROCEED*  
*PROGRAM*  
*PROGRAM\_ID*  
*PROGRAM\_NAME*  
*PROGRAM\_POINTER*  
*PROMPT*  
*QUOTE*  
*RANDOM*  
*RD*  
*READ*  
*RECORD*  
*RECORDING*  
*RECORDS*  
*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*  
*REVERSE\_FUNC*

*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*  
*RIGHT*  
*ROLLBACK*  
*ROUNDED*  
*RUN*  
*SAME*  
*SCREEN*  
*SCREEN\_CONTROL*  
*SCROLL*  
*SD*  
*SEARCH*  
*SECTION*  
*SECURE*  
*SEGMENT\_LIMIT*  
*SELECT*  
*SEMI\_COLON*  
*SENTENCE*  
*SEPARATE*  
*SEQUENCE*  
*SEQUENTIAL*  
*SET*  
*SHARING*  
*SIGN*  
*SIGNED*  
*SIGNED\_INT*  
*SIGNED\_LONG*  
*SIGNED\_SHORT*  
*SIZE*  
*SIZE\_ERROR*  
*SORT*  
*SORT\_MERGE*  
*SOURCE*  
*SOURCE\_COMPUTER*  
*SPACE*  
*SPECIAL\_NAMES*  
*STANDARD*

*STANDARD\_1*  
*STANDARD\_2*  
*START*  
*STATUS*  
*STOP*  
*STRING*  
*SUBSTITUTE\_FUNC*  
*SUBSTITUTE\_CASE\_FUNC*  
*SUBTRACT*  
*SUM*  
*SUPPRESS*  
*SYMBOLIC*  
*SYNCHRONIZED*  
*TALLYING*  
*TAPE*  
*TERMINATE*  
*TEST*  
*THAN*  
*THEN*  
*THRU*  
*TIME*  
*TIMES*  
*TO*  
*TOK\_FALSE*  
*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*

*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*  
*WORD*  
*WORDS*  
*WORKING\_STORAGE*  
*WRITE*  
*YYYYDDD*  
*YYYYMMDD*  
*ZERO*  
*UNARY\_SIGN*  
*TOKEN\_EOF*  
*ACCEPT*  
*ACCESS*  
*ADD*  
*ADDRESS*  
*ADVANCING*  
*AFTER*  
*ALL*  
*ALLOCATE*

*ALPHABET*  
*ALPHABETIC*  
*ALPHABETIC\_LOWER*  
*ALPHABETIC\_UPPER*  
*ALPHANUMERIC*  
*ALPHANUMERIC\_EDITED*  
*ALSO*  
*ALTER*  
*ALTERNATE*  
*AND*  
*ANY*  
*ARE*  
*AREA*  
*ARGUMENT\_NUMBER*  
*ARGUMENT\_VALUE*  
*AS*  
*ASCENDING*  
*ASSIGN*  
*AT*  
*AUTO*  
*AUTOMATIC*  
*BACKGROUND\_COLOR*  
*BASED*  
*BEFORE*  
*BELL*  
*BINARY*  
*BINARY\_C\_LONG*  
*BINARY\_CHAR*  
*BINARY\_DOUBLE*  
*BINARY\_LONG*  
*BINARY\_SHORT*  
*BLANK*  
*BLANK\_LINE*  
*BLANK\_SCREEN*  
*BLINK*  
*BLOCK*  
*BOTTOM*  
*BY*

**BYTE\_LENGTH**  
**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**  
**COMMAND\_LINE**  
**COMMA\_DELIM**  
**COMMIT**  
**COMMON**  
**COMP**  
**COMPUTE**  
**COMP\_1**  
**COMP\_2**  
**COMP\_3**  
**COMP\_4**  
**COMP\_5**  
**COMP\_X**  
**CONCATENATE\_FUNC**  
**CONFIGURATION**  
**CONSTANT**  
**CONTAINS**  
**CONTENT**  
**CONTINUE**  
**CONTROL**  
**CONTROLS**  
**CONTROL\_FOOTING**



*CONTROL\_HEADING*  
*CONVERTING*  
*CORRESPONDING*  
*COUNT*  
*CRT*  
*CURRENCY*  
*CURRENT\_DATE\_FUNC*  
*CURSOR*  
*CYCLE*  
*DATA*  
*DATE*  
*DAY*  
*DAY\_OF\_WEEK*  
*DE*  
*DEBUGGING*  
*DECIMAL\_POINT*  
*DECLARATIVES*  
*DEFAULT*  
*DELETE*  
*DELIMITED*  
*DELIMITER*  
*DEPENDING*  
*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*

*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*

*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*BACKGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*  
*GROUP*  
*HEADING*  
*HIGHLIGHT*  
*HIGH\_VALUE*  
*IDENTIFICATION*  
*IF*  
*IGNORE*  
*IGNORING*  
*IN*  
*INDEX*  
*INDEXED*  
*INDICATE*  
*INITIALIZE*  
*INITIALIZED*  
*INITIATE*  
*INPUT*

*INPUT\_OUTPUT*  
*INSPECT*  
*INTO*  
*INTRINSIC*  
*INVALID*  
*INVALID\_KEY*  
*IS*  
*I\_O*  
*I\_O\_CONTROL*  
*JUSTIFIED*  
*KEY*  
*LABEL*  
*LAST*  
*LAST\_DETAIL*  
*LE*  
*LEADING*  
*LEFT*  
*LENGTH*  
*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*

*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*  
*OR*  
*ORDER*  
*ORGANIZATION*

**OTHER**  
**OUTPUT**  
**OVERFLOW**  
**OVERLINE**  
**PACKED\_DECIMAL**  
**PADDING**  
**PAGE**  
**PAGE\_FOOTING**  
**PAGE\_HEADING**  
**PARAGRAPH**  
**PERFORM**  
**PICTURE**  
**PLUS**  
**POINTER**  
**POSITION**  
**POSITIVE**  
**PRESENT**  
**PREVIOUS**  
**PRINTER**  
**PRINTING**  
**PROCEDURE**  
**PROCEDURES**  
**PROCEED**  
**PROGRAM**  
**PROGRAM\_ID**  
**PROGRAM\_NAME**  
**PROGRAM\_POINTER**  
**PROMPT**  
**QUOTE**  
**RANDOM**  
**RD**  
**READ**  
**RECORD**  
**RECORDING**  
**RECORDS**  
**RECURSIVE**  
**REDEFINES**  
**REEL**

**REFERENCE**  
**RELATIVE**  
**RELEASE**  
**REMAINDER**  
**REMOVAL**  
**RENAMES**  
**REPLACING**  
**REPORT**  
**REPORTING**  
**REPORTS**  
**REPORT\_FOOTING**  
**REPORT\_HEADING**  
**REPOSITORY**  
**REQUIRED**  
**RESERVE**  
**RETURN**  
**RETURNING**  
**REVERSE\_FUNC**  
**REVERSE\_VIDEO**  
**REWIND**  
**REWRITE**  
**RIGHT**  
**ROLLBACK**  
**ROUNDED**  
**RUN**  
**SAME**  
**SCREEN**  
**SCREEN\_CONTROL**  
**SCROLL**  
**SD**  
**SEARCH**  
**SECTION**  
**SECURE**  
**SEGMENT\_LIMIT**  
**SELECT**  
**SEMI\_COLON**  
**SENTENCE**  
**SEPARATE**

**SEQUENCE**  
**SEQUENTIAL**  
**SET**  
**SHARING**  
**SIGN**  
**SIGNED**  
**SIGNED\_INT**  
**SIGNED\_LONG**  
**SIGNED\_SHORT**  
**SIZE**  
**SIZE\_ERROR**  
**SORT**  
**SORT\_MERGE**  
**SOURCE**  
**SOURCE\_COMPUTER**  
**SPACE**  
**SPECIAL\_NAMES**  
**STANDARD**  
**STANDARD\_1**  
**STANDARD\_2**  
**START**  
**STATUS**  
**STOP**  
**STRING**  
**SUBSTITUTE\_FUNC**  
**SUBSTITUTE\_CASE\_FUNC**  
**SUBTRACT**  
**SUM**  
**SUPPRESS**  
**SYMBOLIC**  
**SYNCHRONIZED**  
**TALLYING**  
**TAPE**  
**TERMINATE**  
**TEST**  
**THAN**  
**THEN**  
**THRU**



*TIME*  
*TIMES*  
*TO*  
*TOK\_FALSE*  
*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*

**WITH**  
**WORD**  
**WORDS**  
**WORKING\_STORAGE**  
**WRITE**  
**YYYYDDD**  
**YYYYMMDD**  
**ZERO**  
**UNARY\_SIGN**  
**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**  
**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**

Definition at line 64 of file ppparse.c.

```
    {  
    TOKEN_EOF = 0,  
    COPY = 258,  
    REPLACE = 259,  
    SUPPRESS = 260,  
    PRINTING = 261,  
    REPLACING = 262,  
    OFF = 263,  
    IN = 264,  
    OF = 265,  
    BY = 266,  
    EQEQ = 267,  
    TOKEN = 268  
    };  
};
```

### 7.11.4 Function Documentation

#### 7.11.4.1 int yyparse ( void )

### 7.11.5 Variable Documentation

#### 7.11.5.1 int yychar

Definition at line 782 of file ppparse.c.

#### 7.11.5.2 int yydebug

Definition at line 612 of file ppparse.c.

#### 7.11.5.3 YYSTYPE yylval

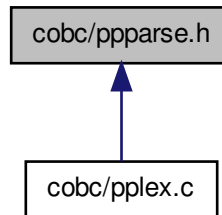
Definition at line 785 of file ppparse.c.

#### 7.11.5.4 int yynerrs

Definition at line 788 of file ppparse.c.

## 7.12 cobc/ppparse.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- union [YYSTYPE](#)

### Defines

- #define [TOKEN\\_EOF](#) 0
- #define [COPY](#) 258
- #define [REPLACE](#) 259
- #define [SUPPRESS](#) 260
- #define [PRINTING](#) 261
- #define [REPLACING](#) 262
- #define [OFF](#) 263
- #define [IN](#) 264
- #define [OF](#) 265
- #define [BY](#) 266
- #define [EQEQ](#) 267
- #define [TOKEN](#) 268
- #define [yystate YYSTYPE](#)
- #define [YYSTYPE\\_IS\\_DECLARED](#) 1
- #define [YYSTYPE\\_IS\\_TRIVIAL](#) 1

### Typedefs

- typedef union [YYSTYPE YYSTYPE](#)

## Enumerations

- enum `yytokentype` {
  - `TOKEN_EOF` = 0, `ACCEPT` = 258, `ACCESS` = 259, `ADD` = 260,
  - `ADDRESS` = 261, `ADVANCING` = 262, `AFTER` = 263, `ALL` = 264,
  - `ALLOCATE` = 265, `ALPHABET` = 266, `ALPHABETIC` = 267, `ALPHABETIC_-LOWER` = 268,
  - `ALPHABETIC_UPPER` = 269, `ALPHANUMERIC` = 270, `ALPHANUMERIC_EDITED` = 271, `ALSO` = 272,
  - `ALTER` = 273, `ALTERNATE` = 274, `AND` = 275, `ANY` = 276,
  - `ARE` = 277, `AREA` = 278, `ARGUMENT_NUMBER` = 279, `ARGUMENT_VALUE` = 280,
  - `AS` = 281, `ASCENDING` = 282, `ASSIGN` = 283, `AT` = 284,
  - `AUTO` = 285, `AUTOMATIC` = 286, `BACKGROUND_COLOR` = 287, `BASED` = 288,
  - `BEFORE` = 289, `BELL` = 290, `BINARY` = 291, `BINARY_C_LONG` = 292,
  - `BINARY_CHAR` = 293, `BINARY_DOUBLE` = 294, `BINARY_LONG` = 295, `BINARY_-SHORT` = 296,
  - `BLANK` = 297, `BLANK_LINE` = 298, `BLANK_SCREEN` = 299, `BLINK` = 300,
  - `BLOCK` = 301, `BOTTOM` = 302, `BY` = 303, `BYTE_LENGTH` = 304,
  - `CALL` = 305, `CANCEL` = 306, `CH` = 307, `CHAINING` = 308,
  - `CHARACTER` = 309, `CHARACTERS` = 310, `CLASS` = 311, `CLOSE` = 312,
  - `CODE` = 313, `CODE_SET` = 314, `COLLATING` = 315, `COL` = 316,
  - `COLS` = 317, `COLUMN` = 318, `COLUMNS` = 319, `COMMA` = 320,
  - `COMMAND_LINE` = 321, `COMMA_DELIM` = 322, `COMMIT` = 323, `COMMON` = 324,
  - `COMP` = 325, `COMPUTE` = 326, `COMP_1` = 327, `COMP_2` = 328,
  - `COMP_3` = 329, `COMP_4` = 330, `COMP_5` = 331, `COMP_X` = 332,
  - `CONCATENATE_FUNC` = 333, `CONFIGURATION` = 334, `CONSTANT` = 335,
  - `CONTAINS` = 336,
  - `CONTENT` = 337, `CONTINUE` = 338, `CONTROL` = 339, `CONTROLS` = 340,
  - `CONTROL_FOOTING` = 341, `CONTROL_HEADING` = 342, `CONVERTING` = 343, `CORRESPONDING` = 344,
  - `COUNT` = 345, `CRT` = 346, `CURRENCY` = 347, `CURRENT_DATE_FUNC` = 348,
  - `CURSOR` = 349, `CYCLE` = 350, `DATA` = 351, `DATE` = 352,
  - `DAY` = 353, `DAY_OF_WEEK` = 354, `DE` = 355, `DEBUGGING` = 356,
  - `DECIMAL_POINT` = 357, `DECLARATIVES` = 358, `DEFAULT` = 359, `DELETE` = 360,
  - `DELIMITED` = 361, `DELIMITER` = 362, `DEPENDING` = 363, `DESCENDING` = 364,
  - `DETAIL` = 365, `DISK` = 366, `DISPLAY` = 367, `DIVIDE` = 368,

`DIVISION` = 369, `DOWN` = 370, `DUPLICATES` = 371, `DYNAMIC` = 372,  
`EBCDIC` = 373, `ELSE` = 374, `END` = 375, `END_ACCEPT` = 376,  
`END_ADD` = 377, `END_CALL` = 378, `END_COMPUTE` = 379, `END_DELETE` =  
380,  
`END_DISPLAY` = 381, `END_DIVIDE` = 382, `END_EVALUATE` = 383, `END_-`  
`FUNCTION` = 384,  
`END_IF` = 385, `END_MULTIPLY` = 386, `END_PERFORM` = 387, `END_PROGRAM`  
= 388,  
`END_READ` = 389, `END_RETURN` = 390, `END_REWRITE` = 391, `END_SEARCH`  
= 392,  
`END_START` = 393, `END_STRING` = 394, `END_SUBTRACT` = 395, `END_UNSTRING`  
= 396,  
`END_WRITE` = 397, `ENTRY` = 398, `ENVIRONMENT` = 399, `ENVIRONMENT_-`  
`NAME` = 400,  
`ENVIRONMENT_VALUE` = 401, `EOL` = 402, `EOP` = 403, `EOS` = 404,  
`EQUAL` = 405, `EQUALS` = 406, `ERASE` = 407, `ERROR` = 408,  
`ESCAPE` = 409, `EVALUATE` = 410, `EVENT_STATUS` = 411, `EXCEPTION` = 412,  
`EXCLUSIVE` = 413, `EXIT` = 414, `EXTEND` = 415, `EXTERNAL` = 416,  
`FD` = 417, `FILE_CONTROL` = 418, `FILE_ID` = 419, `FILLER` = 420,  
`FINAL` = 421, `FIRST` = 422, `FOOTING` = 423, `FOR` = 424,  
`BACKGROUND_COLOR` = 425, `FOREVER` = 426, `FREE` = 427, `FROM` = 428,  
`FULL` = 429, `FUNCTION` = 430, `FUNCTION_ID` = 431, `FUNCTION_NAME` = 432,  
`GE` = 433, `GENERATE` = 434, `GIVING` = 435, `GLOBAL` = 436,  
`GO` = 437, `GOBACK` = 438, `GREATER` = 439, `GROUP` = 440,  
`HEADING` = 441, `HIGHLIGHT` = 442, `HIGH_VALUE` = 443, `IDENTIFICATION` =  
444,  
`IF` = 445, `IGNORE` = 446, `IGNORING` = 447, `IN` = 448,  
`INDEX` = 449, `INDEXED` = 450, `INDICATE` = 451, `INITIALIZE` = 452,  
`INITIALIZED` = 453, `INITIATE` = 454, `INPUT` = 455, `INPUT_OUTPUT` = 456,  
`INSPECT` = 457, `INTO` = 458, `INTRINSIC` = 459, `INVALID` = 460,  
`INVALID_KEY` = 461, `IS` = 462, `I_O` = 463, `I_O_CONTROL` = 464,  
`JUSTIFIED` = 465, `KEY` = 466, `LABEL` = 467, `LAST` = 468,  
`LAST_DETAIL` = 469, `LE` = 470, `LEADING` = 471, `LEFT` = 472,  
`LENGTH` = 473, `LESS` = 474, `LIMIT` = 475, `LIMITS` = 476,  
`LINAGE` = 477, `LINAGE_COUNTER` = 478, `LINE` = 479, `LINES` = 480,  
`LINKAGE` = 481, `LITERAL` = 482, `LOCALE` = 483, `LOCALE_DT_FUNC` = 484,  
`LOCAL_STORAGE` = 485, `LOCK` = 486, `LOWER_CASE_FUNC` = 487, `LOW-`  
`LIGHT` = 488,  
`LOW_VALUE` = 489, `MANUAL` = 490, `MEMORY` = 491, `MERGE` = 492,  
`MINUS` = 493, `MNEMONIC_NAME` = 494, `MODE` = 495, `MOVE` = 496,

`MULTIPLE` = 497, `MULTIPLY` = 498, `NATIONAL` = 499, `NATIONAL_EDITED` = 500,  
`NATIVE` = 501, `NE` = 502, `NEGATIVE` = 503, `NEXT` = 504,  
`NEXT_SENTENCE` = 505, `NO` = 506, `NOT` = 507, `NOT_END` = 508,  
`NOT_EOP` = 509, `NOT_EXCEPTION` = 510, `NOT_INVALID_KEY` = 511, `NOT_OVERFLOW` = 512,  
`NOT_SIZE_ERROR` = 513, `NO_ADVANCING` = 514, `NUMBER` = 515, `NUMBERS` = 516,  
`NUMERIC` = 517, `NUMERIC_EDITED` = 518, `NUMVALC_FUNC` = 519, `OBJECT_COMPUTER` = 520,  
`OCCURS` = 521, `OF` = 522, `OFF` = 523, `OMITTED` = 524,  
`ON` = 525, `ONLY` = 526, `OPEN` = 527, `OPTIONAL` = 528,  
`OR` = 529, `ORDER` = 530, `ORGANIZATION` = 531, `OTHER` = 532,  
`OUTPUT` = 533, `OVERFLOW` = 534, `OVERLINE` = 535, `PACKED_DECIMAL` = 536,  
`PADDING` = 537, `PAGE` = 538, `PAGE_FOOTING` = 539, `PAGE_HEADING` = 540,  
`PARAGRAPH` = 541, `PERFORM` = 542, `PICTURE` = 543, `PLUS` = 544,  
`POINTER` = 545, `POSITION` = 546, `POSITIVE` = 547, `PRESENT` = 548,  
`PREVIOUS` = 549, `PRINTER` = 550, `PRINTING` = 551, `PROCEDURE` = 552,  
`PROCEDURES` = 553, `PROCEED` = 554, `PROGRAM` = 555, `PROGRAM_ID` = 556,  
`PROGRAM_NAME` = 557, `PROGRAM_POINTER` = 558, `PROMPT` = 559, `QUOTE` = 560,  
`RANDOM` = 561, `RD` = 562, `READ` = 563, `RECORD` = 564,  
`RECORDING` = 565, `RECORDS` = 566, `RECURSIVE` = 567, `REDEFINES` = 568,  
`REEL` = 569, `REFERENCE` = 570, `RELATIVE` = 571, `RELEASE` = 572,  
`REMAINDER` = 573, `REMOVAL` = 574, `RENAMES` = 575, `REPLACING` = 576,  
`REPORT` = 577, `REPORTING` = 578, `REPORTS` = 579, `REPORT_FOOTING` = 580,  
`REPORT_HEADING` = 581, `REPOSITORY` = 582, `REQUIRED` = 583, `RESERVE` = 584,  
`RETURN` = 585, `RETURNING` = 586, `REVERSE_FUNC` = 587, `REVERSE_VIDEO` = 588,  
`REWIND` = 589, `REWRITE` = 590, `RIGHT` = 591, `ROLLBACK` = 592,  
`ROUNDED` = 593, `RUN` = 594, `SAME` = 595, `SCREEN` = 596,  
`SCREEN_CONTROL` = 597, `SCROLL` = 598, `SD` = 599, `SEARCH` = 600,  
`SECTION` = 601, `SECURE` = 602, `SEGMENT_LIMIT` = 603, `SELECT` = 604,  
`SEMI_COLON` = 605, `SENTENCE` = 606, `SEPARATE` = 607, `SEQUENCE` = 608,  
`SEQUENTIAL` = 609, `SET` = 610, `SHARING` = 611, `SIGN` = 612,  
`SIGNED` = 613, `SIGNED_INT` = 614, `SIGNED_LONG` = 615, `SIGNED_SHORT` = 616,

SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,  
UPDATE = 669, UPON = 670, UPON\_ARGUMENT\_NUMBER = 671, UPON\_  
COMMAND\_LINE = 672,  
UPON\_ENVIRONMENT\_NAME = 673, UPON\_ENVIRONMENT\_VALUE = 674,  
UPPER\_CASE\_FUNC = 675, USAGE = 676,  
USE = 677, USING = 678, VALUE = 679, VARYING = 680,  
WAIT = 681, WHEN = 682, WHEN\_COMPILED\_FUNC = 683, WITH = 684,  
WORD = 685, WORDS = 686, WORKING\_STORAGE = 687, WRITE = 688,  
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY\_SIGN = 692,  
TOKEN\_EOF = 0, ACCEPT = 258, ACCESS = 259, ADD = 260,  
ADDRESS = 261, ADVANCING = 262, AFTER = 263, ALL = 264,  
ALLOCATE = 265, ALPHABET = 266, ALPHABETIC = 267, ALPHABETIC\_  
LOWER = 268,  
ALPHABETIC\_UPPER = 269, ALPHANUMERIC = 270, ALPHANUMERIC\_EDITED  
= 271, ALSO = 272,  
ALTER = 273, ALTERNATE = 274, AND = 275, ANY = 276,  
ARE = 277, AREA = 278, ARGUMENT\_NUMBER = 279, ARGUMENT\_VALUE  
= 280,  
AS = 281, ASCENDING = 282, ASSIGN = 283, AT = 284,  
AUTO = 285, AUTOMATIC = 286, BACKGROUND\_COLOR = 287, BASED = 288,  
BEFORE = 289, BELL = 290, BINARY = 291, BINARY\_C\_LONG = 292,  
BINARY\_CHAR = 293, BINARY\_DOUBLE = 294, BINARY\_LONG = 295, BINARY\_  
SHORT = 296,  
BLANK = 297, BLANK\_LINE = 298, BLANK\_SCREEN = 299, BLINK = 300,  
BLOCK = 301, BOTTOM = 302, BY = 303, BYTE\_LENGTH = 304,



CALL = 305, CANCEL = 306, CH = 307, CHAINING = 308,  
CHARACTER = 309, CHARACTERS = 310, CLASS = 311, CLOSE = 312,  
CODE = 313, CODE\_SET = 314, COLLATING = 315, COL = 316,  
COLS = 317, COLUMN = 318, COLUMNS = 319, COMMA = 320,  
COMMAND\_LINE = 321, COMMA\_DELIM = 322, COMMIT = 323, COMMON =  
324,  
COMP = 325, COMPUTE = 326, COMP\_1 = 327, COMP\_2 = 328,  
COMP\_3 = 329, COMP\_4 = 330, COMP\_5 = 331, COMP\_X = 332,  
CONCATENATE\_FUNC = 333, CONFIGURATION = 334, CONSTANT = 335,  
CONTAINS = 336,  
CONTENT = 337, CONTINUE = 338, CONTROL = 339, CONTROLS = 340,  
CONTROL\_FOOTING = 341, CONTROL\_HEADING = 342, CONVERTING =  
343, CORRESPONDING = 344,  
COUNT = 345, CRT = 346, CURRENCY = 347, CURRENT\_DATE\_FUNC = 348,  
CURSOR = 349, CYCLE = 350, DATA = 351, DATE = 352,  
DAY = 353, DAY\_OF\_WEEK = 354, DE = 355, DEBUGGING = 356,  
DECIMAL\_POINT = 357, DECLARATIVES = 358, DEFAULT = 359, DELETE =  
360,  
DELIMITED = 361, DELIMITER = 362, DEPENDING = 363, DESCENDING =  
364,  
DETAIL = 365, DISK = 366, DISPLAY = 367, DIVIDE = 368,  
DIVISION = 369, DOWN = 370, DUPLICATES = 371, DYNAMIC = 372,  
EBCDIC = 373, ELSE = 374, END = 375, END\_ACCEPT = 376,  
END\_ADD = 377, END\_CALL = 378, END\_COMPUTE = 379, END\_DELETE =  
380,  
END\_DISPLAY = 381, END\_DIVIDE = 382, END\_EVALUATE = 383, END\_  
FUNCTION = 384,  
END\_IF = 385, END\_MULTIPLY = 386, END\_PERFORM = 387, END\_PROGRAM  
= 388,  
END\_READ = 389, END\_RETURN = 390, END\_REWRITE = 391, END\_SEARCH  
= 392,  
END\_START = 393, END\_STRING = 394, END\_SUBTRACT = 395, END\_UNSTRING  
= 396,  
END\_WRITE = 397, ENTRY = 398, ENVIRONMENT = 399, ENVIRONMENT\_  
NAME = 400,  
ENVIRONMENT\_VALUE = 401, EOL = 402, EOP = 403, EOS = 404,  
EQUAL = 405, EQUALS = 406, ERASE = 407, ERROR = 408,  
ESCAPE = 409, EVALUATE = 410, EVENT\_STATUS = 411, EXCEPTION = 412,  
EXCLUSIVE = 413, EXIT = 414, EXTEND = 415, EXTERNAL = 416,  
FD = 417, FILE\_CONTROL = 418, FILE\_ID = 419, FILLER = 420,

FINAL = 421, FIRST = 422, FOOTING = 423, FOR = 424,  
BACKGROUND\_COLOR = 425, FOREVER = 426, FREE = 427, FROM = 428,  
FULL = 429, FUNCTION = 430, FUNCTION\_ID = 431, FUNCTION\_NAME = 432,  
GE = 433, GENERATE = 434, GIVING = 435, GLOBAL = 436,  
GO = 437, GOBACK = 438, GREATER = 439, GROUP = 440,  
HEADING = 441, HIGHLIGHT = 442, HIGH\_VALUE = 443, IDENTIFICATION =  
444,  
IF = 445, IGNORE = 446, IGNORING = 447, IN = 448,  
INDEX = 449, INDEXED = 450, INDICATE = 451, INITIALIZE = 452,  
INITIALIZED = 453, INITIATE = 454, INPUT = 455, INPUT\_OUTPUT = 456,  
INSPECT = 457, INTO = 458, INTRINSIC = 459, INVALID = 460,  
INVALID\_KEY = 461, IS = 462, I\_O = 463, I\_O\_CONTROL = 464,  
JUSTIFIED = 465, KEY = 466, LABEL = 467, LAST = 468,  
LAST\_DETAIL = 469, LE = 470, LEADING = 471, LEFT = 472,  
LENGTH = 473, LESS = 474, LIMIT = 475, LIMITS = 476,  
LINAGE = 477, LINAGE\_COUNTER = 478, LINE = 479, LINES = 480,  
LINKAGE = 481, LITERAL = 482, LOCALE = 483, LOCALE\_DT\_FUNC = 484,  
LOCAL\_STORAGE = 485, LOCK = 486, LOWER\_CASE\_FUNC = 487, LOW-  
LIGHT = 488,  
LOW\_VALUE = 489, MANUAL = 490, MEMORY = 491, MERGE = 492,  
MINUS = 493, MNEMONIC\_NAME = 494, MODE = 495, MOVE = 496,  
MULTIPLE = 497, MULTIPLY = 498, NATIONAL = 499, NATIONAL\_EDITED =  
500,  
NATIVE = 501, NE = 502, NEGATIVE = 503, NEXT = 504,  
NEXT\_SENTENCE = 505, NO = 506, NOT = 507, NOT\_END = 508,  
NOT\_EOP = 509, NOT\_EXCEPTION = 510, NOT\_INVALID\_KEY = 511, NOT\_-  
OVERFLOW = 512,  
NOT\_SIZE\_ERROR = 513, NO\_ADVANCING = 514, NUMBER = 515, NUM-  
BERS = 516,  
NUMERIC = 517, NUMERIC\_EDITED = 518, NUMVALC\_FUNC = 519, OBJECT\_-  
COMPUTER = 520,  
OCCURS = 521, OF = 522, OFF = 523, OMITTED = 524,  
ON = 525, ONLY = 526, OPEN = 527, OPTIONAL = 528,  
OR = 529, ORDER = 530, ORGANIZATION = 531, OTHER = 532,  
OUTPUT = 533, OVERFLOW = 534, OVERLINE = 535, PACKED\_DECIMAL =  
536,  
PADDING = 537, PAGE = 538, PAGE\_FOOTING = 539, PAGE\_HEADING = 540,  
PARAGRAPH = 541, PERFORM = 542, PICTURE = 543, PLUS = 544,  
POINTER = 545, POSITION = 546, POSITIVE = 547, PRESENT = 548,

PREVIOUS = 549, PRINTER = 550, PRINTING = 551, PROCEDURE = 552,  
PROCEDURES = 553, PROCEED = 554, PROGRAM = 555, PROGRAM\_ID =  
556,  
PROGRAM\_NAME = 557, PROGRAM\_POINTER = 558, PROMPT = 559, QUOTE  
= 560,  
RANDOM = 561, RD = 562, READ = 563, RECORD = 564,  
RECORDING = 565, RECORDS = 566, RECURSIVE = 567, REDEFINES = 568,  
REEL = 569, REFERENCE = 570, RELATIVE = 571, RELEASE = 572,  
REMAINDER = 573, REMOVAL = 574, RENAMES = 575, REPLACING = 576,  
REPORT = 577, REPORTING = 578, REPORTS = 579, REPORT\_FOOTING =  
580,  
REPORT\_HEADING = 581, REPOSITORY = 582, REQUIRED = 583, RESERVE  
= 584,  
RETURN = 585, RETURNING = 586, REVERSE\_FUNC = 587, REVERSE\_  
VIDEO = 588,  
REWIND = 589, REWRITE = 590, RIGHT = 591, ROLLBACK = 592,  
ROUNDED = 593, RUN = 594, SAME = 595, SCREEN = 596,  
SCREEN\_CONTROL = 597, SCROLL = 598, SD = 599, SEARCH = 600,  
SECTION = 601, SECURE = 602, SEGMENT\_LIMIT = 603, SELECT = 604,  
SEMI\_COLON = 605, SENTENCE = 606, SEPARATE = 607, SEQUENCE = 608,  
SEQUENTIAL = 609, SET = 610, SHARING = 611, SIGN = 612,  
SIGNED = 613, SIGNED\_INT = 614, SIGNED\_LONG = 615, SIGNED\_SHORT  
= 616,  
SIZE = 617, SIZE\_ERROR = 618, SORT = 619, SORT\_MERGE = 620,  
SOURCE = 621, SOURCE\_COMPUTER = 622, SPACE = 623, SPECIAL\_NAMES  
= 624,  
STANDARD = 625, STANDARD\_1 = 626, STANDARD\_2 = 627, START = 628,  
STATUS = 629, STOP = 630, STRING = 631, SUBSTITUTE\_FUNC = 632,  
SUBSTITUTE\_CASE\_FUNC = 633, SUBTRACT = 634, SUM = 635, SUPPRESS  
= 636,  
SYMBOLIC = 637, SYNCHRONIZED = 638, TALLYING = 639, TAPE = 640,  
TERMINATE = 641, TEST = 642, THAN = 643, THEN = 644,  
THRU = 645, TIME = 646, TIMES = 647, TO = 648,  
TOK\_FALSE = 649, TOK\_FILE = 650, TOK\_INITIAL = 651, TOK\_NULL = 652,  
TOK\_TRUE = 653, TOP = 654, TRAILING = 655, TRANSFORM = 656,  
TRIM\_FUNCTION = 657, TYPE = 658, UNDERLINE = 659, UNIT = 660,  
UNLOCK = 661, UNSIGNED = 662, UNSIGNED\_INT = 663, UNSIGNED\_LONG  
= 664,  
UNSIGNED\_SHORT = 665, UNSTRING = 666, UNTIL = 667, UP = 668,

```

UPDATE = 669, UPON = 670, UPON_ARGUMENT_NUMBER = 671, UPON_-
COMMAND_LINE = 672,
UPON_ENVIRONMENT_NAME = 673, UPON_ENVIRONMENT_VALUE = 674,
UPPER_CASE_FUNC = 675, USAGE = 676,
USE = 677, USING = 678, VALUE = 679, VARYING = 680,
WAIT = 681, WHEN = 682, WHEN_COMPILED_FUNC = 683, WITH = 684,
WORD = 685, WORDS = 686, WORKING_STORAGE = 687, WRITE = 688,
YYYYDDD = 689, YYYYMMDD = 690, ZERO = 691, UNARY_SIGN = 692,
TOKEN_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,
PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,
OF = 265, BY = 266, EQEQ = 267, TOKEN = 268,
TOKEN_EOF = 0, COPY = 258, REPLACE = 259, SUPPRESS = 260,
PRINTING = 261, REPLACING = 262, OFF = 263, IN = 264,
OF = 265, BY = 266, EQEQ = 267, TOKEN = 268 }

```

## Variables

- [YYSTYPE pplval](#)

### 7.12.1 Define Documentation

#### 7.12.1.1 #define BY 266

Definition at line 55 of file ppparse.h.

#### 7.12.1.2 #define COPY 258

Definition at line 47 of file ppparse.h.

#### 7.12.1.3 #define EQEQ 267

Definition at line 56 of file ppparse.h.

#### 7.12.1.4 #define IN 264

Definition at line 53 of file ppparse.h.

#### 7.12.1.5 #define OF 265

Definition at line 54 of file ppparse.h.

**7.12.1.6 #define OFF 263**

Definition at line 52 of file ppparse.h.

**7.12.1.7 #define PRINTING 261**

Definition at line 50 of file ppparse.h.

**7.12.1.8 #define REPLACE 259**

Definition at line 48 of file ppparse.h.

**7.12.1.9 #define REPLACING 262**

Definition at line 51 of file ppparse.h.

**7.12.1.10 #define SUPPRESS 260**

Definition at line 49 of file ppparse.h.

**7.12.1.11 #define TOKEN 268**

Definition at line 57 of file ppparse.h.

**7.12.1.12 #define TOKEN\_EOF 0**

Definition at line 46 of file ppparse.h.

**7.12.1.13 #define yystype YYSTYPE**

Definition at line 71 of file ppparse.h.

**7.12.1.14 #define YYSTYPE\_IS\_DECLARED 1**

Definition at line 72 of file ppparse.h.

**7.12.1.15 #define YYSTYPE\_IS\_TRIVIAL 1**

Definition at line 73 of file ppparse.h.

## 7.12.2 Typedef Documentation

### 7.12.2.1 typedef union YYSTYPE YYSTYPE

## 7.12.3 Enumeration Type Documentation

### 7.12.3.1 enum yytokentype

#### Enumerator:

***TOKEN\_EOF***  
***ACCEPT***  
***ACCESS***  
***ADD***  
***ADDRESS***  
***ADVANCING***  
***AFTER***  
***ALL***  
***ALLOCATE***  
***ALPHABET***  
***ALPHABETIC***  
***ALPHABETIC\_LOWER***  
***ALPHABETIC\_UPPER***  
***ALPHANUMERIC***  
***ALPHANUMERIC\_EDITED***  
***ALSO***  
***ALTER***  
***ALTERNATE***  
***AND***  
***ANY***  
***ARE***  
***AREA***  
***ARGUMENT\_NUMBER***  
***ARGUMENT\_VALUE***  
***AS***  
***ASCENDING***  
***ASSIGN***  
***AT***  
***AUTO***  
***AUTOMATIC***  
***BACKGROUND\_COLOR***

**BASED**  
**BEFORE**  
**BELL**  
**BINARY**  
**BINARY\_C\_LONG**  
**BINARY\_CHAR**  
**BINARY\_DOUBLE**  
**BINARY\_LONG**  
**BINARY\_SHORT**  
**BLANK**  
**BLANK\_LINE**  
**BLANK\_SCREEN**  
**BLINK**  
**BLOCK**  
**BOTTOM**  
**BY**  
**BYTE\_LENGTH**  
**CALL**  
**CANCEL**  
**CH**  
**CHAINING**  
**CHARACTER**  
**CHARACTERS**  
**CLASS**  
**CLOSE**  
**CODE**  
**CODE\_SET**  
**COLLATING**  
**COL**  
**COLS**  
**COLUMN**  
**COLUMNS**  
**COMMA**  
**COMMAND\_LINE**  
**COMMA\_DELIM**  
**COMMIT**  
**COMMON**  
**COMP**

**COMPUTE**  
**COMP\_1**  
**COMP\_2**  
**COMP\_3**  
**COMP\_4**  
**COMP\_5**  
**COMP\_X**  
**CONCATENATE\_FUNC**  
**CONFIGURATION**  
**CONSTANT**  
**CONTAINS**  
**CONTENT**  
**CONTINUE**  
**CONTROL**  
**CONTROLS**  
**CONTROL\_FOOTING**  
**CONTROL\_HEADING**  
**CONVERTING**  
**CORRESPONDING**  
**COUNT**  
**CRT**  
**CURRENCY**  
**CURRENT\_DATE\_FUNC**  
**CURSOR**  
**CYCLE**  
**DATA**  
**DATE**  
**DAY**  
**DAY\_OF\_WEEK**  
**DE**  
**DEBUGGING**  
**DECIMAL\_POINT**  
**DECLARATIVES**  
**DEFAULT**  
**DELETE**  
**DELIMITED**  
**DELIMITER**  
**DEPENDING**



*DESCENDING*  
*DETAIL*  
*DISK*  
*DISPLAY*  
*DIVIDE*  
*DIVISION*  
*DOWN*  
*DUPLICATES*  
*DYNAMIC*  
*EBCDIC*  
*ELSE*  
*END*  
*END\_ACCEPT*  
*END\_ADD*  
*END\_CALL*  
*END\_COMPUTE*  
*END\_DELETE*  
*END\_DISPLAY*  
*END\_DIVIDE*  
*END\_EVALUATE*  
*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*

*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*  
*FIRST*  
*FOOTING*  
*FOR*  
*BACKGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*

**GROUP**  
**HEADING**  
**HIGHLIGHT**  
**HIGH\_VALUE**  
**IDENTIFICATION**  
**IF**  
**IGNORE**  
**IGNORING**  
**IN**  
**INDEX**  
**INDEXED**  
**INDICATE**  
**INITIALIZE**  
**INITIALIZED**  
**INITIATE**  
**INPUT**  
**INPUT\_OUTPUT**  
**INSPECT**  
**INTO**  
**INTRINSIC**  
**INVALID**  
**INVALID\_KEY**  
**IS**  
**I\_O**  
**I\_O\_CONTROL**  
**JUSTIFIED**  
**KEY**  
**LABEL**  
**LAST**  
**LAST\_DETAIL**  
**LE**  
**LEADING**  
**LEFT**  
**LENGTH**  
**LESS**  
**LIMIT**  
**LIMITS**  
**LINAGE**

*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*  
*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*

**NUMBERS**  
**NUMERIC**  
**NUMERIC\_EDITED**  
**NUMVALC\_FUNC**  
**OBJECT\_COMPUTER**  
**OCCURS**  
**OF**  
**OFF**  
**OMITTED**  
**ON**  
**ONLY**  
**OPEN**  
**OPTIONAL**  
**OR**  
**ORDER**  
**ORGANIZATION**  
**OTHER**  
**OUTPUT**  
**OVERFLOW**  
**OVERLINE**  
**PACKED\_DECIMAL**  
**PADDING**  
**PAGE**  
**PAGE\_FOOTING**  
**PAGE\_HEADING**  
**PARAGRAPH**  
**PERFORM**  
**PICTURE**  
**PLUS**  
**POINTER**  
**POSITION**  
**POSITIVE**  
**PRESENT**  
**PREVIOUS**  
**PRINTER**  
**PRINTING**  
**PROCEDURE**  
**PROCEDURES**

*PROCEED*  
*PROGRAM*  
*PROGRAM\_ID*  
*PROGRAM\_NAME*  
*PROGRAM\_POINTER*  
*PROMPT*  
*QUOTE*  
*RANDOM*  
*RD*  
*READ*  
*RECORD*  
*RECORDING*  
*RECORDS*  
*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*  
*REMOVAL*  
*RENAMES*  
*REPLACING*  
*REPORT*  
*REPORTING*  
*REPORTS*  
*REPORT\_FOOTING*  
*REPORT\_HEADING*  
*REPOSITORY*  
*REQUIRED*  
*RESERVE*  
*RETURN*  
*RETURNING*  
*REVERSE\_FUNC*  
*REVERSE\_VIDEO*  
*REWIND*  
*REWRITE*  
*RIGHT*

**ROLLBACK**  
**ROUNDED**  
**RUN**  
**SAME**  
**SCREEN**  
**SCREEN\_CONTROL**  
**SCROLL**  
**SD**  
**SEARCH**  
**SECTION**  
**SECURE**  
**SEGMENT\_LIMIT**  
**SELECT**  
**SEMI\_COLON**  
**SENTENCE**  
**SEPARATE**  
**SEQUENCE**  
**SEQUENTIAL**  
**SET**  
**SHARING**  
**SIGN**  
**SIGNED**  
**SIGNED\_INT**  
**SIGNED\_LONG**  
**SIGNED\_SHORT**  
**SIZE**  
**SIZE\_ERROR**  
**SORT**  
**SORT\_MERGE**  
**SOURCE**  
**SOURCE\_COMPUTER**  
**SPACE**  
**SPECIAL\_NAMES**  
**STANDARD**  
**STANDARD\_1**  
**STANDARD\_2**  
**START**  
**STATUS**

**STOP**  
**STRING**  
**SUBSTITUTE\_FUNC**  
**SUBSTITUTE\_CASE\_FUNC**  
**SUBTRACT**  
**SUM**  
**SUPPRESS**  
**SYMBOLIC**  
**SYNCHRONIZED**  
**TALLYING**  
**TAPE**  
**TERMINATE**  
**TEST**  
**THAN**  
**THEN**  
**THRU**  
**TIME**  
**TIMES**  
**TO**  
**TOK\_FALSE**  
**TOK\_FILE**  
**TOK\_INITIAL**  
**TOK\_NULL**  
**TOK\_TRUE**  
**TOP**  
**TRAILING**  
**TRANSFORM**  
**TRIM\_FUNCTION**  
**TYPE**  
**UNDERLINE**  
**UNIT**  
**UNLOCK**  
**UNSIGNED**  
**UNSIGNED\_INT**  
**UNSIGNED\_LONG**  
**UNSIGNED\_SHORT**  
**UNSTRING**  
**UNTIL**



*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*  
*WORD*  
*WORDS*  
*WORKING\_STORAGE*  
*WRITE*  
*YYYYDDD*  
*YYYYMMDD*  
*ZERO*  
*UNARY\_SIGN*  
*TOKEN\_EOF*  
*ACCEPT*  
*ACCESS*  
*ADD*  
*ADDRESS*  
*ADVANCING*  
*AFTER*  
*ALL*  
*ALLOCATE*  
*ALPHABET*  
*ALPHABETIC*  
*ALPHABETIC\_LOWER*  
*ALPHABETIC\_UPPER*

*ALPHANUMERIC*  
*ALPHANUMERIC\_EDITED*  
*ALSO*  
*ALTER*  
*ALTERNATE*  
*AND*  
*ANY*  
*ARE*  
*AREA*  
*ARGUMENT\_NUMBER*  
*ARGUMENT\_VALUE*  
*AS*  
*ASCENDING*  
*ASSIGN*  
*AT*  
*AUTO*  
*AUTOMATIC*  
*BACKGROUND\_COLOR*  
*BASED*  
*BEFORE*  
*BELL*  
*BINARY*  
*BINARY\_C\_LONG*  
*BINARY\_CHAR*  
*BINARY\_DOUBLE*  
*BINARY\_LONG*  
*BINARY\_SHORT*  
*BLANK*  
*BLANK\_LINE*  
*BLANK\_SCREEN*  
*BLINK*  
*BLOCK*  
*BOTTOM*  
*BY*  
*BYTE\_LENGTH*  
*CALL*  
*CANCEL*  
*CH*

*CHAINING*  
*CHARACTER*  
*CHARACTERS*  
*CLASS*  
*CLOSE*  
*CODE*  
*CODE\_SET*  
*COLLATING*  
*COL*  
*COLS*  
*COLUMN*  
*COLUMNS*  
*COMMA*  
*COMMAND\_LINE*  
*COMMA\_DELIM*  
*COMMIT*  
*COMMON*  
*COMP*  
*COMPUTE*  
*COMP\_1*  
*COMP\_2*  
*COMP\_3*  
*COMP\_4*  
*COMP\_5*  
*COMP\_X*  
*CONCATENATE\_FUNC*  
*CONFIGURATION*  
*CONSTANT*  
*CONTAINS*  
*CONTENT*  
*CONTINUE*  
*CONTROL*  
*CONTROLS*  
*CONTROL\_FOOTING*  
*CONTROL\_HEADING*  
*CONVERTING*  
*CORRESPONDING*  
*COUNT*

**CRT**  
**CURRENCY**  
**CURRENT\_DATE\_FUNC**  
**CURSOR**  
**CYCLE**  
**DATA**  
**DATE**  
**DAY**  
**DAY\_OF\_WEEK**  
**DE**  
**DEBUGGING**  
**DECIMAL\_POINT**  
**DECLARATIVES**  
**DEFAULT**  
**DELETE**  
**DELIMITED**  
**DELIMITER**  
**DEPENDING**  
**DESCENDING**  
**DETAIL**  
**DISK**  
**DISPLAY**  
**DIVIDE**  
**DIVISION**  
**DOWN**  
**DUPLICATES**  
**DYNAMIC**  
**EBCDIC**  
**ELSE**  
**END**  
**END\_ACCEPT**  
**END\_ADD**  
**END\_CALL**  
**END\_COMPUTE**  
**END\_DELETE**  
**END\_DISPLAY**  
**END\_DIVIDE**  
**END\_EVALUATE**

*END\_FUNCTION*  
*END\_IF*  
*END\_MULTIPLY*  
*END\_PERFORM*  
*END\_PROGRAM*  
*END\_READ*  
*END\_RETURN*  
*END\_REWRITE*  
*END\_SEARCH*  
*END\_START*  
*END\_STRING*  
*END\_SUBTRACT*  
*END\_UNSTRING*  
*END\_WRITE*  
*ENTRY*  
*ENVIRONMENT*  
*ENVIRONMENT\_NAME*  
*ENVIRONMENT\_VALUE*  
*EOL*  
*EOP*  
*EOS*  
*EQUAL*  
*EQUALS*  
*ERASE*  
*ERROR*  
*ESCAPE*  
*EVALUATE*  
*EVENT\_STATUS*  
*EXCEPTION*  
*EXCLUSIVE*  
*EXIT*  
*EXTEND*  
*EXTERNAL*  
*FD*  
*FILE\_CONTROL*  
*FILE\_ID*  
*FILLER*  
*FINAL*

*FIRST*  
*FOOTING*  
*FOR*  
*FOREGROUND\_COLOR*  
*FOREVER*  
*FREE*  
*FROM*  
*FULL*  
*FUNCTION*  
*FUNCTION\_ID*  
*FUNCTION\_NAME*  
*GE*  
*GENERATE*  
*GIVING*  
*GLOBAL*  
*GO*  
*GOBACK*  
*GREATER*  
*GROUP*  
*HEADING*  
*HIGHLIGHT*  
*HIGH\_VALUE*  
*IDENTIFICATION*  
*IF*  
*IGNORE*  
*IGNORING*  
*IN*  
*INDEX*  
*INDEXED*  
*INDICATE*  
*INITIALIZE*  
*INITIALIZED*  
*INITIATE*  
*INPUT*  
*INPUT\_OUTPUT*  
*INSPECT*  
*INTO*  
*INTRINSIC*

*INVALID*  
*INVALID\_KEY*  
*IS*  
*I\_O*  
*I\_O\_CONTROL*  
*JUSTIFIED*  
*KEY*  
*LABEL*  
*LAST*  
*LAST\_DETAIL*  
*LE*  
*LEADING*  
*LEFT*  
*LENGTH*  
*LESS*  
*LIMIT*  
*LIMITS*  
*LINAGE*  
*LINAGE\_COUNTER*  
*LINE*  
*LINES*  
*LINKAGE*  
*LITERAL*  
*LOCALE*  
*LOCALE\_DT\_FUNC*  
*LOCAL\_STORAGE*  
*LOCK*  
*LOWER\_CASE\_FUNC*  
*LOWLIGHT*  
*LOW\_VALUE*  
*MANUAL*  
*MEMORY*  
*MERGE*  
*MINUS*  
*MNEMONIC\_NAME*  
*MODE*  
*MOVE*  
*MULTIPLE*

*MULTIPLY*  
*NATIONAL*  
*NATIONAL\_EDITED*  
*NATIVE*  
*NE*  
*NEGATIVE*  
*NEXT*  
*NEXT\_SENTENCE*  
*NO*  
*NOT*  
*NOT\_END*  
*NOT\_EOP*  
*NOT\_EXCEPTION*  
*NOT\_INVALID\_KEY*  
*NOT\_OVERFLOW*  
*NOT\_SIZE\_ERROR*  
*NO\_ADVANCING*  
*NUMBER*  
*NUMBERS*  
*NUMERIC*  
*NUMERIC\_EDITED*  
*NUMVALC\_FUNC*  
*OBJECT\_COMPUTER*  
*OCCURS*  
*OF*  
*OFF*  
*OMITTED*  
*ON*  
*ONLY*  
*OPEN*  
*OPTIONAL*  
*OR*  
*ORDER*  
*ORGANIZATION*  
*OTHER*  
*OUTPUT*  
*OVERFLOW*  
*OVERLINE*



*PACKED\_DECIMAL*  
*PADDING*  
*PAGE*  
*PAGE\_FOOTING*  
*PAGE\_HEADING*  
*PARAGRAPH*  
*PERFORM*  
*PICTURE*  
*PLUS*  
*POINTER*  
*POSITION*  
*POSITIVE*  
*PRESENT*  
*PREVIOUS*  
*PRINTER*  
*PRINTING*  
*PROCEDURE*  
*PROCEDURES*  
*PROCEED*  
*PROGRAM*  
*PROGRAM\_ID*  
*PROGRAM\_NAME*  
*PROGRAM\_POINTER*  
*PROMPT*  
*QUOTE*  
*RANDOM*  
*RD*  
*READ*  
*RECORD*  
*RECORDING*  
*RECORDS*  
*RECURSIVE*  
*REDEFINES*  
*REEL*  
*REFERENCE*  
*RELATIVE*  
*RELEASE*  
*REMAINDER*

**REMOVAL**  
**RENAMES**  
**REPLACING**  
**REPORT**  
**REPORTING**  
**REPORTS**  
**REPORT\_FOOTING**  
**REPORT\_HEADING**  
**REPOSITORY**  
**REQUIRED**  
**RESERVE**  
**RETURN**  
**RETURNING**  
**REVERSE\_FUNC**  
**REVERSE\_VIDEO**  
**REWIND**  
**REWRITE**  
**RIGHT**  
**ROLLBACK**  
**ROUNDED**  
**RUN**  
**SAME**  
**SCREEN**  
**SCREEN\_CONTROL**  
**SCROLL**  
**SD**  
**SEARCH**  
**SECTION**  
**SECURE**  
**SEGMENT\_LIMIT**  
**SELECT**  
**SEMI\_COLON**  
**SENTENCE**  
**SEPARATE**  
**SEQUENCE**  
**SEQUENTIAL**  
**SET**  
**SHARING**

*SIGN*  
*SIGNED*  
*SIGNED\_INT*  
*SIGNED\_LONG*  
*SIGNED\_SHORT*  
*SIZE*  
*SIZE\_ERROR*  
*SORT*  
*SORT\_MERGE*  
*SOURCE*  
*SOURCE\_COMPUTER*  
*SPACE*  
*SPECIAL\_NAMES*  
*STANDARD*  
*STANDARD\_1*  
*STANDARD\_2*  
*START*  
*STATUS*  
*STOP*  
*STRING*  
*SUBSTITUTE\_FUNC*  
*SUBSTITUTE\_CASE\_FUNC*  
*SUBTRACT*  
*SUM*  
*SUPPRESS*  
*SYMBOLIC*  
*SYNCHRONIZED*  
*TALLYING*  
*TAPE*  
*TERMINATE*  
*TEST*  
*THAN*  
*THEN*  
*THRU*  
*TIME*  
*TIMES*  
*TO*  
*TOK\_FALSE*

*TOK\_FILE*  
*TOK\_INITIAL*  
*TOK\_NULL*  
*TOK\_TRUE*  
*TOP*  
*TRAILING*  
*TRANSFORM*  
*TRIM\_FUNCTION*  
*TYPE*  
*UNDERLINE*  
*UNIT*  
*UNLOCK*  
*UNSIGNED*  
*UNSIGNED\_INT*  
*UNSIGNED\_LONG*  
*UNSIGNED\_SHORT*  
*UNSTRING*  
*UNTIL*  
*UP*  
*UPDATE*  
*UPON*  
*UPON\_ARGUMENT\_NUMBER*  
*UPON\_COMMAND\_LINE*  
*UPON\_ENVIRONMENT\_NAME*  
*UPON\_ENVIRONMENT\_VALUE*  
*UPPER\_CASE\_FUNC*  
*USAGE*  
*USE*  
*USING*  
*VALUE*  
*VARYING*  
*WAIT*  
*WHEN*  
*WHEN\_COMPILED\_FUNC*  
*WITH*  
*WORD*  
*WORDS*  
*WORKING\_STORAGE*

**WRITE**  
**YYYYDDD**  
**YYYYMMDD**  
**ZERO**  
**UNARY\_SIGN**  
**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**  
**TOKEN\_EOF**  
**COPY**  
**REPLACE**  
**SUPPRESS**  
**PRINTING**  
**REPLACING**  
**OFF**  
**IN**  
**OF**  
**BY**  
**EQEQ**  
**TOKEN**

Definition at line 31 of file ppparse.h.

```
{  
    TOKEN_EOF = 0,  
    COPY = 258,  
    REPLACE = 259,  
    SUPPRESS = 260,  
    PRINTING = 261,  
    REPLACING = 262,  
    OFF = 263,  
    IN = 264,  
    OF = 265,  
    BY = 266,  
    EQEQ = 267,  
    TOKEN = 268  
};
```

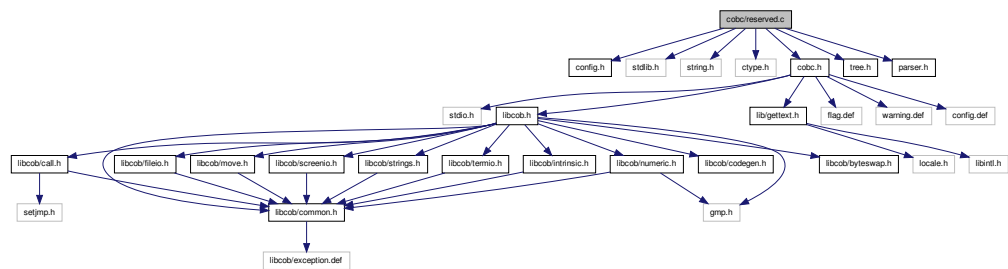
## 7.12.4 Variable Documentation

### 7.12.4.1 YYSTYPE ppval

## 7.13 cobc/reserved.c File Reference

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobc.h"
#include "tree.h"
#include "parser.h"
```

Include dependency graph for reserved.c:



## Classes

- struct [reserved](#)

## Defines

- #define [NUM\\_RESERVED\\_WORDS](#) sizeof (reserved\_words) / sizeof (struct [re-served](#))
- #define [NUM\\_INTRINSICS](#) sizeof(function\_list) / sizeof(struct [cb\\_intrinsic\\_table](#))

## Functions

- [cb\\_tree](#) [lookup\\_system\\_name](#) (const char \*name)
- int [lookup\\_reserved\\_word](#) (const char \*name)
- struct [cb\\_intrinsic\\_table](#) \* [lookup\\_intrinsic](#) (const char \*name, const int checkres)

- void [cb\\_list\\_reserved](#) (void)
- void [cb\\_list\\_intrinsics](#) (void)
- void [cb\\_list\\_mnemonics](#) (void)
- void [cb\\_init\\_reserved](#) (void)

### 7.13.1 Define Documentation

#### 7.13.1.1 #define NUM\_INTRINSICS sizeof(function\_list) / sizeof(struct cb\_intrinsic\_table)

Definition at line 878 of file reserved.c.

#### 7.13.1.2 #define NUM\_RESERVED\_WORDS sizeof(reserved\_words) / sizeof(struct reserved)

Definition at line 615 of file reserved.c.

### 7.13.2 Function Documentation

#### 7.13.2.1 void cb\_init\_reserved ( void )

Definition at line 1019 of file reserved.c.

```
{
    int    i;

    /* build system-name table */
    for (i = 0; system_table[i].name != NULL; ++i) {
        system_table[i].node =
            cb_build_system_name (system_table[i].category, system_table[i]
.token);
    }
}
```

#### 7.13.2.2 void cb\_list\_intrinsics ( void )

Definition at line 979 of file reserved.c.

```
{
    const char    *s;
    size_t        i;
    size_t        n;

    printf ("Intrinsic Function (Implemented Y/N)\n\n");
    for (i = 0; i < NUM_INTRINSICS; ++i) {
        n = strlen (function_list[i].name);
        switch (n / 8) {
            case 0:
                s = "\t\t\t\t\t";
                break;
            case 1:

```

```

        s = "\t\t\t";
        break;
    case 2:
        s = "\t\t";
        break;
    default:
        s = "\t";
        break;
    }
    printf ("%s%s(%s)\n", function_list[i].name, s,
           function_list[i].implemented ? "Y" : "N");
}
}

```

### 7.13.2.3 void cb\_list\_mnemonics ( void )

Definition at line 1008 of file reserved.c.

```

{
    size_t      i;

    printf ("Mnemonic names\n\n");
    for (i = 0; system_table[i].name != NULL; ++i) {
        printf ("%s\n", system_table[i].name);
    }
}

```

### 7.13.2.4 void cb\_list\_reserved ( void )

Definition at line 950 of file reserved.c.

```

{
    const char  *s;
    size_t      i;
    size_t      n;

    printf ("Reserved Words (Parsed Y/N)\n\n");
    for (i = 0; i < NUM_RESERVED_WORDS; ++i) {
        n = strlen (reserved_words[i].name);
        switch (n / 8) {
            case 0:
                s = "\t\t\t\t\t";
                break;
            case 1:
                s = "\t\t\t\t";
                break;
            case 2:
                s = "\t\t\t";
                break;
            default:
                s = "\t";
                break;
        }
        printf ("%s%s(%s)\n", reserved_words[i].name, s,
               reserved_words[i].token != -1 ? "Y" : "N");
    }
}

```



### 7.13.2.5 struct cb\_intrinsic\_table\* lookup\_intrinsic ( const char \* name, const int checkres ) [read]

Definition at line 929 of file reserved.c.

```
{
    struct cb_intrinsic_table    *cbp;
    struct noreserve            *noresp;

    if (checkres) {
        for (noresp = noresp; noresp; noresp = noresp->next) {
            if (strcasecmp (name, noresp->noresword) == 0) {
                return NULL;
            }
        }
    }
    cbp = bsearch (name, function_list, NUM_INTRINSICS,
                  sizeof (struct cb_intrinsic_table), intrinsic_comp);
    if (cbp && cbp->implemented) {
        return cbp;
    }
    return NULL;
}
```

### 7.13.2.6 int lookup\_reserved\_word ( const char \* name )

Definition at line 906 of file reserved.c.

```
{
    struct reserved *p;
    struct noreserve    *noresp;

    p = bsearch (name, reserved_words, NUM_RESERVED_WORDS,
                sizeof (struct reserved), reserve_comp);
    if (!p) {
        return 0;
    }
    for (noresp = noresp; noresp; noresp = noresp->next) {
        if (strcasecmp (name, noresp->noresword) == 0) {
            return 0;
        }
    }
    if (p->token != -1) {
        return p->token;
    }
    cb_error (_("%s' reserved word, but not supported yet"), name);
    return 0;
}
```

### 7.13.2.7 cb\_tree lookup\_system\_name ( const char \* name )

Definition at line 893 of file reserved.c.

```
{
```

```
int    i;

for (i = 0; system_table[i].name != NULL; ++i) {
    if (strcasecmp (name, system_table[i].name) == 0) {
        return system_table[i].node;
    }
}
return cb_error_node;
}
```

### 7.13.3 Variable Documentation

#### 7.13.3.1 enum `cb_system_name_category` category

Definition at line 33 of file reserved.c.

#### 7.13.3.2 `const char*` name

Definition at line 32 of file reserved.c.

#### 7.13.3.3 `cb_tree` node

Definition at line 35 of file reserved.c.

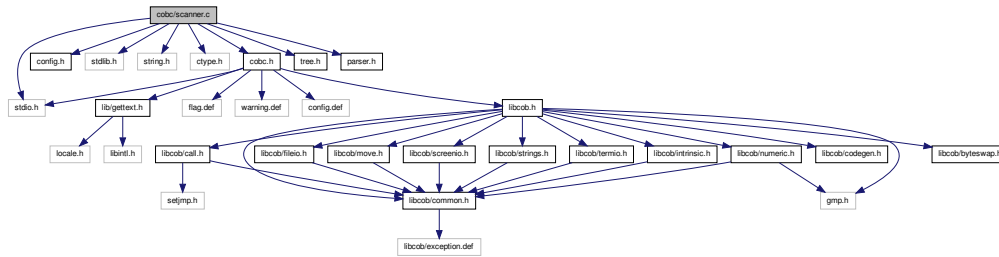
#### 7.13.3.4 `const int` token

Definition at line 34 of file reserved.c.

## 7.14 `cobc/scanner.c` File Reference

```
#include <stdio.h>
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobc.h"
#include "tree.h"
#include "parser.h"
```

Include dependency graph for scanner.c:



## Classes

- struct [yy\\_buffer\\_state](#)
- struct [cb\\_level\\_78](#)

## Defines

- #define [FLEX\\_SCANNER](#)
- #define [YY\\_FLEX\\_MAJOR\\_VERSION](#) 2
- #define [YY\\_FLEX\\_MINOR\\_VERSION](#) 5
- #define [yyconst](#)
- #define [YY\\_PROTO](#)(proto) ()
- #define [YY\\_NULL](#) 0
- #define [YY\\_SC\\_TO\\_UI](#)(c) ((unsigned int) (unsigned char) c)
- #define [BEGIN](#) yy\_start = 1 + 2 \*
- #define [YY\\_START](#) ((yy\_start - 1) / 2)
- #define [YYSTATE](#) YY\_START
- #define [YY\\_STATE\\_EOF](#)(state) (YY\_END\_OF\_BUFFER + state + 1)
- #define [YY\\_NEW\\_FILE](#) yyrestart( yyin )
- #define [YY\\_END\\_OF\\_BUFFER\\_CHAR](#) 0
- #define [YY\\_BUF\\_SIZE](#) 16384
- #define [EOB\\_ACT\\_CONTINUE\\_SCAN](#) 0
- #define [EOB\\_ACT\\_END\\_OF\\_FILE](#) 1
- #define [EOB\\_ACT\\_LAST\\_MATCH](#) 2
- #define [yyless](#)(n)
- #define [unput](#)(c) yyunput( c, yytext\_ptr )
- #define [YY\\_BUFFER\\_NEW](#) 0
- #define [YY\\_BUFFER\\_NORMAL](#) 1
- #define [YY\\_BUFFER\\_EOF\\_PENDING](#) 2
- #define [YY\\_CURRENT\\_BUFFER](#) yy\_current\_buffer
- #define [YY\\_FLUSH\\_BUFFER](#) yy\_flush\_buffer( yy\_current\_buffer )
- #define [yy\\_new\\_buffer](#) yy\_create\_buffer

- #define `yy_set_interactive`(is\_interactive)
- #define `yy_set_bol`(at\_bol)
- #define `YY_AT_BOL`() (yy\_current\_buffer->yy\_at\_bol)
- #define `yywrap`() 1
- #define `YY_SKIP_YWRAP`
- #define `yytext_ptr` yytext
- #define `YY_DO_BEFORE_ACTION`
- #define `YY_NUM_RULES` 99
- #define `YY_END_OF_BUFFER` 100
- #define `REJECT` reject\_used\_but\_not\_detected
- #define `yymore`() yymore\_used\_but\_not\_detected
- #define `YY_MORE_ADJ` 0
- #define `YY_RESTORE_YY_MORE_OFFSET`
- #define `INITIAL` 0
- #define `YY_NEVER_INTERACTIVE` 1
- #define `SET_LOCATION`(x)
- #define `DECIMAL_IS_PERIOD` 1
- #define `DECIMAL_IS_COMMA` 2
- #define `PICTURE_STATE` 3
- #define `FUNCTION_STATE` 4
- #define `YY_NO_PUSH_STATE` 1
- #define `YY_NO_POP_STATE` 1
- #define `YY_NO_TOP_STATE` 1
- #define `YY_READ_BUF_SIZE` 8192
- #define `ECHO` (void) fwrite( `yytext`, `yyleng`, 1, `yyout` )
- #define `YY_INPUT`(buf, result, max\_size)
- #define `yyterminate`() return YY\_NULL
- #define `YY_START_STACK_INCR` 25
- #define `YY_FATAL_ERROR`(msg) yy\_fatal\_error( msg )
- #define `YY_DECL` int yylex YY\_PROTO(( void ))
- #define `YY_USER_ACTION`
- #define `YY_BREAK` break;
- #define `YY_RULE_SETUP`
- #define `YY_EXIT_FAILURE` 2
- #define `yyles`(n)

## Typedefs

- typedef struct `yy_buffer_state` \* `YY_BUFFER_STATE`
- typedef unsigned int `yy_size_t`
- typedef unsigned char `YY_CHAR`
- typedef int `yy_state_type`

## Functions

- void `yyrestart YY_PROTO` ((FILE \*input\_file))
- void `yy_switch_to_buffer YY_PROTO` ((YY\_BUFFER\_STATE new\_buffer))
- void `yy_load_buffer_state YY_PROTO` ((void))
- YY\_BUFFER\_STATE `yy_create_buffer YY_PROTO` ((FILE \*file, int size))
- void `yy_delete_buffer YY_PROTO` ((YY\_BUFFER\_STATE b))
- void `yy_init_buffer YY_PROTO` ((YY\_BUFFER\_STATE b, FILE \*file))
- YY\_BUFFER\_STATE `yy_scan_buffer YY_PROTO` ((char \*base, yy\_size\_t size))
- YY\_BUFFER\_STATE `yy_scan_string YY_PROTO` ((yyconst char \*yy\_str))
- YY\_BUFFER\_STATE `yy_scan_bytes YY_PROTO` ((yyconst char \*bytes, int len))

## Variables

- int `yyleng`
- FILE \* `yyin` = (FILE \*) 0
- FILE \* `yyout` = (FILE \*) 0
- char \* `yytext`
- register char \* `yy_bp`
- int `size`
- FILE \* `file`
- int `len`

### 7.14.1 Define Documentation

#### 7.14.1.1 #define BEGIN yy\_start = 1 + 2 \*

Definition at line 80 of file scanner.c.

#### 7.14.1.2 #define DECIMAL\_IS\_COMMA 2

Definition at line 925 of file scanner.c.

#### 7.14.1.3 #define DECIMAL\_IS\_PERIOD 1

Definition at line 924 of file scanner.c.

#### 7.14.1.4 #define ECHO (void) fwrite( yytext, yyleng, 1, yyout )

Definition at line 1010 of file scanner.c.

#### 7.14.1.5 #define EOB\_ACT\_CONTINUE\_SCAN 0

Definition at line 105 of file scanner.c.

7.14.1.6 `#define EOB_ACT_END_OF_FILE 1`

Definition at line 106 of file scanner.c.

7.14.1.7 `#define EOB_ACT_LAST_MATCH 2`

Definition at line 107 of file scanner.c.

7.14.1.8 `#define FLEX_SCANNER`

Definition at line 8 of file scanner.c.

7.14.1.9 `#define FUNCTION_STATE 4`

Definition at line 928 of file scanner.c.

7.14.1.10 `#define INITIAL 0`

Definition at line 864 of file scanner.c.

7.14.1.11 `#define PICTURE_STATE 3`

Definition at line 927 of file scanner.c.

7.14.1.12 `#define REJECT reject_used_but_not_detected`

Definition at line 858 of file scanner.c.

7.14.1.13 `#define SET_LOCATION( x )`

**Value:**

```
(x)->source_file = (unsigned char *)cb_source_file;      \  
  (x)->source_line = cb_source_line
```

Definition at line 897 of file scanner.c.

7.14.1.14 `#define unput( c ) yyunput( c, yytext_ptr )`

Definition at line 136 of file scanner.c.

7.14.1.15 `#define YY_AT_BOL( ) (yy_current_buffer->yy_at_bol)`

Definition at line 263 of file scanner.c.

7.14.1.16 `#define YY_BREAK break;`

Definition at line 1069 of file scanner.c.

7.14.1.17 `#define YY_BUF_SIZE 16384`

Definition at line 98 of file scanner.c.

7.14.1.18 `#define YY_BUFFER_EOF_PENDING 2`

Definition at line 199 of file scanner.c.

7.14.1.19 `#define YY_BUFFER_NEW 0`

Definition at line 187 of file scanner.c.

7.14.1.20 `#define YY_BUFFER_NORMAL 1`

Definition at line 188 of file scanner.c.

7.14.1.21 `#define YY_CURRENT_BUFFER yy_current_buffer`

Definition at line 208 of file scanner.c.

7.14.1.22 `#define YY_DECL int yylex YY_PROTO(( void ))`

Definition at line 1057 of file scanner.c.

7.14.1.23 `#define YY_DO_BEFORE_ACTION`

**Value:**

```
yytext_ptr = yy_bp; \  
  yyleng = (int) (yy_cp - yy_bp); \  
  yy_hold_char = *yy_cp; \  
  *yy_cp = '\0'; \  
  yy_c_buf_p = yy_cp;
```

Definition at line 282 of file scanner.c.

7.14.1.24 `#define YY_END_OF_BUFFER 100`

Definition at line 290 of file scanner.c.

7.14.1.25 `#define YY_END_OF_BUFFER_CHAR 0`

Definition at line 95 of file scanner.c.

7.14.1.26 `#define YY_EXIT_FAILURE 2`7.14.1.27 `#define YY_FATAL_ERROR( msg ) yy_fatal_error( msg )`

Definition at line 1050 of file scanner.c.

7.14.1.28 `#define YY_FLEX_MAJOR_VERSION 2`

Definition at line 9 of file scanner.c.

7.14.1.29 `#define YY_FLEX_MINOR_VERSION 5`

Definition at line 10 of file scanner.c.

7.14.1.30 `#define YY_FLUSH_BUFFER yy_flush_buffer( yy_current_buffer )`

Definition at line 237 of file scanner.c.

7.14.1.31 `#define YY_INPUT( buf, result, max_size )`**Value:**

```

if ( yy_current_buffer->yy_is_interactive ) \
    { \
        int c = '*', n; \
        for ( n = 0; n < max_size && \
              (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
            buf[n] = (char) c; \
        if ( c == '\n' ) \
            buf[n++] = (char) c; \
        if ( c == EOF && ferror( yyin ) ) \
            YY_FATAL_ERROR( "input in flex scanner failed" ); \
        result = n; \
    } \
else if ( ((result = fread( buf, 1, max_size, yyin )) == 0) \
          && ferror( yyin ) ) \
        YY_FATAL_ERROR( "input in flex scanner failed" );

```

Definition at line 1017 of file scanner.c.

7.14.1.32 `#define YY_MORE_ADJ 0`

Definition at line 860 of file scanner.c.



7.14.1.33 `#define YY_NEVER_INTERACTIVE 1`

Definition at line 884 of file scanner.c.

7.14.1.34 `#define yy_new_buffer yy_create_buffer`

Definition at line 247 of file scanner.c.

7.14.1.35 `#define YY_NEW_FILE yyrestart( yyin )`

Definition at line 93 of file scanner.c.

7.14.1.36 `#define YY_NO_POP_STATE 1`

Definition at line 980 of file scanner.c.

7.14.1.37 `#define YY_NO_PUSH_STATE 1`

Definition at line 979 of file scanner.c.

7.14.1.38 `#define YY_NO_TOP_STATE 1`

Definition at line 981 of file scanner.c.

7.14.1.39 `#define YY_NULL 0`

Definition at line 67 of file scanner.c.

7.14.1.40 `#define YY_NUM_RULES 99`

Definition at line 289 of file scanner.c.

7.14.1.41 `#define YY_PROTO( proto )()`

Definition at line 63 of file scanner.c.

7.14.1.42 `#define YY_READ_BUF_SIZE 8192`

Definition at line 1001 of file scanner.c.

**7.14.1.43 #define YY\_RESTORE\_YY\_MORE\_OFFSET**

Definition at line 861 of file scanner.c.

**7.14.1.44 #define YY\_RULE\_SETUP****Value:**

```
if ( yyleng > 0 ) \  
    yy_current_buffer->yy_at_bol = \  
        (yytext[yyleng - 1] == '\n'); \  
    YY_USER_ACTION
```

Definition at line 1072 of file scanner.c.

**7.14.1.45 #define YY\_SC\_TO\_UI( c )((unsigned int)(unsigned char) c)**

Definition at line 74 of file scanner.c.

**7.14.1.46 #define yy\_set\_bol( at\_bol )****Value:**

```
{ \  
    if ( ! yy_current_buffer ) \  
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \  
    yy_current_buffer->yy_at_bol = at_bol; \  
}
```

Definition at line 256 of file scanner.c.

**7.14.1.47 #define yy\_set\_interactive( is\_interactive )****Value:**

```
{ \  
    if ( ! yy_current_buffer ) \  
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \  
    yy_current_buffer->yy_is_interactive = is_interactive; \  
}
```

Definition at line 249 of file scanner.c.

**7.14.1.48 #define YY\_SKIP\_YYWRAP**

Definition at line 267 of file scanner.c.

7.14.1.49 `#define YY_START ((yy_start - 1) / 2)`

Definition at line 86 of file scanner.c.

7.14.1.50 `#define YY_START_STACK_INCR 25`

Definition at line 1045 of file scanner.c.

7.14.1.51 `#define YY_STATE_EOF( state ) (YY_END_OF_BUFFER + state + 1)`

Definition at line 90 of file scanner.c.

7.14.1.52 `#define YY_USER_ACTION`

Definition at line 1064 of file scanner.c.

7.14.1.53 `#define yyconst`

Definition at line 56 of file scanner.c.

7.14.1.54 `#define yyless( n )`**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        yytext[yy leng] = yy_hold_char; \
        yy_c_buf_p = yytext + n; \
        yy_hold_char = *yy_c_buf_p; \
        *yy_c_buf_p = '\0'; \
        yy leng = n; \
    } \
while ( 0 )
```

Definition at line 125 of file scanner.c.

7.14.1.55 `#define yyless( n )`**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        *yy_cp = yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yy_c_buf_p = yy_cp = yy_bp + n - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
while ( 0 )
```

Definition at line 125 of file scanner.c.

7.14.1.56 `#define yymore( ) yymore_used_but_not_detected`

Definition at line 859 of file scanner.c.

7.14.1.57 `#define YYSTATE YY_START`

Definition at line 87 of file scanner.c.

7.14.1.58 `#define yyterminate( ) return YY_NULL`

Definition at line 1040 of file scanner.c.

7.14.1.59 `#define yytext_ptr yytext`

Definition at line 272 of file scanner.c.

7.14.1.60 `#define yywrap( ) 1`

Definition at line 266 of file scanner.c.

## 7.14.2 Typedef Documentation

7.14.2.1 `typedef struct yy_buffer_state* YY_BUFFER_STATE`

Definition at line 100 of file scanner.c.

7.14.2.2 `typedef unsigned char YY_CHAR`

Definition at line 268 of file scanner.c.

7.14.2.3 `typedef unsigned int yy_size_t`

Definition at line 142 of file scanner.c.

7.14.2.4 `typedef int yy_state_type`

Definition at line 270 of file scanner.c.

### 7.14.3 Function Documentation

7.14.3.1 `YY_BUFFER_STATE yy_scan_string YY_PROTO ( (yyconst char *yy_str) )`

7.14.3.2 `YY_BUFFER_STATE yy_scan_buffer YY_PROTO ( (char *base, yy_size_t size) )`

7.14.3.3 `void yy_delete_buffer YY_PROTO ( (YY_BUFFER_STATE b) )`

7.14.3.4 `void yy_switch_to_buffer YY_PROTO ( (YY_BUFFER_STATE new_buffer) )`

7.14.3.5 `void yy_load_buffer_state YY_PROTO ( (void) )`

7.14.3.6 `void yy_init_buffer YY_PROTO ( (YY_BUFFER_STATE b, FILE *file) )`

7.14.3.7 `YY_BUFFER_STATE yy_scan_bytes YY_PROTO ( (yyconst char *bytes, int len) )`

7.14.3.8 `void yyrestart YY_PROTO ( (FILE *input_file) )`

7.14.3.9 `YY_BUFFER_STATE yy_create_buffer YY_PROTO ( (FILE *file, int size) )`

### 7.14.4 Variable Documentation

7.14.4.1 `FILE* file`

Definition at line 2622 of file scanner.c.

7.14.4.2 `int len`

Definition at line 2744 of file scanner.c.

7.14.4.3 `yy_size_t size`

Definition at line 2533 of file scanner.c.

7.14.4.4 `register char* yy_bp`

Definition at line 2385 of file scanner.c.

7.14.4.5 `FILE* yyin = (FILE *) 0`

Definition at line 269 of file scanner.c.

7.14.4.6 `int yyleng`

Definition at line 217 of file scanner.c.

#### 7.14.4.7 FILE \* yyout = (FILE \*) 0

Definition at line 269 of file scanner.c.

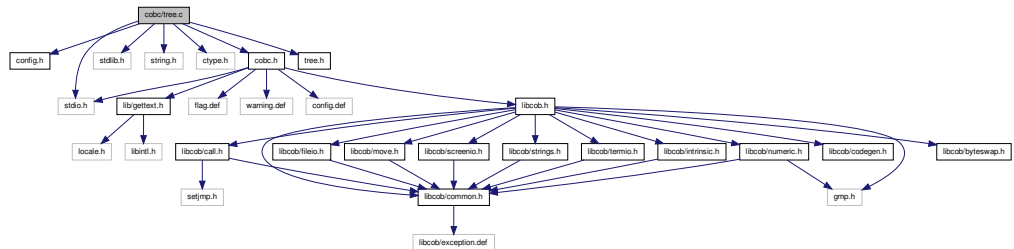
#### 7.14.4.8 char\* yytext

Definition at line 862 of file scanner.c.

### 7.15 cobb/tree.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "cobb.h"
#include "tree.h"
```

Include dependency graph for tree.c:



#### Classes

- struct **int\_node**

#### Defines

- #define **PIC\_ALPHABETIC** 0x01
- #define **PIC\_NUMERIC** 0x02
- #define **PIC\_NATIONAL** 0x04
- #define **PIC\_EDITED** 0x08
- #define **PIC\_ALPHANUMERIC** (PIC\_ALPHABETIC | PIC\_NUMERIC)

- #define `PIC_ALPHABETIC_EDITED` (`PIC_ALPHABETIC` | `PIC_EDITED`)
- #define `PIC_ALPHANUMERIC_EDITED` (`PIC_ALPHANUMERIC` | `PIC_EDITED`)
- #define `PIC_NUMERIC_EDITED` (`PIC_NUMERIC` | `PIC_EDITED`)
- #define `PIC_NATIONAL_EDITED` (`PIC_NATIONAL` | `PIC_EDITED`)

## Functions

- struct `cb_literal` \* `build_literal` (enum `cb_category` `category`, const unsigned char \*`data`, `size_t` `size`)
- char \* `cb_name` (`cb_tree` `x`)
- enum `cb_class` `cb_tree_class` (`cb_tree` `x`)
- enum `cb_category` `cb_tree_category` (`cb_tree` `x`)
- int `cb_tree_type` (`cb_tree` `x`)
- int `cb_fits_int` (`cb_tree` `x`)
- int `cb_fits_long_long` (`cb_tree` `x`)
- int `cb_get_int` (`cb_tree` `x`)
- long long `cb_get_long_long` (`cb_tree` `x`)
- void `cb_init_constants` (void)
- `cb_tree` `cb_build_list` (`cb_tree` `purpose`, `cb_tree` `value`, `cb_tree` `rest`)
- `cb_tree` `cb_list_append` (`cb_tree` `l1`, `cb_tree` `l2`)
- `cb_tree` `cb_list_add` (`cb_tree` `l`, `cb_tree` `x`)
- `cb_tree` `cb_list_reverse` (`cb_tree` `l`)
- int `cb_list_length` (`cb_tree` `l`)
- void `cb_list_map` (`cb_tree`(\*`func`)(`cb_tree` `x`), `cb_tree` `l`)
- struct `cb_program` \* `cb_build_program` (struct `cb_program` \*`last_program`, int `nest_level`)
- `cb_tree` `cb_int` (int `n`)
- `cb_tree` `cb_build_string` (const unsigned char \*`data`, `size_t` `size`)
- `cb_tree` `cb_build_alphabet_name` (`cb_tree` `name`, enum `cb_alphabet_name_type` `type`)
- `cb_tree` `cb_build_class_name` (`cb_tree` `name`, `cb_tree` `list`)
- `cb_tree` `cb_build_locale_name` (`cb_tree` `name`, `cb_tree` `list`)
- `cb_tree` `cb_build_system_name` (enum `cb_system_name_category` `category`, int `token`)
- `cb_tree` `cb_build_numeric_literal` (int `sign`, const unsigned char \*`data`, int `scale`)
- `cb_tree` `cb_build_alphanumeric_literal` (const unsigned char \*`data`, `size_t` `size`)
- `cb_tree` `cb_concat_literals` (`cb_tree` `x1`, `cb_tree` `x2`)
- `cb_tree` `cb_build_decimal` (int `id`)
- `cb_tree` `cb_build_picture` (const char \*`str`)
- `cb_tree` `cb_build_field` (`cb_tree` `name`)
- `cb_tree` `cb_build_implicit_field` (`cb_tree` `name`, int `len`)
- `cb_tree` `cb_build_constant` (`cb_tree` `name`, `cb_tree` `value`)
- struct `cb_field` \* `cb_field` (`cb_tree` `x`)
- struct `cb_field` \* `cb_field_add` (struct `cb_field` \*`f`, struct `cb_field` \*`p`)
- int `cb_field_size` (`cb_tree` `x`)
- struct `cb_field` \* `cb_field_founder` (struct `cb_field` \*`f`)

- struct `cb_field` \* `cb_field_variable_size` (struct `cb_field` \*f)
- struct `cb_field` \* `cb_field_variable_address` (struct `cb_field` \*f)
- int `cb_field_subordinate` (struct `cb_field` \*p, struct `cb_field` \*f)
- struct `cb_file` \* `build_file` (cb\_tree name)
- void `validate_file` (struct `cb_file` \*f, cb\_tree name)
- void `finalize_file` (struct `cb_file` \*f, struct `cb_field` \*records)
- cb\_tree `cb_build_reference` (const char \*name)
- cb\_tree `cb_build_filler` (void)
- cb\_tree `cb_build_field_reference` (struct `cb_field` \*f, cb\_tree ref)
- const char \* `cb_define` (cb\_tree name, cb\_tree val)
- void `cb_define_system_name` (const char \*name)
- cb\_tree `cb_ref` (cb\_tree x)
- cb\_tree `cb_build_binary_op` (cb\_tree x, int op, cb\_tree y)
- cb\_tree `cb_build_binary_list` (cb\_tree l, int op)
- cb\_tree `cb_build_funcall` (const char \*name, int argc, cb\_tree a1, cb\_tree a2, cb\_tree a3, cb\_tree a4, cb\_tree a5, cb\_tree a6, cb\_tree a7)
- cb\_tree `cb_build_cast` (enum `cb_cast_type` type, cb\_tree val)
- cb\_tree `cb_build_label` (cb\_tree name, struct `cb_label` \*section)
- cb\_tree `cb_build_assign` (cb\_tree var, cb\_tree val)
- cb\_tree `cb_build_initialize` (cb\_tree var, cb\_tree val, cb\_tree rep, cb\_tree def, int flag)
- cb\_tree `cb_build_search` (int flag\_all, cb\_tree table, cb\_tree var, cb\_tree end\_stmt, cb\_tree whens)
- cb\_tree `cb_build_call` (cb\_tree name, cb\_tree args, cb\_tree stmt1, cb\_tree stmt2, cb\_tree returning, int is\_system\_call)
- cb\_tree `cb_build_goto` (cb\_tree target, cb\_tree depending)
- cb\_tree `cb_build_if` (cb\_tree test, cb\_tree stmt1, cb\_tree stmt2)
- cb\_tree `cb_build_perform` (int type)
- cb\_tree `cb_build_perform_varying` (cb\_tree name, cb\_tree from, cb\_tree by, cb\_tree until)
- struct `cb_statement` \* `cb_build_statement` (const char \*name)
- cb\_tree `cb_build_continue` (void)
- cb\_tree `cb_build_any_intrinsic` (cb\_tree args)
- cb\_tree `cb_build_intrinsic` (cb\_tree name, cb\_tree args, cb\_tree refmod)

## Variables

- cb\_tree `cb_any`
- cb\_tree `cb_true`
- cb\_tree `cb_false`
- cb\_tree `cb_null`
- cb\_tree `cb_zero`
- cb\_tree `cb_one`
- cb\_tree `cb_space`
- cb\_tree `cb_low`
- cb\_tree `cb_high`



- [cb\\_tree cb\\_norm\\_low](#)
- [cb\\_tree cb\\_norm\\_high](#)
- [cb\\_tree cb\\_quote](#)
- [cb\\_tree cb\\_int0](#)
- [cb\\_tree cb\\_int1](#)
- [cb\\_tree cb\\_int2](#)
- [cb\\_tree cb\\_int3](#)
- [cb\\_tree cb\\_int4](#)
- [cb\\_tree cb\\_int5](#)
- [cb\\_tree cb\\_i \[8\]](#)
- [cb\\_tree cb\\_error\\_node](#)
- [cb\\_tree cb\\_intr\\_whencomp](#)
- [cb\\_tree cb\\_intr\\_pi](#)
- [cb\\_tree cb\\_intr\\_e](#)
- [cb\\_tree cb\\_standard\\_error\\_handler](#)
- [size\\_t gen\\_screen\\_ptr = 0](#)

### 7.15.1 Define Documentation

#### 7.15.1.1 `#define PIC_ALPHABETIC 0x01`

Definition at line 31 of file tree.c.

#### 7.15.1.2 `#define PIC_ALPHABETIC_EDITED (PIC_ALPHABETIC | PIC_EDITED)`

Definition at line 36 of file tree.c.

#### 7.15.1.3 `#define PIC_ALPHANUMERIC (PIC_ALPHABETIC | PIC_NUMERIC)`

Definition at line 35 of file tree.c.

#### 7.15.1.4 `#define PIC_ALPHANUMERIC_EDITED (PIC_ALPHANUMERIC | PIC_EDITED)`

Definition at line 37 of file tree.c.

#### 7.15.1.5 `#define PIC_EDITED 0x08`

Definition at line 34 of file tree.c.

#### 7.15.1.6 `#define PIC_NATIONAL 0x04`

Definition at line 33 of file tree.c.

7.15.1.7 `#define PIC_NATIONAL_EDITED (PIC_NATIONAL | PIC_EDITED)`

Definition at line 39 of file tree.c.

7.15.1.8 `#define PIC_NUMERIC 0x02`

Definition at line 32 of file tree.c.

7.15.1.9 `#define PIC_NUMERIC_EDITED (PIC_NUMERIC | PIC_EDITED)`

Definition at line 38 of file tree.c.

## 7.15.2 Function Documentation

7.15.2.1 `struct cb_file* build_file ( cb_tree name )` [read]

Definition at line 1577 of file tree.c.

```
{
    struct cb_file *p;

    p = make_tree (CB_TAG_FILE, CB_CATEGORY_UNKNOWN, sizeof (struct cb_file))
;
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);

    p->organization = COB_ORG_SEQUENTIAL;
    p->access_mode = COB_ACCESS_SEQUENTIAL;
    p->handler = CB_LABEL (cb_standard_error_handler);
    p->handler_prog = current_program;
    return p;
}
```

7.15.2.2 `struct cb_literal* build_literal ( enum cb_category category, const unsigned char * data, size_t size )` [read]

Definition at line 426 of file tree.c.

```
{
    struct cb_literal *p;

    p = make_tree (CB_TAG_LITERAL, category, sizeof (struct cb_literal));
    p->data = cobc_malloc ((size_t) (size + 1));
    p->size = size;
    memcpy (p->data, data, (size_t) size);
    /* RXW - malloc zeroes
    p->data[size] = 0;
    */
    return p;
}
```

**7.15.2.3 `cb_tree` `cb_build_alphabet_name` ( `cb_tree name`, enum `cb_alphabet_name_type type` )**

Definition at line 923 of file tree.c.

```
{
    struct cb_alphabet_name *p;

    p = make_tree (CB_TAG_ALPHABET_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_alphabet_name));
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);
    p->type = type;
    return CB_TREE (p);
}
```

**7.15.2.4 `cb_tree` `cb_build_alphanumeric_literal` ( const unsigned char \* `data`, size\_t `size` )**

Definition at line 999 of file tree.c.

```
{
    return CB_TREE (build_literal (CB_CATEGORY_ALPHANUMERIC, data, size));
}
```

**7.15.2.5 `cb_tree` `cb_build_any_intrinsic` ( `cb_tree args` )**

Definition at line 2253 of file tree.c.

```
{
    struct cb_intrinsic_table *cbp;

    cbp = lookup_intrinsic ("LENGTH", 0);
    return make_intrinsic (NULL, cbp, args, NULL, NULL);
}
```

**7.15.2.6 `cb_tree` `cb_build_assign` ( `cb_tree var`, `cb_tree val` )**

Definition at line 2098 of file tree.c.

```
{
    struct cb_assign *p;

    p = make_tree (CB_TAG_ASSIGN, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_assign));
    p->var = var;
    p->val = val;
    return CB_TREE (p);
}
```

7.15.2.7 `cb_tree cb_build_binary_list ( cb_tree l, int op )`

Definition at line 2019 of file tree.c.

```

{
    cb_tree e;

    e = CB_VALUE (l);
    for (l = CB_CHAIN (l); l; l = CB_CHAIN (l)) {
        e = cb_build_binary_op (e, op, CB_VALUE (l));
    }
    return e;
}

```

7.15.2.8 `cb_tree cb_build_binary_op ( cb_tree x, int op, cb_tree y )`

Definition at line 1954 of file tree.c.

```

{
    struct cb_binary_op *p;
    enum cb_category category = CB_CATEGORY_UNKNOWN;

    switch (op) {
    case '+':
    case '-':
    case '*':
    case '/':
    case '^':
        /* arithmetic operators */
        if (CB_TREE_CLASS (x) == CB_CLASS_POINTER ||
            CB_TREE_CLASS (y) == CB_CLASS_POINTER) {
            category = CB_CATEGORY_DATA_POINTER;
            break;
        }
        x = cb_check_numeric_value (x);
        y = cb_check_numeric_value (y);
        if (x == cb_error_node || y == cb_error_node) {
            return cb_error_node;
        }
        category = CB_CATEGORY_NUMERIC;
        break;

    case '=':
    case '~':
    case '<':
    case '>':
    case '[':
    case ']':
        /* relational operators */
        category = CB_CATEGORY_BOOLEAN;
        break;

    case '!':
    case '&':
    case '|':
        /* logical operators */
        if (CB_TREE_CLASS (x) != CB_CLASS_BOOLEAN ||
            (y && CB_TREE_CLASS (y) != CB_CLASS_BOOLEAN)) {

```

```

        cb_error (_("Invalid expression"));
        return cb_error_node;
    }
    category = CB_CATEGORY_BOOLEAN;
    break;

case '@':
    /* parentheses */
    category = CB_TREE_CATEGORY (x);
    break;

default:
    fprintf (stderr, "Unexpected operator -> %d\n", op);
    ABORT ();
}

p = make_tree (CB_TAG_BINARY_OP, category, sizeof (struct cb_binary_op));

p->op = op;
p->x = x;
p->y = y;
return CB_TREE (p);
}

```

#### 7.15.2.9 **cb\_tree** **cb\_build\_call** ( **cb\_tree** *name*, **cb\_tree** *args*, **cb\_tree** *stmt1*, **cb\_tree** *stmt2*, **cb\_tree** *returning*, **int** *is\_system\_call* )

Definition at line 2149 of file tree.c.

```

{
    struct cb_call *p;

    p = make_tree (CB_TAG_CALL, CB_CATEGORY_UNKNOWN, sizeof (struct cb_call))
;
    p->name = name;
    p->args = args;
    p->stmt1 = stmt1;
    p->stmt2 = stmt2;
    p->returning = returning;
    p->is_system = is_system_call;
    return CB_TREE (p);
}

```

#### 7.15.2.10 **cb\_tree** **cb\_build\_cast** ( **enum** **cb\_cast\_type** *type*, **cb\_tree** *val* )

Definition at line 2060 of file tree.c.

```

{
    struct cb_cast      *p;
    enum cb_category    category;

    if (type == CB_CAST_INTEGER) {
        category = CB_CATEGORY_NUMERIC;
    } else {
        category = CB_CATEGORY_UNKNOWN;
    }
}

```

```

    }
    p = make_tree (CB_TAG_CAST, category, sizeof (struct cb_cast));
    p->type = type;
    p->val = val;
    return CB_TREE (p);
}

```

#### 7.15.2.11 `cb_tree cb_build_class_name ( cb_tree name, cb_tree list )`

Definition at line 939 of file tree.c.

```

{
    struct cb_class_name    *p;
    char                    buff[COB_MINI_BUFF];

    p = make_tree (CB_TAG_CLASS_NAME, CB_CATEGORY_BOOLEAN, sizeof (struct
cb_class_name));
    p->name = cb_define (name, CB_TREE (p));
    snprintf (buff, COB_MINI_MAX, "is_%s", to_cname (p->name));
    p->cname = strdup (buff);
    p->list = list;
    return CB_TREE (p);
}

```

#### 7.15.2.12 `cb_tree cb_build_constant ( cb_tree name, cb_tree value )`

Definition at line 1446 of file tree.c.

```

{
    cb_tree x;

    x = cb_build_field (name);
    x->category = cb_tree_category (value);
    CB_FIELD (x)->storage = CB_STORAGE_CONSTANT;
    CB_FIELD (x)->values = cb_list_init (value);
    return x;
}

```

#### 7.15.2.13 `cb_tree cb_build_continue ( void )`

Definition at line 2240 of file tree.c.

```

{
    struct cb_continue *p;

    p = make_tree (CB_TAG_CONTINUE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_continue));
    return CB_TREE (p);
}

```

**7.15.2.14 `cb_tree cb_build_decimal ( int id )`**

Definition at line 1076 of file tree.c.

```
{
    struct cb_decimal *p;

    p = make_tree (CB_TAG_DECIMAL, CB_CATEGORY_NUMERIC, sizeof (struct
cb_decimal));
    p->id = id;
    return CB_TREE (p);
}
```

**7.15.2.15 `cb_tree cb_build_field ( cb_tree name )`**

Definition at line 1417 of file tree.c.

```
{
    struct cb_field *p;

    p = make_tree (CB_TAG_FIELD, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_field));
    p->id = cb_field_id++;
    p->name = cb_define (name, CB_TREE (p));
    p->ename = NULL;
    p->usage = CB_USAGE_DISPLAY;
    p->storage = CB_STORAGE_WORKING;
    p->occurs_max = 1;
    return CB_TREE (p);
}
```

**7.15.2.16 `cb_tree cb_build_field_reference ( struct cb_field * f, cb_tree ref )`**

Definition at line 1752 of file tree.c.

```
{
    cb_tree      x;
    struct cb_word *word;

    x = cb_build_reference (f->name);
    word = CB_REFERENCE (x)->word;
    if (ref) {
        memcpy (x, ref, sizeof (struct cb_reference));
    }
    x->category = CB_CATEGORY_UNKNOWN;
    CB_REFERENCE (x)->word = word;
    CB_REFERENCE (x)->value = CB_TREE (f);
    return x;
}
```

**7.15.2.17   `cb_tree cb_build_filler ( void )`**

Definition at line 1740 of file tree.c.

```

{
    cb_tree      x;
    char         name[16];

    sprintf (name, "WORK$%d", filler_id++);
    x = cb_build_reference (name);
    x->source_line = cb_source_line;
    return x;
}

```

**7.15.2.18   `cb_tree cb_build_funcall ( const char * name, int argc, cb_tree a1, cb_tree a2, cb_tree a3, cb_tree a4, cb_tree a5, cb_tree a6, cb_tree a7 )`**

Definition at line 2035 of file tree.c.

```

{
    struct cb_funcall *p;

    p = make_tree (CB_TAG_FUNCALL, CB_CATEGORY_BOOLEAN, sizeof (struct
cb_funcall));
    p->name = name;
    p->argc = argc;
    p->varcnt = 0;
    p->screenptr = gen_screen_ptr;
    p->argv[0] = a1;
    p->argv[1] = a2;
    p->argv[2] = a3;
    p->argv[3] = a4;
    p->argv[4] = a5;
    p->argv[5] = a6;
    p->argv[6] = a7;
    return CB_TREE (p);
}

```

**7.15.2.19   `cb_tree cb_build_goto ( cb_tree target, cb_tree depending )`**

Definition at line 2168 of file tree.c.

```

{
    struct cb_goto *p;

    p = make_tree (CB_TAG_GOTO, CB_CATEGORY_UNKNOWN, sizeof (struct cb_goto))
;
    p->target = target;
    p->depending = depending;
    return CB_TREE (p);
}

```



**7.15.2.20 `cb_tree cb_build_if ( cb_tree test, cb_tree stmt1, cb_tree stmt2 )`**

Definition at line 2183 of file tree.c.

```

{
    struct cb_if *p;

    p = make_tree (CB_TAG_IF, CB_CATEGORY_UNKNOWN, sizeof (struct cb_if));
    p->test = test;
    p->stmt1 = stmt1;
    p->stmt2 = stmt2;
    return CB_TREE (p);
}

```

**7.15.2.21 `cb_tree cb_build_implicit_field ( cb_tree name, int len )`**

Definition at line 1432 of file tree.c.

```

{
    cb_tree x;
    char    pic[32];

    x = cb_build_field (name);
    memset (pic, 0, sizeof(pic));
    sprintf (pic, "X(%d)", len);
    CB_FIELD (x)->pic = CB_PICTURE (cb_build_picture (pic));
    cb_validate_field (CB_FIELD (x));
    return x;
}

```

**7.15.2.22 `cb_tree cb_build_initialize ( cb_tree var, cb_tree val, cb_tree rep, cb_tree def, int flag )`**

Definition at line 2113 of file tree.c.

```

{
    struct cb_initialize *p;

    p = make_tree (CB_TAG_INITIALIZE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_initialize));
    p->var = var;
    p->val = val;
    p->rep = rep;
    p->def = def;
    p->flag_statement = flag;
    return CB_TREE (p);
}

```

**7.15.2.23 `cb_tree cb_build_intrinsic ( cb_tree name, cb_tree args, cb_tree refmod )`**

Definition at line 2262 of file tree.c.

```

{
    struct cb_intrinsic_table      *cbp;
    cb_tree                       x;
    int                           numargs;

    numargs = cb_list_length (args);

    cbp = lookup_intrinsic (CB_NAME (name), 0);
    if (cbp) {
        if ((cbp->args != -1 && numargs != cbp->args) ||
            (cbp->args == -1 && cbp->intr_enum != CB_INTR_RANDOM && numar
gs < 1)) {
            cb_error_x (name, _("FUNCTION %s has wrong number of argu
ments"), cbp->name);
            return cb_error_node;
        }
        if (refmod) {
            if (!cbp->refmod) {
                cb_error_x (name, _("FUNCTION %s can not have ref
erence modification"), cbp->name);
                return cb_error_node;
            }
            if (CB_LITERAL_P(CB_PAIR_X(refmod)) &&
                cb_get_int (CB_PAIR_X(refmod)) < 1) {
                cb_error_x (name, _("FUNCTION %s has invalid refe
rence modification"), cbp->name);
                return cb_error_node;
            }
            if (CB_PAIR_Y(refmod) && CB_LITERAL_P(CB_PAIR_Y(refmod))
&&
                cb_get_int (CB_PAIR_Y(refmod)) < 1) {
                cb_error_x (name, _("FUNCTION %s has invalid refe
rence modification"), cbp->name);
                return cb_error_node;
            }
        }
        /* cb_tree      x; */
        switch (cbp->intr_enum) {
        case CB_INTR_LENGTH:
        case CB_INTR_BYTE_LENGTH:
            x = CB_VALUE (args);
            if (CB_INTRINSIC_P (x)) {
                return make_intrinsic (name, cbp, args, NULL,
NULL);
            } else if ((CB_FIELD_P (x) || CB_REFERENCE_P (x)) &&
                cb_field(x)->flag_any_length) {
                return make_intrinsic (name, cbp, args, NULL,
NULL);
            } else {
                return cb_build_length (CB_VALUE (args));
            }
        case CB_INTR_WHEN_COMPILED:
            if (refmod) {
                return make_intrinsic (name, cbp,
                    cb_list_init (cb_intr_whencomp), NULL, re
fmod);
            } else {
                return cb_intr_whencomp;
            }
        case CB_INTR_PI:
            return cb_intr_pi;

```

```
        case CB_INTR_E:
            return cb_intr_e;

        case CB_INTR_LOWER_CASE:
        case CB_INTR_UPPER_CASE:
        case CB_INTR_REVERSE:
/* RXW Why did I do this ? - still do not know
        if (CB_INTRINSIC_P (CB_VALUE (args))) {
            return make_intrinsic (name, cbp, args, cb_int0);

        } else {
            return make_intrinsic (name, cbp, args,
                                   cb_build_length (CB_VALUE
(Args)));
        }
RXW */

        case CB_INTR_ABS:
        case CB_INTR_ACOS:
        case CB_INTR_ANNUITY:
        case CB_INTR_ASIN:
        case CB_INTR_ATAN:
        case CB_INTR_CHAR:
        case CB_INTR_COMBINED_DATETIME:
        case CB_INTR_COS:
        case CB_INTR_CURRENT_DATE:
        case CB_INTR_DATE_OF_INTEGER:
        case CB_INTR_DAY_OF_INTEGER:
        case CB_INTR_EXCEPTION_FILE:
        case CB_INTR_EXCEPTION_LOCATION:
        case CB_INTR_EXCEPTION_STATUS:
        case CB_INTR_EXCEPTION_STATEMENT:
        case CB_INTR_EXP:
        case CB_INTR_EXP10:
        case CB_INTR_FACTORIAL:
        case CB_INTR_FRACTION_PART:
        case CB_INTR_INTEGER:
        case CB_INTR_INTEGER_OF_DATE:
        case CB_INTR_INTEGER_OF_DAY:
        case CB_INTR_INTEGER_PART:
        case CB_INTR_LOCALE_DATE:
        case CB_INTR_LOCALE_TIME:
        case CB_INTR_LOCALE_TIME_FROM_SECS:
        case CB_INTR_LOG:
        case CB_INTR_LOG10:
        case CB_INTR_MOD:
        case CB_INTR_NUMVAL:
        case CB_INTR_NUMVAL_C:
        case CB_INTR_ORD:
        case CB_INTR_REM:
        case CB_INTR_SECONDS_FROM_FORMATTED_TIME:
        case CB_INTR_SECONDS_PAST_MIDNIGHT:
        case CB_INTR_SIGN:
        case CB_INTR_SIN:
        case CB_INTR_SQRT:
        case CB_INTR_STORED_CHAR_LENGTH:
        case CB_INTR_TAN:
        case CB_INTR_TEST_DATE_YYYYMMDD:
        case CB_INTR_TEST_DAY_YYYYDDD:
        case CB_INTR_TRIM:
            return make_intrinsic (name, cbp, args, NULL, refmod);
```

```

        case CB_INTR_CONCATENATE:
            return make_intrinsic (name, cbp, args, cb_int1, refmod);

        case CB_INTR_DATE_TO_YYYYMMDD:
        case CB_INTR_DAY_TO_YYYYDDD:
        case CB_INTR_MAX:
        case CB_INTR_MEAN:
        case CB_INTR_MEDIAN:
        case CB_INTR_MIDRANGE:
        case CB_INTR_MIN:
        case CB_INTR_ORD_MAX:
        case CB_INTR_ORD_MIN:
        case CB_INTR_PRESENT_VALUE:
        case CB_INTR_RANDOM:
        case CB_INTR_RANGE:
        case CB_INTR_STANDARD_DEVIATION:
        case CB_INTR_SUM:
        case CB_INTR_VARIANCE:
        case CB_INTR_YEAR_TO_YYYY:
            return make_intrinsic (name, cbp, args, cb_int1, NULL);
        case CB_INTR_SUBSTITUTE:
        case CB_INTR_SUBSTITUTE_CASE:
            if (numargs < 3 || (numargs % 2) == 0) {
                cb_error_x (name, _("FUNCTION %s has wrong number
of arguments"), cbp->name);
                return cb_error_node;
            }
            return make_intrinsic (name, cbp, args, cb_int1, refmod);

        default:
            break;
    }
}
cb_error_x (name, _("FUNCTION %s not implemented"), CB_NAME (name));
return cb_error_node;
}

```

#### 7.15.2.24 `cb_tree cb_build_label ( cb_tree name, struct cb_label * section )`

Definition at line 2081 of file tree.c.

```

{
    struct cb_label *p;

    p = make_tree (CB_TAG_LABEL, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_label));
    p->id = cb_id++;
    p->name = (const unsigned char *)cb_define (name, CB_TREE (p));
    p->orig_name = p->name;
    p->section = section;
    return CB_TREE (p);
}

```

#### 7.15.2.25 `cb_tree cb_build_list ( cb_tree purpose, cb_tree value, cb_tree rest )`

Definition at line 769 of file tree.c.

```
{
    struct cb_list *p;

    p = make_tree (CB_TAG_LIST, CB_CATEGORY_UNKNOWN, sizeof (struct cb_list))
;
    p->purpose = purpose;
    p->value = value;
    p->chain = rest;
    return CB_TREE (p);
}
```

#### 7.15.2.26 `cb_tree cb_build_locale_name ( cb_tree name, cb_tree list )`

Definition at line 957 of file tree.c.

```
{
    struct cb_class_name *p;

    p = make_tree (CB_TAG_LOCALE_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_locale_name));
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);
    p->list = list;
    return CB_TREE (p);
}
```

#### 7.15.2.27 `cb_tree cb_build_numeric_literal ( int sign, const unsigned char * data, int scale )`

Definition at line 988 of file tree.c.

```
{
    struct cb_literal *p;

    p = build_literal (CB_CATEGORY_NUMERIC, data, strlen ((char *)data));
    p->sign = (char)sign;
    p->scale = (char)scale;
    return CB_TREE (p);
}
```

#### 7.15.2.28 `cb_tree cb_build_perform ( int type )`

Definition at line 2199 of file tree.c.

```
{
    struct cb_perform *p;

    p = make_tree (CB_TAG_PERFORM, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_perform));
    p->type = type;
    return CB_TREE (p);
}
```

### 7.15.2.29 `cb_tree cb_build_perform_varying ( cb_tree name, cb_tree from, cb_tree by, cb_tree until )`

Definition at line 2209 of file tree.c.

```
{
    struct cb_perform_varying *p;

    p = make_tree (CB_TAG_PERFORM_VARYING, CB_CATEGORY_UNKNOWN, sizeof (struct
    cb_perform_varying));
    p->name = name;
    p->from = from;
    p->step = name ? cb_build_add (name, by, cb_high) : NULL;
    p->until = until;
    return CB_TREE (p);
}
```

### 7.15.2.30 `cb_tree cb_build_picture ( const char * str )`

Definition at line 1090 of file tree.c.

```
{
    struct cb_picture          *pic;
    const char                *p;
    size_t                    idx = 0;
    size_t                    buffcnt = 0;
    size_t                    at_beginning;
    size_t                    at_end;
    size_t                    p_char_seen;
    size_t                    s_char_seen;
    size_t                    category = 0;
    int                       size = 0;
    int                       allocated = 0;
    int                       digits = 0;
    int                       scale = 0;
    int                       s_count = 0;
    int                       v_count = 0;
    int                       i;
    int                       n;
    unsigned char             c;
    unsigned char             lastonechar = 0;
    unsigned char             lasttwochar = 0;
    unsigned char             buff[COB_SMALL_BUFF];

    pic = make_tree (CB_TAG_PICTURE, CB_CATEGORY_UNKNOWN, sizeof (struct
    cb_picture));
    if (strlen (str) > 50) {
        goto error;
    }
    memset (buff, 0, sizeof (buff));
    p_char_seen = 0;
    s_char_seen = 0;
    for (p = str; *p; p++) {
        n = 1;
        c = *p;
repeat:
        /* count the number of repeated chars */
        while (p[1] == c) {
```

```
        p++, n++;
    }

    /* add parenthesized numbers */
    if (p[1] == '(') {
        i = 0;
        p += 2;
        for (; *p == '0'; p++) {
            ;
        }
        for (; *p != ')'; p++) {
            if (!isdigit(*p)) {
                goto error;
            } else {
                allocated++;
                if (allocated > 9) {
                    goto error;
                }
                i = i * 10 + (*p - '0');
            }
        }
        if (i == 0) {
            goto error;
        }
        n += i - 1;
        goto repeat;
    }

    /* check grammar and category */
    /* FIXME: need more error check */
    switch (c) {
    case 'A':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_ALPHABETIC;
        break;

    case 'X':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_ALPHANUMERIC;
        break;

    case '9':
        category |= PIC_NUMERIC;
        digits += n;
        if (v_count) {
            scale += n;
        }
        break;

    case 'N':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_NATIONAL;
        break;

    case 'S':
        category |= PIC_NUMERIC;
```

```

        if (category & PIC_ALPHABETIC) {
            goto error;
        }
        s_count += n;
        if (s_count > 1 || idx != 0) {
            goto error;
        }
        s_char_seen = 1;
        continue;

case ',':
case '.':
    category |= PIC_NUMERIC_EDITED;
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    if (c != current_program->decimal_point) {
        break;
    }
    /* fall through */
case 'V':
    category |= PIC_NUMERIC;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    v_count += n;
    if (v_count > 1) {
        goto error;
    }
    break;

case 'P':
    category |= PIC_NUMERIC;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (p_char_seen) {
        goto error;
    }
    at_beginning = 0;
    at_end = 0;
    switch (buffcnt) {
    case 0:
        /* P..... */
        at_beginning = 1;
        break;
    case 1:
        /* VP.... */
        /* SP.... */
        if (lastonechar == 'V' || lastonechar == 'S') {
            at_beginning = 1;
        }
        break;
    case 2:
        /* SVP... */
        if (lasttwochar == 'S' && lastonechar == 'V') {
            at_beginning = 1;
        }
        break;
    }
    if (p[1] == 0 || (p[1] == 'V' && p[2] == 0)) {
        /* .....P */

```



```

        /* ....PV */
        at_end = 1;
    }
    if (!at_beginning && !at_end) {
        goto error;
    }
    p_char_seen = 1;
    if (at_beginning) {
        v_count++;      /* implicit V */
    }
    digits += n;
    if (v_count) {
        scale += n;
    } else {
        scale -= n;
    }
    break;

case '0':
case 'B':
case '/':
    category |= PIC_EDITED;
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    break;

case '*':
case 'Z':
    category |= PIC_NUMERIC_EDITED;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    digits += n;
    if (v_count) {
        scale += n;
    }
    break;

case '+':
case '-':
    category |= PIC_NUMERIC_EDITED;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    digits += n - 1;
    s_count++;
    /* FIXME: need more check */
    break;

case 'C':
    category |= PIC_NUMERIC_EDITED;
    if (!(p[1] == 'R' && p[2] == 0)) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {

```

```

        goto error;
    }
    p++;
    s_count++;
    break;

case 'D':
    category |= PIC_NUMERIC_EDITED;
    if (!(p[1] == 'B' && p[2] == 0)) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    p++;
    s_count++;
    break;

default:
    if (c == current_program->currency_symbol) {
        category |= PIC_NUMERIC_EDITED;
        digits += n - 1;
        /* FIXME: need more check */
        break;
    }

    goto error;
}

/* calculate size */
if (c != 'V' && c != 'P') {
    size += n;
}
if (c == 'C' || c == 'D' || c == 'N') {
    size += n;
}

/* store in the buffer */
buff[idx++] = c;
lasttwochar = lastonechar;
lastonechar = c;
memcpy (&buff[idx], (unsigned char *)&n, sizeof(int));
idx += sizeof(int);
++buffcnt;
}
buff[idx] = 0;

if (size == 0 && v_count) {
    goto error;
}

/* set picture */
pic->orig = strdup (str);
pic->size = size;
pic->digits = (unsigned char)digits;
pic->scale = (signed char)scale;
pic->have_sign = (unsigned char)s_count;

/* set picture category */
switch (category) {
case PIC_ALPHABETIC:
    pic->category = CB_CATEGORY_ALPHABETIC;
    break;

```

```

    case PIC_NUMERIC:
        pic->category = CB_CATEGORY_NUMERIC;
        if (digits > 36) {
            cb_error (_("Numeric field cannot be larger than 36 digit
s"));
        }
        break;
    case PIC_ALPHANUMERIC:
    case PIC_NATIONAL:
        pic->category = CB_CATEGORY_ALPHANUMERIC;
        break;
    case PIC_NUMERIC_EDITED:
        pic->str = cobc_malloc (idx + 1);
        memcpy (pic->str, buff, idx);
        pic->category = CB_CATEGORY_NUMERIC_EDITED;
        pic->lenstr = idx;
        break;
    case PIC_EDITED:
    case PIC_ALPHABETIC_EDITED:
    case PIC_ALPHANUMERIC_EDITED:
    case PIC_NATIONAL_EDITED:
        pic->str = cobc_malloc (idx + 1);
        memcpy (pic->str, buff, idx);
        pic->category = CB_CATEGORY_ALPHANUMERIC_EDITED;
        pic->lenstr = idx;
        break;
    default:
        goto error;
}
goto end;

error:
    cb_error (_("Invalid picture string - '%s'"), str);

end:
    return CB_TREE (pic);
}

```

#### 7.15.2.31 struct cb\_program\* cb\_build\_program ( struct cb\_program \* last\_program, int nest\_level ) [read]

Definition at line 841 of file tree.c.

```

{
    struct cb_program *p;

    cb_reset_78 ();
    cb_reset_in_procedure ();
    cb_clear_real_field ();
    p = cobc_malloc (sizeof (struct cb_program));
    p->next_program = last_program;
    p->nested_level = nest_level;
    p->decimal_point = '.';
    p->currency_symbol = '$';
    p->numeric_separator = ',';
    if (nest_level) {
        p->global_file_list = last_program->global_file_list;
        p->collating_sequence = last_program->collating_sequence;
        p->function_spec_list = last_program->function_spec_list;
    }
}

```

```

        p->class_spec_list = last_program->class_spec_list;
        p->interface_spec_list = last_program->interface_spec_list;
        p->program_spec_list = last_program->program_spec_list;
        p->property_spec_list = last_program->property_spec_list;
        p->alphabet_name_list = last_program->alphabet_name_list;
        p->class_name_list = last_program->class_name_list;
        p->locale_list = last_program->locale_list;
        p->symbolic_list = last_program->symbolic_list;
        p->decimal_point = last_program->decimal_point;
        p->numeric_separator = last_program->numeric_separator;
        p->currency_symbol = last_program->currency_symbol;
        p->cb_return_code = last_program->cb_return_code;
    } else {
        functions_are_all = cb_flag_functions_all;
    }
    return p;
}

```

### 7.15.2.32 `cb_tree cb_build_reference ( const char * name )`

Definition at line 1730 of file tree.c.

```

{
    struct cb_reference *p;

    p = make_tree (CB_TAG_REFERENCE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_reference));
    p->word = lookup_word (name);
    return CB_TREE (p);
}

```

### 7.15.2.33 `cb_tree cb_build_search ( int flag_all, cb_tree table, cb_tree var, cb_tree end_stmt, cb_tree whens )`

Definition at line 2131 of file tree.c.

```

{
    struct cb_search *p;

    p = make_tree (CB_TAG_SEARCH, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_search));
    p->flag_all = flag_all;
    p->table = table;
    p->var = var;
    p->end_stmt = end_stmt;
    p->whens = whens;
    return CB_TREE (p);
}

```

### 7.15.2.34 `struct cb_statement* cb_build_statement ( const char * name )` [read]

Definition at line 2226 of file tree.c.

```
{
    struct cb_statement *p;

    p = make_tree (CB_TAG_STATEMENT, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_statement));
    p->name = name;
    return p;
}
```

#### 7.15.2.35 `cb_tree cb_build_string ( const unsigned char * data, size_t size )`

Definition at line 908 of file tree.c.

```
{
    struct cb_string *p;

    p = make_tree (CB_TAG_STRING, CB_CATEGORY_ALPHANUMERIC, sizeof (struct
cb_string));
    p->size = size;
    p->data = data;
    return CB_TREE (p);
}
```

#### 7.15.2.36 `cb_tree cb_build_system_name ( enum cb_system_name_category category, int token )`

Definition at line 973 of file tree.c.

```
{
    struct cb_system_name *p;

    p = make_tree (CB_TAG_SYSTEM_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_system_name));
    p->category = category;
    p->token = token;
    return CB_TREE (p);
}
```

#### 7.15.2.37 `cb_tree cb_concat_literals ( cb_tree x1, cb_tree x2 )`

Definition at line 1005 of file tree.c.

```
{
    unsigned char      *buff;
    cb_tree            x;
    unsigned char      *data1;
    unsigned char      *data2;
    size_t             size1;
    size_t             size2;

    if (x1 == cb_error_node || x2 == cb_error_node) {
```

```

        return cb_error_node;
    }
    if (CB_LITERAL_P (x1)) {
        data1 = CB_LITERAL (x1)->data;
        size1 = CB_LITERAL (x1)->size;
    } else if (CB_CONST_P (x1)) {
        size1 = 1;
        if (x1 == cb_space) {
            data1 = (unsigned char *) " ";
        } else if (x1 == cb_zero) {
            data1 = (unsigned char *) "0";
        } else if (x1 == cb_quote) {
            data1 = (unsigned char *) "\"";
        } else if (x1 == cb_norm_low) {
            data1 = (unsigned char *) "\0";
        } else if (x1 == cb_norm_high) {
            data1 = (unsigned char *) "\255";
        } else if (x1 == cb_null) {
            data1 = (unsigned char *) "\0";
        } else {
            return cb_error_node;
        }
    } else {
        return cb_error_node;
    }
}
if (CB_LITERAL_P (x2)) {
    data2 = CB_LITERAL (x2)->data;
    size2 = CB_LITERAL (x2)->size;
} else if (CB_CONST_P (x2)) {
    size2 = 1;
    if (x2 == cb_space) {
        data2 = (unsigned char *) " ";
    } else if (x2 == cb_zero) {
        data2 = (unsigned char *) "0";
    } else if (x2 == cb_quote) {
        data2 = (unsigned char *) "\"";
    } else if (x2 == cb_norm_low) {
        data2 = (unsigned char *) "\0";
    } else if (x2 == cb_norm_high) {
        data2 = (unsigned char *) "\255";
    } else if (x2 == cb_null) {
        data2 = (unsigned char *) "\0";
    } else {
        return cb_error_node;
    }
} else {
    return cb_error_node;
}
buff = cobc_malloc (size1 + size2 + 3);
memcpy (buff, data1, size1);
memcpy (buff + size1, data2, size2);
x = cb_build_alphanumeric_literal (buff, size1 + size2);
free (buff);
return x;
}

```

### 7.15.2.38 const char\* cb.define ( cb\_tree name, cb\_tree val )

Definition at line 1769 of file tree.c.

```
{
    struct cb_word *w;

    w = CB_REFERENCE (name)->word;
    w->items = cb_list_add (w->items, val);
    w->count++;
    val->source_file = name->source_file;
    val->source_line = name->source_line;
    CB_REFERENCE (name)->value = val;
    return w->name;
}
```

#### 7.15.2.39 void cb\_define\_system\_name ( const char \* name )

Definition at line 1783 of file tree.c.

```
{
    cb_tree x;

    x = cb_build_reference (name);
    if (CB_REFERENCE (x)->word->count == 0) {
        cb_define (x, lookup_system_name (name));
    }
}
```

#### 7.15.2.40 struct cb\_field\* cb\_field ( cb\_tree x ) [read]

Definition at line 1458 of file tree.c.

```
{
    if (CB_REFERENCE_P (x)) {
        return CB_FIELD (cb_ref (x));
    } else {
        return CB_FIELD (x);
    }
}
```

#### 7.15.2.41 struct cb\_field\* cb\_field.add ( struct cb\_field \* f, struct cb\_field \* p ) [read]

Definition at line 1468 of file tree.c.

```
{
    struct cb_field *t;

    if (f == NULL) {
        return p;
    }
    for (t = f; t->sister; t = t->sister) {
        ;
    }
    t->sister = p;
    return f;
}
```

**7.15.2.42 struct cb\_field\* cb\_field\_founder ( struct cb\_field \* f )** [read]

Definition at line 1521 of file tree.c.

```

{
    while (f->parent) {
        f = f->parent;
    }
    return f;
}

```

**7.15.2.43 int cb\_field\_size ( cb\_tree x )**

Definition at line 1483 of file tree.c.

```

{
    struct cb_reference    *r;
    struct cb_field       *f;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_LITERAL:
        return CB_LITERAL (x)->size;
    case CB_TAG_FIELD:
        return CB_FIELD (x)->size;
    case CB_TAG_REFERENCE:
        r = CB_REFERENCE (x);
        f = CB_FIELD (r->value);

        if (r->length) {
            if (CB_LITERAL_P (r->length)) {
                return cb_get_int (r->length);
            } else {
                return -1;
            }
        } else if (r->offset) {
            if (CB_LITERAL_P (r->offset)) {
                return f->size - cb_get_int (r->offset) + 1;
            } else {
                return -1;
            }
        } else {
            return f->size;
        }
    default:
        fprintf (stderr, "Unexpected tree tag %d\n", CB_TREE_TAG (x));
        ABORT ();
    }
}
/* NOT REACHED */
return 0;
}

```

**7.15.2.44 int cb\_field\_subordinate ( struct cb\_field \* p, struct cb\_field \* f )**

Definition at line 1562 of file tree.c.



```

{
    for (p = p->parent; p; p = p->parent) {
        if (p == f) {
            return 1;
        }
    }
    return 0;
}

```

#### 7.15.2.45 struct cb\_field\* cb\_field\_variable\_address ( struct cb\_field \* f ) [read]

Definition at line 1545 of file tree.c.

```

{
    struct cb_field *p;

    for (p = f->parent; p; f = f->parent, p = f->parent) {
        for (p = p->children; p != f; p = p->sister) {
            if (p->occursDepending || cb_field_variable_size (p)) {
                return p;
            }
        }
    }
    return NULL;
}

```

#### 7.15.2.46 struct cb\_field\* cb\_field\_variable\_size ( struct cb\_field \* f ) [read]

Definition at line 1530 of file tree.c.

```

{
    struct cb_field *p;

    for (f = f->children; f; f = f->sister) {
        if (f->occursDepending) {
            return f;
        } else if ((p = cb_field_variable_size (f)) != NULL) {
            return p;
        }
    }
    return NULL;
}

```

#### 7.15.2.47 int cb.fits\_int ( cb\_tree x )

Definition at line 587 of file tree.c.

```

{
    struct cb_literal    *l;
    struct cb_field     *f;

    switch (CB_TREE_TAG (x)) {

```

```

case CB_TAG_LITERAL:
    l = CB_LITERAL (x);
    if (l->scale <= 0 && l->size < 10) {
        return 1;
    }
    return 0;
case CB_TAG_FIELD:
    f = CB_FIELD (x);
    switch (f->usage) {
case CB_USAGE_INDEX:
case CB_USAGE_LENGTH:
        return 1;
case CB_USAGE_BINARY:
case CB_USAGE_COMP_5:
case CB_USAGE_COMP_X:
        if (f->pic->scale <= 0 && f->size <= (int)sizeof (int)) {
            return 1;
        }
        return 0;
case CB_USAGE_DISPLAY:
        if (f->size < 10) {
            if (!f->pic || f->pic->scale <= 0) {
                return 1;
            }
        }
        return 0;
case CB_USAGE_PACKED:
        if (f->pic->scale <= 0 && f->pic->digits < 10) {
            return 1;
        }
        return 0;
default:
        return 0;
    }
case CB_TAG_REFERENCE:
    return cb_fits_int (CB_REFERENCE (x)->value);
default:
    return 0;
}
}

```

#### 7.15.2.48 int cb\_fits\_long\_long ( cb\_tree x )

Definition at line 635 of file tree.c.

```

{
    struct cb_literal    *l;
    struct cb_field     *f;

    switch (CB_TREE_TAG (x)) {
case CB_TAG_LITERAL:
        l = CB_LITERAL (x);
        if (l->scale <= 0 && l->size < 19) {
            return 1;
        }
        return 0;
case CB_TAG_FIELD:
        f = CB_FIELD (x);

```

```

        switch (f->usage) {
        case CB_USAGE_INDEX:
        case CB_USAGE_LENGTH:
            return 1;
        case CB_USAGE_BINARY:
        case CB_USAGE_COMP_5:
        case CB_USAGE_COMP_X:
            if (f->pic->scale <= 0 && f->size <= (int)sizeof (long lo
ng)) {
                return 1;
            }
            return 0;
        case CB_USAGE_DISPLAY:
            if (f->pic->scale <= 0 && f->size < 19) {
                return 1;
            }
            return 0;
        default:
            return 0;
        }
    case CB_TAG_REFERENCE:
        return cb_fits_long_long (CB_REFERENCE (x)->value);
    default:
        return 0;
    }
}

```

#### 7.15.2.49 int cb\_get\_int ( cb\_tree x )

Definition at line 676 of file tree.c.

```

{
    struct cb_literal    *l;
    size_t               i;
    int                  val = 0;

    l = CB_LITERAL (x);
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            break;
        }
    }

    /* RXWRXW
       if (l->size - i >= 10) {
           ABORT ();
       }
    */

    for (; i < l->size; i++) {
        val = val * 10 + l->data[i] - '0';
    }
    if (l->sign < 0) {
        val = -val;
    }
    return val;
}

```

## 7.15.2.50 long long cb\_get\_long\_long ( cb\_tree x )

Definition at line 705 of file tree.c.

```

{
    struct cb_literal    *l;
    size_t              i;
    long long           val = 0;

    l = CB_LITERAL (x);
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            break;
        }
    }

    if (l->size - i >= 19) {
        ABORT ();
    }

    for (; i < l->size; i++) {
        val = val * 10 + l->data[i] - '0';
    }
    if (l->sign < 0) {
        val = -val;
    }
    return val;
}

```

## 7.15.2.51 void cb\_init\_constants ( void )

Definition at line 732 of file tree.c.

```

{
    char    *s;
    int     i;

    cb_error_node = make_constant (CB_CATEGORY_UNKNOWN, NULL);
    cb_any = make_constant (CB_CATEGORY_UNKNOWN, NULL);
    cb_true = make_constant (CB_CATEGORY_BOOLEAN, "1");
    cb_false = make_constant (CB_CATEGORY_BOOLEAN, "0");
    cb_null = make_constant (CB_CATEGORY_DATA_POINTER, "0");
    cb_zero = make_constant (CB_CATEGORY_NUMERIC, "&cob_zero");
    cb_one = make_constant (CB_CATEGORY_NUMERIC, "&cob_one");
    cb_space = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_space");
    cb_low = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_low");
    cb_norm_low = cb_low;
    cb_high = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_high");
    cb_norm_high = cb_high;
    cb_quote = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_quote");
    cb_int0 = cb_int (0);
    cb_int1 = cb_int (1);
    cb_int2 = cb_int (2);
    cb_int3 = cb_int (3);
    cb_int4 = cb_int (4);
    cb_int5 = cb_int (5);
    for (i = 1; i < 8; i++) {
        s = cobc_malloc (4);
    }
}

```

```

        sprintf (s, "%d", i);
        cb_i[i] = make_constant (CB_CATEGORY_NUMERIC, s);
    }
    cb_standard_error_handler = make_constant_label ("Default Error Handler")
;
}

```

#### 7.15.2.52 `cb_tree cb_int ( int n )`

Definition at line 881 of file tree.c.

```

{
    struct cb_integer      *x;
    struct int_node       *p;

    for (p = int_node_table; p; p = p->next) {
        if (p->n == n) {
            return p->node;
        }
    }

    x = make_tree (CB_TAG_INTEGER, CB_CATEGORY_NUMERIC, sizeof (struct
cb_integer));
    x->val = n;

    p = cobc_malloc (sizeof (struct int_node));
    p->n = n;
    p->node = CB_TREE (x);
    p->next = int_node_table;
    int_node_table = p;
    return p->node;
}

```

#### 7.15.2.53 `cb_tree cb_list.add ( cb_tree l, cb_tree x )`

Definition at line 798 of file tree.c.

```

{
    return cb_list_append (l, cb_list_init (x));
}

```

#### 7.15.2.54 `cb_tree cb_list.append ( cb_tree l1, cb_tree l2 )`

Definition at line 781 of file tree.c.

```

{
    cb_tree l;

    if (l1 == NULL) {
        return l2;
    } else {
        l = l1;
    }
}

```

```
        while (CB_CHAIN (l)) {
            l = CB_CHAIN (l);
        }
        CB_CHAIN (l) = 12;
        return l1;
    }
}
```

#### 7.15.2.55 int cb\_list\_length ( cb\_tree l )

Definition at line 818 of file tree.c.

```
{
    int n = 0;

    for (; l; l = CB_CHAIN (l)) {
        n++;
    }
    return n;
}
```

#### 7.15.2.56 void cb\_list\_map ( cb\_tree\*(cb\_tree x) func, cb\_tree l )

Definition at line 829 of file tree.c.

```
{
    for (; l; l = CB_CHAIN (l)) {
        CB_VALUE (l) = func (CB_VALUE (l));
    }
}
```

#### 7.15.2.57 cb\_tree cb\_list\_reverse ( cb\_tree l )

Definition at line 804 of file tree.c.

```
{
    cb_tree next;
    cb_tree last = NULL;

    for (; l; l = next) {
        next = CB_CHAIN (l);
        CB_CHAIN (l) = last;
        last = l;
    }
    return last;
}
```

## 7.15.2.58 char\* cb\_name ( cb\_tree x )

Definition at line 441 of file tree.c.

```

{
    if (!treenamebuff) {
        treenamebuff = cobc_malloc (COB_NORMAL_BUFF);
    }
    cb_name_1 (treenamebuff, x);
    return treenamebuff;
}

```

## 7.15.2.59 cb\_tree cb\_ref ( cb\_tree x )

Definition at line 1794 of file tree.c.

```

{
    struct cb_reference    *r;
    struct cb_field       *p;
    struct cb_label       *s;
    cb_tree               candidate = NULL;
    cb_tree               items;
    cb_tree               cb1;
    cb_tree               cb2;
    cb_tree               v;
    cb_tree               c;
    struct cb_program     *prog;
    struct cb_word        *w;
    size_t                val;
    size_t                ambiguous = 0;

    r = CB_REFERENCE (x);
    /* if this reference has already been resolved (and the value
       has been cached), then just return the value */
    if (r->value) {
        return r->value;
    }
    /* resolve the value */

    items = r->word->items;
    for (; items; items = CB_CHAIN (items)) {
        /* find a candidate value by resolving qualification */
        v = CB_VALUE (items);
        c = r->chain;
        switch (CB_TREE_TAG (v)) {
            case CB_TAG_FIELD:
                /* in case the value is a field, it might be qualified
                   by its parent names and a file name */
                if (CB_FIELD (v)->flag_indexed_by) {
                    p = CB_FIELD (v)->index_qual;
                } else {
                    p = CB_FIELD (v)->parent;
                }
                /* resolve by parents */
                for (; p; p = p->parent) {
                    if (c && strcasecmp (CB_NAME (c), p->name) == 0)
                {
                    c = CB_REFERENCE (c)->chain;

```

```

    }
}

/* resolve by file */
if (c && CB_REFERENCE (c)->chain == NULL) {
    if (CB_REFERENCE (c)->word->count == 1 &&
CB_FILE_P (cb_ref (c))
        && (CB_FILE (cb_ref (c)) == cb_field_founder
(CB_FIELD (v))->file)) {
            c = CB_REFERENCE (c)->chain;
        }
    }

break;
case CB_TAG_LABEL:
/* in case the value is a label, it might be qualified
by its section name */
s = CB_LABEL (v)->section;

/* unqualified paragraph name referenced within the secti
on
is resolved without ambiguity check if not duplicated
*/
if (c == NULL && r->offset && s == CB_LABEL (r->offset))
{
    for (cb1 = CB_CHAIN (items); cb1; cb1 = CB_CHAIN
(cb1)) {
        cb2 = CB_VALUE (cb1);
        if (s == CB_LABEL (cb2)->section) {
            ambiguous_error (x);
            goto error;
        }
    }
    candidate = v;
    goto end;
}

/* resolve by section name */
if (c && s && strcasecmp (CB_NAME (c), (char *)s->name) =
= 0) {
    c = CB_REFERENCE (c)->chain;
}

break;
default:
/* other values cannot be qualified */
break;
}

/* a well qualified value is a good candidate */
if (c == NULL) {
    if (candidate == NULL) {
        /* keep the first candidate */
        candidate = v;
    } else {
        /* multiple candidates and possibly ambiguous */
        ambiguous = 1;
        /* continue search because the reference might no
t
be ambiguous and exit loop by "goto end" later
*/
    }
}

```



```

    }
}

/* there is no candidate */
if (candidate == NULL) {
    if (current_program->nested_level > 0) {
        /* Nested program - check parents for GLOBAL candidate */

        ambiguous = 0;
        val = hash ((const unsigned char *)r->word->name);
        prog = current_program->next_program;
        for (; prog; prog = prog->next_program) {
            if (prog->nested_level >= current_program->
nested_level) {
                continue;
            }
            for (w = prog->word_table[val]; w; w = w->next) {

                if (strcasecmp (r->word->name, w->name) =
= 0) {
                    candidate = global_check (r, w->
items, &ambiguous);

                    if (candidate) {
                        if (ambiguous) {
                            ambiguous_error (
x);

                                goto error;
                            }
                            if (CB_FILE_P(candidate))
                                current_program->
gen_file_error = 1;
                            }
                            goto end;
                        }
                    }
                }
            }
            if (prog->nested_level == 0) {
                break;
            }
        }
        }
        undefined_error (x);
        goto error;
    }

/* the reference is ambiguous */
if (ambiguous) {
    ambiguous_error (x);
    goto error;
}

end:
if (CB_FIELD_P (candidate)) {
    CB_FIELD (candidate)->count++;
    if (CB_FIELD (candidate)->flag_invalid) {
        goto error;
    }
}

r->value = candidate;
return r->value;

```

```

error:
    r->value = cb_error_node;
    return cb_error_node;
}

```

### 7.15.2.60 enum `cb_category` `cb_tree_category` ( `cb_tree x` )

Definition at line 458 of file `tree.c`.

```

{
    struct cb_cast      *p;
    struct cb_reference *r;
    struct cb_field     *f;

    if (x == cb_error_node) {
        return 0;
    }
    if (x->category != CB_CATEGORY_UNKNOWN) {
        return x->category;
    }

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_CAST:
        p = CB_CAST (x);
        switch (p->type) {
        case CB_CAST_ADDRESS:
        case CB_CAST_ADDR_OF_ADDR:
            x->category = CB_CATEGORY_DATA_POINTER;
            break;
        case CB_CAST_PROGRAM_POINTER:
            x->category = CB_CATEGORY_PROGRAM_POINTER;
            break;
        default:
            fprintf (stderr, "Unexpected cast type -> %d\n", p->type)
;
            ABORT ();
        }
        break;
    case CB_TAG_REFERENCE:
        r = CB_REFERENCE (x);
        if (r->offset) {
            x->category = CB_CATEGORY_ALPHANUMERIC;
        } else {
            x->category = cb_tree_category (r->value);
        }
        break;
    case CB_TAG_FIELD:
        f = CB_FIELD (x);
        if (f->children) {
            x->category = CB_CATEGORY_ALPHANUMERIC;
        } else if (f->usage == CB_USAGE_POINTER && f->level != 88) {
            x->category = CB_CATEGORY_DATA_POINTER;
        } else if (f->usage == CB_USAGE_PROGRAM_POINTER && f->level != 88
) {
            x->category = CB_CATEGORY_PROGRAM_POINTER;
        } else {
            switch (f->level) {
            case 66:

```

```

        if (f->rename_thru) {
            x->category = CB_CATEGORY_ALPHANUMERIC;
        } else {
            x->category = cb_tree_category (CB_TREE (
f->redefines));
        }
        break;
    case 88:
        x->category = CB_CATEGORY_BOOLEAN;
        break;
    default:
        x->category = f->pic->category;
        break;
    }
    break;
case CB_TAG_ALPHABET_NAME:
case CB_TAG_LOCALE_NAME:
    x->category = CB_CATEGORY_ALPHANUMERIC;
    break;
case CB_TAG_BINARY_OP:
    x->category = CB_CATEGORY_BOOLEAN;
    break;
default:
    fprintf (stderr, "Unknown tree tag %d Category %d\n",
CB_TREE_TAG (x), x->category);
    ABORT ();
}

return x->category;
}

```

#### 7.15.2.61 enum cb\_class cb\_tree\_class ( cb\_tree x )

Definition at line 451 of file tree.c.

```

{
    return category_to_class_table[CB_TREE_CATEGORY (x)];
}

```

#### 7.15.2.62 int cb\_tree\_type ( cb\_tree x )

Definition at line 537 of file tree.c.

```

{
    struct cb_field *f;

    f = cb_field (x);
    if (f->children) {
        return COB_TYPE_GROUP;
    }

    switch (CB_TREE_CATEGORY (x)) {
    case CB_CATEGORY_ALPHABETIC:

```

```

case CB_CATEGORY_ALPHANUMERIC:
    return COB_TYPE_ALPHANUMERIC;
case CB_CATEGORY_ALPHANUMERIC_EDITED:
    return COB_TYPE_ALPHANUMERIC_EDITED;
case CB_CATEGORY_NUMERIC:
    switch (f->usage) {
case CB_USAGE_DISPLAY:
    return COB_TYPE_NUMERIC_DISPLAY;
case CB_USAGE_BINARY:
case CB_USAGE_COMP_5:
case CB_USAGE_COMP_X:
case CB_USAGE_INDEX:
case CB_USAGE_LENGTH:
    return COB_TYPE_NUMERIC_BINARY;
case CB_USAGE_FLOAT:
    return COB_TYPE_NUMERIC_FLOAT;
case CB_USAGE_DOUBLE:
    return COB_TYPE_NUMERIC_DOUBLE;
case CB_USAGE_PACKED:
    return COB_TYPE_NUMERIC_PACKED;
default:
    fprintf (stderr, "Unexpected numeric usage -> %d\n", f->
usage);
        ABORT ();
    }
case CB_CATEGORY_NUMERIC_EDITED:
    return COB_TYPE_NUMERIC_EDITED;
case CB_CATEGORY_OBJECT_REFERENCE:
case CB_CATEGORY_DATA_POINTER:
case CB_CATEGORY_PROGRAM_POINTER:
    return COB_TYPE_NUMERIC_BINARY;
default:
    fprintf (stderr, "Unexpected category -> %d\n", CB_TREE_CATEGORY
(x));
        ABORT ();
    }
}
/* NOT REACHED */
return 0;
}

```

### 7.15.2.63 void finalize\_file ( struct cb\_file \* f, struct cb\_field \* records )

Definition at line 1611 of file tree.c.

```

{
    struct cb_field *p;
    struct cb_field *v;
    cb_tree      l;
    cb_tree      x;
    char          buff[COB_MINI_BUFF];

    if (f->special) {
        f->organization = COB_ORG_LINE_SEQUENTIAL;
    }
    if (f->fileid_assign && !f->assign) {
        f->assign = cb_build_alphanumeric_literal ((unsigned char *)f->
name,
                                                strlen (f->name));
    }
}

```

```

/* check the record size if it is limited */
for (p = records; p; p = p->sister) {
    if (f->record_min > 0) {
        if (p->size < f->record_min) {
            cb_error (_("Record size too small '%s'", p->
name);
        }
    }
    if (f->record_max > 0) {
        if (p->size > f->record_max) {
            cb_error (_("Record size too large '%s' (%d)",
p->name, p->size);
        }
    }
}

/* compute the record size */
if (f->record_min == 0) {
    if (records) {
        f->record_min = records->size;
    } else {
        f->record_min = 0;
    }
}
for (p = records; p; p = p->sister) {
    v = cb_field_variable_size (p);
    if (v && v->offset + v->size * v->occurs_min < f->record_min) {
        f->record_min = v->offset + v->size * v->occurs_min;
    }
    if (p->size < f->record_min) {
        f->record_min = p->size;
    }
    if (p->size > f->record_max) {
        f->record_max = p->size;
    }
}

if (f->same_clause) {
    for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
        if (CB_FILE (CB_VALUE (l))->same_clause == f->
same_clause) {
            if (CB_FILE (CB_VALUE (l))->finalized) {
                if (f->record_max > CB_FILE (CB_VALUE (l))
->record->memory_size) {
                    CB_FILE (CB_VALUE (l))->record->m
emory_size =
                        f->record_max;
                }
                f->record = CB_FILE (CB_VALUE (l))->recor
d;
            }
            for (p = records; p; p = p->sister) {
                p->file = f;
                p->redefines = f->record;
            }
            for (p = f->record->sister; p; p = p->
sister) {
                if (!p->sister) {
                    p->sister = records;
                    break;
                }
            }
        }
    }
}

```

```

f->finalized = 1;
return;
    }
}
}
/* create record */
snprintf (buff, COB_MINI_MAX, "%s_record", f->name);
if (f->record_max == 0) {
    f->record_max = 32;
    f->record_min = 32;
}
if (f->organization == COB_ORG_LINE_SEQUENTIAL) {
    f->record_min = 0;
}
f->record = CB_FIELD (cb_build_implicit_field (cb_build_reference (buff),
f->record_max));
f->record->sister = records;
f->record->count++;
if (f->external) {
    has_external = 1;
    f->record->flag_external = 1;
}

for (p = records; p; p = p->sister) {
    p->file = f;
    p->redefines = f->record;
}
f->finalized = 1;
if (f->linage) {
    snprintf (buff, COB_MINI_MAX, "LC_%s", f->name);
    x = cb_build_field (cb_build_reference (buff));
    CB_FIELD (x)->pic = CB_PICTURE (cb_build_picture ("9(9)"));
    CB_FIELD (x)->usage = CB_USAGE_COMP_5;
    CB_FIELD (x)->values = cb_list_init (cb_zero);
    CB_FIELD (x)->count++;
    cb_validate_field (CB_FIELD (x));
    f->linage_ctr = cb_build_field_reference (CB_FIELD (x), NULL);
    current_program->working_storage =
        cb_field_add (current_program->working_storage, CB_FIELD (x))
;
}
}
}

```

#### 7.15.2.64 void validate\_file ( struct cb\_file \* f, cb\_tree name )

Definition at line 1593 of file tree.c.

```

{
    /* check RECORD/RELATIVE KEY clause */
    switch (f->organization) {
    case COB_ORG_INDEXED:
        if (f->key == NULL) {
            file_error (name, "RECORD KEY");
        }
        break;
    case COB_ORG_RELATIVE:
        if (f->key == NULL && f->access_mode != COB_ACCESS_SEQUENTIAL) {

```

```
        file_error (name, "RELATIVE KEY");
    }
    break;
}
}
```

### 7.15.3 Variable Documentation

#### 7.15.3.1 `cb_tree` `cb_any`

Definition at line 74 of file tree.c.

#### 7.15.3.2 `cb_tree` `cb_error_node`

Definition at line 93 of file tree.c.

#### 7.15.3.3 `cb_tree` `cb_false`

Definition at line 76 of file tree.c.

#### 7.15.3.4 `cb_tree` `cb_high`

Definition at line 82 of file tree.c.

#### 7.15.3.5 `cb_tree` `cb_i[8]`

Definition at line 92 of file tree.c.

#### 7.15.3.6 `cb_tree` `cb_int0`

Definition at line 86 of file tree.c.

#### 7.15.3.7 `cb_tree` `cb_int1`

Definition at line 87 of file tree.c.

#### 7.15.3.8 `cb_tree` `cb_int2`

Definition at line 88 of file tree.c.

#### 7.15.3.9 `cb_tree` `cb_int3`

Definition at line 89 of file tree.c.

**7.15.3.10   cb\_tree cb\_int4**

Definition at line 90 of file tree.c.

**7.15.3.11   cb\_tree cb\_int5**

Definition at line 91 of file tree.c.

**7.15.3.12   cb\_tree cb\_intr\_e**

Definition at line 97 of file tree.c.

**7.15.3.13   cb\_tree cb\_intr\_pi**

Definition at line 96 of file tree.c.

**7.15.3.14   cb\_tree cb\_intr\_whencomp**

Definition at line 95 of file tree.c.

**7.15.3.15   cb\_tree cb\_low**

Definition at line 81 of file tree.c.

**7.15.3.16   cb\_tree cb\_norm\_high**

Definition at line 84 of file tree.c.

**7.15.3.17   cb\_tree cb\_norm\_low**

Definition at line 83 of file tree.c.

**7.15.3.18   cb\_tree cb\_null**

Definition at line 77 of file tree.c.

**7.15.3.19   cb\_tree cb\_one**

Definition at line 79 of file tree.c.



### 7.15.3.20 `cb_tree cb_quote`

Definition at line 85 of file tree.c.

### 7.15.3.21 `cb_tree cb_space`

Definition at line 80 of file tree.c.

### 7.15.3.22 `cb_tree cb_standard_error_handler`

Definition at line 99 of file tree.c.

### 7.15.3.23 `cb_tree cb_true`

Definition at line 75 of file tree.c.

### 7.15.3.24 `cb_tree cb_zero`

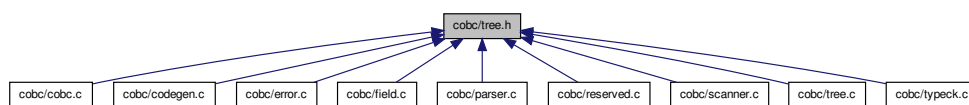
Definition at line 78 of file tree.c.

### 7.15.3.25 `size_t gen_screen_ptr = 0`

Definition at line 101 of file tree.c.

## 7.16 `cobc/tree.h` File Reference

This graph shows which files directly or indirectly include this file:



## Classes

- struct [cb\\_tree\\_common](#)
- struct [cb\\_const](#)
- struct [cb\\_integer](#)
- struct [cb\\_string](#)
- struct [cb\\_alphabet\\_name](#)

- struct [cb\\_class\\_name](#)
- struct [cb\\_locale\\_name](#)
- struct [cb\\_system\\_name](#)
- struct [cb\\_literal](#)
- struct [cb\\_decimal](#)
- struct [cb\\_picture](#)
- struct [cb\\_field](#)
- struct [cb\\_field::cb\\_key](#)
- struct [cb\\_label](#)
- struct [handler\\_struct](#)
- struct [cb\\_alt\\_key](#)
- struct [cb\\_file](#)
- struct [cb\\_word](#)
- struct [cb\\_reference](#)
- struct [cb\\_binary\\_op](#)
- struct [cb\\_funcall](#)
- struct [cb\\_cast](#)
- struct [cb\\_assign](#)
- struct [cb\\_intrinsic\\_table](#)
- struct [cb\\_intrinsic](#)
- struct [cb\\_initialize](#)
- struct [cb\\_search](#)
- struct [cb\\_call](#)
- struct [cb\\_goto](#)
- struct [cb\\_if](#)
- struct [cb\\_perform\\_varying](#)
- struct [cb\\_perform](#)
- struct [cb\\_statement](#)
- struct [cb\\_continue](#)
- struct [cb\\_list](#)
- struct [cb\\_program](#)

## Defines

- #define [YYSTYPE](#) [cb\\_tree](#)
- #define [CB\\_BEFORE](#) [cb\\_int0](#)
- #define [CB\\_AFTER](#) [cb\\_int1](#)
- #define [COB\\_MAX\\_SUBSCRIPTS](#) 16
- #define [CB\\_PREFIX\\_ATTR](#) "a\_"
- #define [CB\\_PREFIX\\_BASE](#) "b\_"
- #define [CB\\_PREFIX\\_CONST](#) "c\_"
- #define [CB\\_PREFIX\\_DECIMAL](#) "d\_"
- #define [CB\\_PREFIX\\_FIELD](#) "f\_"
- #define [CB\\_PREFIX\\_FILE](#) "h\_"
- #define [CB\\_PREFIX\\_KEYS](#) "k\_"
- #define [CB\\_PREFIX\\_LABEL](#) "l\_"

- #define `CB_PREFIX_SEQUENCE` "s\_"
- #define `CB_PROGRAM_TYPE` 0
- #define `CB_FUNCTION_TYPE` 1
- #define `CB_TREE(x)` ((struct `cb_tree_common` \*) (x))
- #define `CB_TREE_TAG(x)` (CB\_TREE (x)->tag)
- #define `CB_TREE_CLASS(x)` `cb_tree_class` (CB\_TREE (x))
- #define `CB_TREE_CATEGORY(x)` `cb_tree_category` (CB\_TREE (x))
- #define `CB_TREE_CAST(tg, ty, x)` ((ty \*) (x))
- #define `CB_CONST(x)` (CB\_TREE\_CAST (CB\_TAG\_CONST, struct `cb_const`, x))
- #define `CB_CONST_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_CONST)
- #define `CB_INTEGER(x)` (CB\_TREE\_CAST (CB\_TAG\_INTEGER, struct `cb_integer`, x))
- #define `CB_INTEGER_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_INTEGER)
- #define `CB_STRING(x)` (CB\_TREE\_CAST (CB\_TAG\_STRING, struct `cb_string`, x))
- #define `CB_STRING_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_STRING)
- #define `cb_build_string0(str)` `cb_build_string` (str, strlen ((char \*)str))
- #define `CB_ALPHABET_NAME(x)` (CB\_TREE\_CAST (CB\_TAG\_ALPHABET\_NAME, struct `cb_alphabet_name`, x))
- #define `CB_ALPHABET_NAME_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_ALPHABET\_NAME)
- #define `CB_CLASS_NAME(x)` (CB\_TREE\_CAST (CB\_TAG\_CLASS\_NAME, struct `cb_class_name`, x))
- #define `CB_CLASS_NAME_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_CLASS\_NAME)
- #define `CB_LOCALE_NAME(x)` (CB\_TREE\_CAST (CB\_TAG\_LOCALE\_NAME, struct `cb_locale_name`, x))
- #define `CB_LOCALE_NAME_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_LOCALE\_NAME)
- #define `CB_SYSTEM_NAME(x)` (CB\_TREE\_CAST (CB\_TAG\_SYSTEM\_NAME, struct `cb_system_name`, x))
- #define `CB_SYSTEM_NAME_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_SYSTEM\_NAME)
- #define `CB_LITERAL(x)` (CB\_TREE\_CAST (CB\_TAG\_LITERAL, struct `cb_literal`, x))
- #define `CB_LITERAL_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_LITERAL)
- #define `CB_NUMERIC_LITERAL_P(x)` (CB\_LITERAL\_P (x) && CB\_TREE\_CATEGORY (x) == CB\_CATEGORY\_NUMERIC)
- #define `CB_DECIMAL(x)` (CB\_TREE\_CAST (CB\_TAG\_DECIMAL, struct `cb_decimal`, x))
- #define `CB_DECIMAL_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_DECIMAL)
- #define `CB_PICTURE(x)` (CB\_TREE\_CAST (CB\_TAG\_PICTURE, struct `cb_picture`, x))
- #define `CB_PICTURE_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_PICTURE)
- #define `CB_FIELD(x)` (CB\_TREE\_CAST (CB\_TAG\_FIELD, struct `cb_field`, x))
- #define `CB_FIELD_P(x)` (CB\_TREE\_TAG (x) == CB\_TAG\_FIELD)
- #define `CB_REF_OR_FIELD_P(x)` ((CB\_FIELD\_P (x) || CB\_REFERENCE\_P (x)))

- #define `CB_INDEX_P(x)`
- #define `CB_LABEL(x)` (`CB_TREE_CAST (CB_TAG_LABEL, struct cb_label, x)`)
- #define `CB_LABEL_P(x)` (`CB_TREE_TAG (x) == CB_TAG_LABEL`)
- #define `CB_FILE(x)` (`CB_TREE_CAST (CB_TAG_FILE, struct cb_file, x)`)
- #define `CB_FILE_P(x)` (`CB_TREE_TAG (x) == CB_TAG_FILE`)
- #define `CB_WORD_HASH_SIZE` 133
- #define `CB_REFERENCE(x)` (`CB_TREE_CAST (CB_TAG_REFERENCE, struct cb_reference, x)`)
- #define `CB_REFERENCE_P(x)` (`CB_TREE_TAG (x) == CB_TAG_REFERENCE`)
- #define `CB_NAME(x)` (`CB_REFERENCE (x)->word->name`)
- #define `CB_BINARY_OP(x)` (`CB_TREE_CAST (CB_TAG_BINARY_OP, struct cb_binary_op, x)`)
- #define `CB_BINARY_OP_P(x)` (`CB_TREE_TAG (x) == CB_TAG_BINARY_OP`)
- #define `cb_build_parenthesis(x)` `cb_build_binary_op (x, '@', NULL)`
- #define `cb_build_negation(x)` `cb_build_binary_op (x, '!', NULL)`
- #define `CB_FUNCALL(x)` (`CB_TREE_CAST (CB_TAG_FUNCALL, struct cb_funcall, x)`)
- #define `CB_FUNCALL_P(x)` (`CB_TREE_TAG (x) == CB_TAG_FUNCALL`)
- #define `cb_build_funcall_0(f)` `cb_build_funcall(f, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL)`
- #define `cb_build_funcall_1(f, a1)` `cb_build_funcall(f, 1, a1, NULL, NULL, NULL, NULL, NULL, NULL)`
- #define `cb_build_funcall_2(f, a1, a2)` `cb_build_funcall(f, 2, a1, a2, NULL, NULL, NULL, NULL, NULL)`
- #define `cb_build_funcall_3(f, a1, a2, a3)` `cb_build_funcall(f, 3, a1, a2, a3, NULL, NULL, NULL, NULL)`
- #define `cb_build_funcall_4(f, a1, a2, a3, a4)` `cb_build_funcall(f, 4, a1, a2, a3, a4, NULL, NULL, NULL)`
- #define `cb_build_funcall_5(f, a1, a2, a3, a4, a5)` `cb_build_funcall(f, 5, a1, a2, a3, a4, a5, NULL, NULL)`
- #define `cb_build_funcall_6(f, a1, a2, a3, a4, a5, a6)` `cb_build_funcall(f, 6, a1, a2, a3, a4, a5, a6, NULL)`
- #define `cb_build_funcall_7(f, a1, a2, a3, a4, a5, a6, a7)` `cb_build_funcall(f, 7, a1, a2, a3, a4, a5, a6, a7)`
- #define `CB_CAST(x)` (`CB_TREE_CAST (CB_TAG_CAST, struct cb_cast, x)`)
- #define `CB_CAST_P(x)` (`CB_TREE_TAG (x) == CB_TAG_CAST`)
- #define `cb_build_cast_integer(x)` `cb_build_cast (CB_CAST_INTEGER, x)`
- #define `cb_build_cast_address(x)` `cb_build_cast (CB_CAST_ADDRESS, x)`
- #define `cb_build_cast_addr_of_addr(x)` `cb_build_cast (CB_CAST_ADDR_OF_ADDR, x)`
- #define `cb_build_cast_length(x)` `cb_build_cast (CB_CAST_LENGTH, x)`
- #define `cb_build_cast_ppointer(x)` `cb_build_cast (CB_CAST_PROGRAM_POINTER, x)`
- #define `CB_ASSIGN(x)` (`CB_TREE_CAST (CB_TAG_ASSIGN, struct cb_assign, x)`)
- #define `CB_ASSIGN_P(x)` (`CB_TREE_TAG (x) == CB_TAG_ASSIGN`)
- #define `CB_INTRINSIC(x)` (`CB_TREE_CAST (CB_TAG_INTRINSIC, struct cb_intrinsic, x)`)

- #define `CB_INTRINSIC_P(x)` (`CB_TREE_TAG (x) == CB_TAG_INTRINSIC`)
- #define `CB_INITIALIZE(x)` (`CB_TREE_CAST (CB_TAG_INITIALIZE, struct cb_initialize, x)`)
- #define `CB_INITIALIZE_P(x)` (`CB_TREE_TAG (x) == CB_TAG_INITIALIZE`)
- #define `CB_SEARCH(x)` (`CB_TREE_CAST (CB_TAG_SEARCH, struct cb_search, x)`)
- #define `CB_SEARCH_P(x)` (`CB_TREE_TAG (x) == CB_TAG_SEARCH`)
- #define `CB_CALL_BY_REFERENCE` 1
- #define `CB_CALL_BY_CONTENT` 2
- #define `CB_CALL_BY_VALUE` 3
- #define `CB_CALL(x)` (`CB_TREE_CAST (CB_TAG_CALL, struct cb_call, x)`)
- #define `CB_CALL_P(x)` (`CB_TREE_TAG (x) == CB_TAG_CALL`)
- #define `CB_GOTO(x)` (`CB_TREE_CAST (CB_TAG_GOTO, struct cb_goto, x)`)
- #define `CB_GOTO_P(x)` (`CB_TREE_TAG (x) == CB_TAG_GOTO`)
- #define `CB_IF(x)` (`CB_TREE_CAST (CB_TAG_IF, struct cb_if, x)`)
- #define `CB_IF_P(x)` (`CB_TREE_TAG (x) == CB_TAG_IF`)
- #define `CB_PERFORM_VARYING(x)` (`CB_TREE_CAST (CB_TAG_PERFORM_VARYING, struct cb_perform_varying, x)`)
- #define `CB_PERFORM(x)` (`CB_TREE_CAST (CB_TAG_PERFORM, struct cb_perform, x)`)
- #define `CB_PERFORM_P(x)` (`CB_TREE_TAG (x) == CB_TAG_PERFORM`)
- #define `CB_STATEMENT(x)` (`CB_TREE_CAST (CB_TAG_STATEMENT, struct cb_statement, x)`)
- #define `CB_STATEMENT_P(x)` (`CB_TREE_TAG (x) == CB_TAG_STATEMENT`)
- #define `CB_CONTINUE(x)` (`CB_TREE_CAST (CB_TAG_CONTINUE, struct cb_continue, x)`)
- #define `CB_CONTINUE_P(x)` (`CB_TREE_TAG (x) == CB_TAG_CONTINUE`)
- #define `CB_LIST(x)` (`CB_TREE_CAST (CB_TAG_LIST, struct cb_list, x)`)
- #define `CB_LIST_P(x)` (`CB_TREE_TAG (x) == CB_TAG_LIST`)
- #define `CB_PURPOSE(x)` (`CB_LIST (x)->purpose`)
- #define `CB_VALUE(x)` (`CB_LIST (x)->value`)
- #define `CB_CHAIN(x)` (`CB_LIST (x)->chain`)
- #define `CB_SIZES(x)` (`CB_LIST (x)->sizes`)
- #define `CB_PURPOSE_INT(x)` (`CB_INTEGER (CB_PURPOSE (x))->val`)
- #define `CB_SIZE_AUTO` 0
- #define `CB_SIZE_1` 1
- #define `CB_SIZE_2` 2
- #define `CB_SIZE_4` 3
- #define `CB_SIZE_8` 4
- #define `CB_SIZE_UNSIGNED` 8
- #define `CB_SIZES_INT(x)` (`((CB_LIST (x)->sizes) & 0x07)`)
- #define `CB_SIZES_INT_UNSIGNED(x)` (`((CB_LIST (x)->sizes) & CB_SIZE_UNSIGNED)`)
- #define `cb_list_init(x)` `cb_build_list (NULL, x, NULL)`
- #define `cb_cons(x, l)` `cb_build_list (NULL, x, l)`
- #define `CB_PAIR_P(x)` (`CB_LIST_P (x) && CB_PAIR_X (x)`)
- #define `CB_PAIR_X(x)` `CB_PURPOSE (x)`
- #define `CB_PAIR_Y(x)` `CB_VALUE (x)`
- #define `cb_build_pair(x, y)` `cb_build_list (x, y, NULL)`

## Typedefs

- typedef struct `cb_tree_common` \* `cb_tree`

## Enumerations

- enum `cb_tag` {  
`CB_TAG_CONST`, `CB_TAG_INTEGER`, `CB_TAG_STRING`, `CB_TAG_ALPHABET_`  
`NAME`,  
`CB_TAG_CLASS_NAME`, `CB_TAG_LOCALE_NAME`, `CB_TAG_SYSTEM_NAME`,  
`CB_TAG_LITERAL`,  
`CB_TAG_DECIMAL`, `CB_TAG_FIELD`, `CB_TAG_FILE`, `CB_TAG_REFERENCE`,  
`CB_TAG_BINARY_OP`, `CB_TAG_FUNCALL`, `CB_TAG_CAST`, `CB_TAG_INTRINSIC`,  
`CB_TAG_LABEL`, `CB_TAG_ASSIGN`, `CB_TAG_INITIALIZE`, `CB_TAG_SEARCH`,  
`CB_TAG_CALL`, `CB_TAG_GOTO`, `CB_TAG_IF`, `CB_TAG_PERFORM`,  
`CB_TAG_STATEMENT`, `CB_TAG_CONTINUE`, `CB_TAG_PERFORM_VARYING`,  
`CB_TAG_PICTURE`,  
`CB_TAG_LIST` }
- enum `cb_alphabet_name_type` {  
`CB_ALPHABET_NATIVE`, `CB_ALPHABET_STANDARD_1`, `CB_ALPHABET_STANDARD_`  
`2`, `CB_ALPHABET_EBCDIC`,  
`CB_ALPHABET_CUSTOM` }
- enum `cb_system_name_category` {  
`CB_CALL_CONVENTION_NAME`, `CB_CODE_NAME`, `CB_COMPUTER_NAME`,  
`CB_DEVICE_NAME`,  
`CB_ENTRY_CONVENTION_NAME`, `CB_EXTERNAL_LOCALE_NAME`, `CB_FEATURE_`  
`NAME`, `CB_LIBRARY_NAME`,  
`CB_SWITCH_NAME`, `CB_TEXT_NAME` }
- enum `cb_device_name` { `CB_DEVICE_SYSIN`, `CB_DEVICE_SYSOUT`, `CB_DEVICE_`  
`SYSERR`, `CB_DEVICE_CONSOLE` }
- enum `cb_feature_name` {  
`CB_FEATURE_FORMFEED`, `CB_FEATURE_C01`, `CB_FEATURE_C02`, `CB_FEATURE_`  
`C03`,  
`CB_FEATURE_C04`, `CB_FEATURE_C05`, `CB_FEATURE_C06`, `CB_FEATURE_`  
`C07`,  
`CB_FEATURE_C08`, `CB_FEATURE_C09`, `CB_FEATURE_C10`, `CB_FEATURE_`  
`C11`,  
`CB_FEATURE_C12` }
- enum `cb_switch_name` {  
`CB_SWITCH_1`, `CB_SWITCH_2`, `CB_SWITCH_3`, `CB_SWITCH_4`,  
`CB_SWITCH_5`, `CB_SWITCH_6`, `CB_SWITCH_7`, `CB_SWITCH_8` }

- enum `cb_class` {  
    `CB_CLASS_UNKNOWN`, `CB_CLASS_ALPHABETIC`, `CB_CLASS_ALPHANUMERIC`,  
    `CB_CLASS_BOOLEAN`,  
  
    `CB_CLASS_INDEX`, `CB_CLASS_NATIONAL`, `CB_CLASS_NUMERIC`, `CB_CLASS_-`  
    `OBJECT`,  
  
    `CB_CLASS_POINTER` }
- enum `cb_category` {  
    `CB_CATEGORY_UNKNOWN`, `CB_CATEGORY_ALPHABETIC`, `CB_CATEGORY_-`  
    `ALPHANUMERIC`, `CB_CATEGORY_ALPHANUMERIC_EDITED`,  
  
    `CB_CATEGORY_BOOLEAN`, `CB_CATEGORY_INDEX`, `CB_CATEGORY_NATIONAL`,  
    `CB_CATEGORY_NATIONAL_EDITED`,  
  
    `CB_CATEGORY_NUMERIC`, `CB_CATEGORY_NUMERIC_EDITED`, `CB_CATEGORY_-`  
    `OBJECT_REFERENCE`, `CB_CATEGORY_DATA_POINTER`,  
  
    `CB_CATEGORY_PROGRAM_POINTER` }
- enum `cb_storage` {  
    `CB_STORAGE_CONSTANT`, `CB_STORAGE_FILE`, `CB_STORAGE_WORKING`,  
    `CB_STORAGE_LOCAL`,  
  
    `CB_STORAGE_LINKAGE`, `CB_STORAGE_SCREEN`, `CB_STORAGE_REPORT`,  
    `CB_STORAGE_COMMUNICATION` }
- enum `cb_usage` {  
    `CB_USAGE_BINARY`, `CB_USAGE_BIT`, `CB_USAGE_COMP_5`, `CB_USAGE_-`  
    `COMP_X`,  
  
    `CB_USAGE_DISPLAY`, `CB_USAGE_FLOAT`, `CB_USAGE_DOUBLE`, `CB_USAGE_-`  
    `INDEX`,  
  
    `CB_USAGE_NATIONAL`, `CB_USAGE_OBJECT`, `CB_USAGE_PACKED`, `CB_USAGE_-`  
    `POINTER`,  
  
    `CB_USAGE_PROGRAM`, `CB_USAGE_LENGTH`, `CB_USAGE_PROGRAM_POINTER`,  
    `CB_USAGE_UNSIGNED_CHAR`,  
  
    `CB_USAGE_SIGNED_CHAR`, `CB_USAGE_UNSIGNED_SHORT`, `CB_USAGE_-`  
    `SIGNED_SHORT`, `CB_USAGE_UNSIGNED_INT`,  
  
    `CB_USAGE_SIGNED_INT`, `CB_USAGE_UNSIGNED_LONG`, `CB_USAGE_SIGNED_-`  
    `LONG` }
- enum `cb_operand_type` { `CB_SENDING_OPERAND`, `CB_RECEIVING_OPERAND`  
    }
- enum `cb_cast_type` {  
    `CB_CAST_INTEGER`, `CB_CAST_ADDRESS`, `CB_CAST_ADDR_OF_ADDR`, `CB_-`  
    `CAST_LENGTH`,  
  
    `CB_CAST_PROGRAM_POINTER` }
- enum `cb_intr_enum` {  
    `CB_INTR_ABS = 1`, `CB_INTR_ACOS`, `CB_INTR_ANNUITY`, `CB_INTR_ASIN`,  
  
    `CB_INTR_ATAN`, `CB_INTR_BOOLEAN_OF_INTEGER`, `CB_INTR_BYTE_LENGTH`,  
    `CB_INTR_CHAR`,

```

CB_INTR_CHAR_NATIONAL, CB_INTR_COMBINED_DATETIME, CB_INTR_CONCATENATE,
CB_INTR_COS,
CB_INTR_CURRENT_DATE, CB_INTR_DATE_OF_INTEGER, CB_INTR_DATE_-
TO_YYYYMMDD, CB_INTR_DAY_OF_INTEGER,
CB_INTR_DAY_TO_YYYYDDD, CB_INTR_DISPLAY_OF, CB_INTR_E, CB_INTR_-
EXCEPTION_FILE,
CB_INTR_EXCEPTION_FILE_N, CB_INTR_EXCEPTION_LOCATION, CB_INTR_-
EXCEPTION_LOCATION_N, CB_INTR_EXCEPTION_STATEMENT,
CB_INTR_EXCEPTION_STATUS, CB_INTR_EXP, CB_INTR_EXP10, CB_INTR_-
FACTORIAL,
CB_INTR_FRACTION_PART, CB_INTR_HIGHEST_ALGEBRAIC, CB_INTR_INTEGER,
CB_INTR_INTEGER_OF_BOOLEAN,
CB_INTR_INTEGER_OF_DATE, CB_INTR_INTEGER_OF_DAY, CB_INTR_INTEGER_-
PART, CB_INTR_LENGTH,
CB_INTR_LOCALE_COMPARE, CB_INTR_LOCALE_DATE, CB_INTR_LOCALE_-
TIME, CB_INTR_LOCALE_TIME_FROM_SECS,
CB_INTR_LOG, CB_INTR_LOG10, CB_INTR_LOWER_CASE, CB_INTR_LOWEST_-
ALGEBRAIC,
CB_INTR_MAX, CB_INTR_MEAN, CB_INTR_MEDIAN, CB_INTR_MIDRANGE,
CB_INTR_MIN, CB_INTR_MOD, CB_INTR_NATIONAL_OF, CB_INTR_NUMVAL,
CB_INTR_NUMVAL_C, CB_INTR_NUMVAL_F, CB_INTR_ORD, CB_INTR_ORD_-
MAX,
CB_INTR_ORD_MIN, CB_INTR_PI, CB_INTR_PRESENT_VALUE, CB_INTR_-
RANDOM,
CB_INTR_RANGE, CB_INTR_REM, CB_INTR_REVERSE, CB_INTR_SECONDS_-
FROM_FORMATTED_TIME,
CB_INTR_SECONDS_PAST_MIDNIGHT, CB_INTR_SIGN, CB_INTR_SIN, CB_-
INTR_SQRT,
CB_INTR_STANDARD_COMPARE, CB_INTR_STANDARD_DEVIATION, CB_INTR_-
STORED_CHAR_LENGTH, CB_INTR_SUBSTITUTE,
CB_INTR_SUBSTITUTE_CASE, CB_INTR_SUM, CB_INTR_TAN, CB_INTR_-
TEST_DATE_YYYYMMDD,
CB_INTR_TEST_DAY_YYYYDDD, CB_INTR_TEST_NUMVAL, CB_INTR_TEST_-
NUMVAL_C, CB_INTR_TEST_NUMVAL_F,
CB_INTR_TRIM, CB_INTR_UPPER_CASE, CB_INTR_VARIANCE, CB_INTR_-
WHEN_COMPILED,
CB_INTR_YEAR_TO_YYYY }
• enum cb_perform_type {
CB_PERFORM_EXIT, CB_PERFORM_ONCE, CB_PERFORM_TIMES, CB_PERFORM_-
UNTIL,
CB_PERFORM_FOREVER }

```



## Functions

- char \* [cb\\_name](#) (cb\_tree x)
- enum [cb\\_class](#) [cb\\_tree\\_class](#) (cb\_tree x)
- enum [cb\\_category](#) [cb\\_tree\\_category](#) (cb\_tree x)
- int [cb\\_tree\\_type](#) (cb\_tree x)
- int [cb\\_fits\\_int](#) (cb\_tree x)
- int [cb\\_fits\\_long\\_long](#) (cb\_tree x)
- int [cb\\_get\\_int](#) (cb\_tree x)
- long long [cb\\_get\\_long\\_long](#) (cb\_tree x)
- void [cb\\_init\\_constants](#) (void)
- [cb\\_tree](#) [cb\\_int](#) (int n)
- [cb\\_tree](#) [cb\\_build\\_string](#) (const unsigned char \*data, size\_t size)
- [cb\\_tree](#) [cb\\_build\\_alphabet\\_name](#) (cb\_tree name, enum [cb\\_alphabet\\_name\\_type](#) type)
- [cb\\_tree](#) [cb\\_build\\_class\\_name](#) (cb\_tree name, cb\_tree list)
- [cb\\_tree](#) [cb\\_build\\_locale\\_name](#) (cb\_tree name, cb\_tree list)
- [cb\\_tree](#) [cb\\_build\\_system\\_name](#) (enum [cb\\_system\\_name\\_category](#) category, int token)
- [cb\\_tree](#) [cb\\_build\\_numeric\\_literal](#) (int sign, const unsigned char \*data, int scale)
- [cb\\_tree](#) [cb\\_build\\_alphanumeric\\_literal](#) (const unsigned char \*data, size\_t size)
- [cb\\_tree](#) [cb\\_concat\\_literals](#) (cb\_tree x1, cb\_tree x2)
- [cb\\_tree](#) [cb\\_build\\_decimal](#) (int id)
- [cb\\_tree](#) [cb\\_build\\_picture](#) (const char \*str)
- [cb\\_tree](#) [cb\\_build\\_field](#) (cb\_tree name)
- [cb\\_tree](#) [cb\\_build\\_implicit\\_field](#) (cb\_tree name, int len)
- [cb\\_tree](#) [cb\\_build\\_constant](#) (cb\_tree name, cb\_tree value)
- struct [cb\\_field](#) \* [cb\\_field](#) (cb\_tree x)
- struct [cb\\_field](#) \* [cb\\_field\\_add](#) (struct [cb\\_field](#) \*f, struct [cb\\_field](#) \*p)
- int [cb\\_field\\_size](#) (cb\_tree x)
- struct [cb\\_field](#) \* [cb\\_field\\_founder](#) (struct [cb\\_field](#) \*f)
- struct [cb\\_field](#) \* [cb\\_field\\_variable\\_size](#) (struct [cb\\_field](#) \*f)
- struct [cb\\_field](#) \* [cb\\_field\\_variable\\_address](#) (struct [cb\\_field](#) \*f)
- int [cb\\_field\\_subordinate](#) (struct [cb\\_field](#) \*p, struct [cb\\_field](#) \*f)
- [cb\\_tree](#) [cb\\_build\\_label](#) (cb\_tree name, struct [cb\\_label](#) \*section)
- struct [cb\\_file](#) \* [build\\_file](#) (cb\_tree name)
- void [validate\\_file](#) (struct [cb\\_file](#) \*f, cb\_tree name)
- void [finalize\\_file](#) (struct [cb\\_file](#) \*f, struct [cb\\_field](#) \*records)
- [cb\\_tree](#) [cb\\_build\\_filler](#) (void)
- [cb\\_tree](#) [cb\\_build\\_reference](#) (const char \*name)
- [cb\\_tree](#) [cb\\_build\\_field\\_reference](#) (struct [cb\\_field](#) \*f, cb\_tree ref)
- const char \* [cb\\_define](#) (cb\_tree name, cb\_tree val)
- void [cb\\_define\\_system\\_name](#) (const char \*name)
- [cb\\_tree](#) [cb\\_ref](#) (cb\_tree x)
- [cb\\_tree](#) [cb\\_build\\_binary\\_op](#) (cb\_tree x, int op, cb\_tree y)
- [cb\\_tree](#) [cb\\_build\\_binary\\_list](#) (cb\_tree l, int op)

- `cb_tree cb_build_funcall` (const char \*name, int argc, `cb_tree` a1, `cb_tree` a2, `cb_tree` a3, `cb_tree` a4, `cb_tree` a5, `cb_tree` a6, `cb_tree` a7)
- `cb_tree cb_build_cast` (enum `cb_cast_type` type, `cb_tree` val)
- `cb_tree cb_build_assign` (`cb_tree` var, `cb_tree` val)
- struct `cb_intrinsic_table` \* `lookup_intrinsic` (const char \*name, const int checkres)
- `cb_tree cb_build_intrinsic` (`cb_tree` name, `cb_tree` args, `cb_tree` refmod)
- `cb_tree cb_build_any_intrinsic` (`cb_tree` args)
- `cb_tree cb_build_initialize` (`cb_tree` var, `cb_tree` val, `cb_tree` rep, `cb_tree` def, int flag)
- `cb_tree cb_build_search` (int flag\_all, `cb_tree` table, `cb_tree` var, `cb_tree` end\_stmt, `cb_tree` whens)
- `cb_tree cb_build_call` (`cb_tree` name, `cb_tree` args, `cb_tree` stmt1, `cb_tree` stmt2, `cb_tree` returning, int is\_system\_call)
- `cb_tree cb_build_goto` (`cb_tree` target, `cb_tree` depending)
- `cb_tree cb_build_if` (`cb_tree` test, `cb_tree` stmt1, `cb_tree` stmt2)
- `cb_tree cb_build_perform` (int type)
- `cb_tree cb_build_perform_varying` (`cb_tree` name, `cb_tree` from, `cb_tree` step, `cb_tree` until)
- struct `cb_statement` \* `cb_build_statement` (const char \*name)
- `cb_tree cb_build_continue` (void)
- `cb_tree cb_build_list` (`cb_tree` purpose, `cb_tree` value, `cb_tree` rest)
- `cb_tree cb_list_add` (`cb_tree` l, `cb_tree` x)
- `cb_tree cb_list_append` (`cb_tree` l1, `cb_tree` l2)
- `cb_tree cb_list_reverse` (`cb_tree` l)
- int `cb_list_length` (`cb_tree` l)
- struct `cb_program` \* `cb_build_program` (struct `cb_program` \*last\_program, int nest\_level)
- `cb_tree lookup_system_name` (const char \*name)
- int `lookup_reserved_word` (const char \*name)
- void `cb_list_reserved` (void)
- void `cb_list_intrinsics` (void)
- void `cb_list_mnemonics` (void)
- void `cb_init_reserved` (void)
- void `cb_list_map` (`cb_tree`(\*func)(`cb_tree` x), `cb_tree` l)
- void `cb_warning_x` (`cb_tree` x, const char \*fmt,...)
- void `cb_error_x` (`cb_tree` x, const char \*fmt,...)
- char \* `check_filler_name` (char \*name)
- void `redefinition_error` (`cb_tree` x)
- void `redefinition_warning` (`cb_tree` x, `cb_tree` y)
- void `undefined_error` (`cb_tree` x)
- void `ambiguous_error` (`cb_tree` x)
- void `group_error` (`cb_tree` x, const char \*clause)
- void `level_redundant_error` (`cb_tree` x, const char \*clause)
- void `level_require_error` (`cb_tree` x, const char \*clause)
- void `level_except_error` (`cb_tree` x, const char \*clause)
- struct `cb_literal` \* `build_literal` (enum `cb_category` category, const unsigned char \*data, size\_t size)

- `int cb_get_level (cb_tree x)`
- `cb_tree cb_build_field_tree (cb_tree level, cb_tree name, struct cb_field *last_field, enum cb_storage storage, struct cb_file *fn)`
- `struct cb_field * cb_resolve_redefines (struct cb_field *field, cb_tree redefines)`
- `void cb_validate_field (struct cb_field *p)`
- `void cb_validate_88_item (struct cb_field *p)`
- `struct cb_field * cb_validate_78_item (struct cb_field *p)`
- `void cb_clear_real_field (void)`
- `cb_tree cb_check_numeric_value (cb_tree x)`
- `void cb_build_registers (void)`
- `char * cb_encode_program_id (const char *name)`
- `const char * cb_build_program_id (cb_tree name, cb_tree alt_name)`
- `void cb_define_switch_name (cb_tree name, cb_tree sname, cb_tree flag, cb_tree ref)`
- `cb_tree cb_build_section_name (cb_tree name, int sect_or_para)`
- `cb_tree cb_build_assignment_name (struct cb_file *curfile, cb_tree name)`
- `cb_tree cb_build_index (cb_tree name, cb_tree values, int indexed_by, struct cb_field *qual)`
- `cb_tree cb_build_identifier (cb_tree x)`
- `cb_tree cb_build_length (cb_tree x)`
- `cb_tree cb_build_const_length (cb_tree x)`
- `cb_tree cb_build_address (cb_tree x)`
- `cb_tree cb_build_ppointer (cb_tree x)`
- `void cb_validate_program_environment (struct cb_program *prog)`
- `void cb_validate_program_data (struct cb_program *prog)`
- `void cb_validate_program_body (struct cb_program *prog)`
- `cb_tree cb_build_expr (cb_tree list)`
- `cb_tree cb_build_cond (cb_tree x)`
- `void cb_emit_arithmetic (cb_tree vars, int op, cb_tree val)`
- `cb_tree cb_build_add (cb_tree v, cb_tree n, cb_tree round_opt)`
- `cb_tree cb_build_sub (cb_tree v, cb_tree n, cb_tree round_opt)`
- `void cb_emit_corresponding (cb_tree(*func)(cb_tree f1, cb_tree f2, cb_tree f3), cb_tree x1, cb_tree x2, cb_tree opt)`
- `void cb_emit_move_corresponding (cb_tree x1, cb_tree x2)`
- `void cb_emit_accept (cb_tree var, cb_tree pos, cb_tree fgc, cb_tree bgc, cb_tree scroll, int dispattrs)`
- `void cb_emit_accept_line_or_col (cb_tree var, const int l_or_c)`
- `void cb_emit_accept_date (cb_tree var)`
- `void cb_emit_accept_date_yyyymmdd (cb_tree var)`
- `void cb_emit_accept_day (cb_tree var)`
- `void cb_emit_accept_day_yyyydd (cb_tree var)`
- `void cb_emit_accept_day_of_week (cb_tree var)`
- `void cb_emit_accept_time (cb_tree var)`
- `void cb_emit_accept_command_line (cb_tree var)`
- `void cb_emit_get_environment (cb_tree envvar, cb_tree envval)`
- `void cb_emit_accept_environment (cb_tree var)`
- `void cb_emit_accept_mnemonic (cb_tree var, cb_tree mnemonic)`

- void `cb_emit_accept_name` (`cb_tree` var, `cb_tree` name)
- void `cb_emit_accept_arg_number` (`cb_tree` var)
- void `cb_emit_accept_arg_value` (`cb_tree` var)
- void `cb_emit_allocate` (`cb_tree` target1, `cb_tree` target2, `cb_tree` size, `cb_tree` initialize)
- void `cb_emit_free` (`cb_tree` vars)
- void `cb_emit_call` (`cb_tree` prog, `cb_tree` using, `cb_tree` returning, `cb_tree` on\_exception, `cb_tree` not\_on\_exception)
- void `cb_emit_cancel` (`cb_tree` prog)
- void `cb_emit_close` (`cb_tree` file, `cb_tree` opt)
- void `cb_emit_commit` (void)
- void `cb_emit_continue` (void)
- void `cb_emit_delete` (`cb_tree` file)
- void `cb_emit_display` (`cb_tree` values, `cb_tree` upon, `cb_tree` no\_adv, `cb_tree` pos, `cb_tree` fg, `cb_tree` bg, `cb_tree` scroll, int dispattrs)
- `cb_tree` `cb_build_display_upon` (`cb_tree` x)
- `cb_tree` `cb_build_display_upon_direct` (`cb_tree` x)
- void `cb_emit_env_name` (`cb_tree` value)
- void `cb_emit_env_value` (`cb_tree` value)
- void `cb_emit_arg_number` (`cb_tree` value)
- void `cb_emit_command_line` (`cb_tree` value)
- void `cb_emit_divide` (`cb_tree` dividend, `cb_tree` divisor, `cb_tree` quotient, `cb_tree` remainder)
- void `cb_emit_evaluate` (`cb_tree` subject\_list, `cb_tree` case\_list)
- void `cb_emit_goto` (`cb_tree` target, `cb_tree` depending)
- void `cb_emit_exit` (size\_t goback)
- void `cb_emit_if` (`cb_tree` cond, `cb_tree` stmt1, `cb_tree` stmt2)
- void `cb_emit_initialize` (`cb_tree` vars, `cb_tree` fillinit, `cb_tree` value, `cb_tree` replacing, `cb_tree` def)
- void `cb_emit_inspect` (`cb_tree` var, `cb_tree` body, `cb_tree` replacing, int replconv)
- void `cb_init_tarrying` (void)
- `cb_tree` `cb_build_tarrying_data` (`cb_tree` x)
- `cb_tree` `cb_build_tarrying_characters` (`cb_tree` l)
- `cb_tree` `cb_build_tarrying_all` (void)
- `cb_tree` `cb_build_tarrying_leading` (void)
- `cb_tree` `cb_build_tarrying_trailing` (void)
- `cb_tree` `cb_build_tarrying_value` (`cb_tree` x, `cb_tree` l)
- `cb_tree` `cb_build_replacing_characters` (`cb_tree` x, `cb_tree` l)
- `cb_tree` `cb_build_replacing_all` (`cb_tree` x, `cb_tree` y, `cb_tree` l)
- `cb_tree` `cb_build_replacing_leading` (`cb_tree` x, `cb_tree` y, `cb_tree` l)
- `cb_tree` `cb_build_replacing_first` (`cb_tree` x, `cb_tree` y, `cb_tree` l)
- `cb_tree` `cb_build_replacing_trailing` (`cb_tree` x, `cb_tree` y, `cb_tree` l)
- `cb_tree` `cb_build_converting` (`cb_tree` x, `cb_tree` y, `cb_tree` l)
- `cb_tree` `cb_build_inspect_region_start` (void)
- `cb_tree` `cb_build_inspect_region` (`cb_tree` l, `cb_tree` pos, `cb_tree` x)
- int `validate_move` (`cb_tree` src, `cb_tree` dst, size\_t is\_value)
- `cb_tree` `cb_build_move` (`cb_tree` src, `cb_tree` dst)

- void `cb_emit_move` (`cb_tree` src, `cb_tree` dsts)
- void `cb_emit_open` (`cb_tree` file, `cb_tree` mode, `cb_tree` sharing)
- void `cb_emit_perform` (`cb_tree` perform, `cb_tree` body)
- `cb_tree` `cb_build_perform_once` (`cb_tree` body)
- `cb_tree` `cb_build_perform_times` (`cb_tree` count)
- `cb_tree` `cb_build_perform_until` (`cb_tree` condition, `cb_tree` varying)
- `cb_tree` `cb_build_perform_forever` (`cb_tree` body)
- `cb_tree` `cb_build_perform_exit` (struct `cb_label` \*label)
- void `cb_emit_read` (`cb_tree` ref, `cb_tree` next, `cb_tree` into, `cb_tree` key, `cb_tree` lock\_opts)
- void `cb_emit_rewrite` (`cb_tree` record, `cb_tree` from, `cb_tree` lockopt)
- void `cb_emit_release` (`cb_tree` ref, `cb_tree` from)
- void `cb_emit_return` (`cb_tree` ref, `cb_tree` into)
- void `cb_emit_rollback` (void)
- void `cb_emit_search` (`cb_tree` table, `cb_tree` varying, `cb_tree` at\_end, `cb_tree` whens)
- void `cb_emit_search_all` (`cb_tree` table, `cb_tree` at\_end, `cb_tree` when, `cb_tree` stmts)
- void `cb_emit_setenv` (`cb_tree` x, `cb_tree` y)
- void `cb_emit_set_to` (`cb_tree` l, `cb_tree` x)
- void `cb_emit_set_up_down` (`cb_tree` l, `cb_tree` flag, `cb_tree` x)
- void `cb_emit_set_on_off` (`cb_tree` l, `cb_tree` flag)
- void `cb_emit_set_true` (`cb_tree` l)
- void `cb_emit_set_false` (`cb_tree` l)
- void `cb_emit_sort_init` (`cb_tree` name, `cb_tree` keys, `cb_tree` col)
- void `cb_emit_sort_using` (`cb_tree` file, `cb_tree` l)
- void `cb_emit_sort_input` (`cb_tree` proc)
- void `cb_emit_sort_giving` (`cb_tree` file, `cb_tree` l)
- void `cb_emit_sort_output` (`cb_tree` proc)
- void `cb_emit_sort_finish` (`cb_tree` file)
- void `cb_emit_start` (`cb_tree` file, `cb_tree` op, `cb_tree` key)
- void `cb_emit_stop_run` (`cb_tree` x)
- void `cb_emit_string` (`cb_tree` items, `cb_tree` into, `cb_tree` pointer)
- void `cb_emit_unlock` (`cb_tree` ref)
- void `cb_emit_unstring` (`cb_tree` name, `cb_tree` delimited, `cb_tree` into, `cb_tree` pointer, `cb_tree` tallying)
- `cb_tree` `cb_build_unstring_delimited` (`cb_tree` all, `cb_tree` value)
- `cb_tree` `cb_build_unstring_into` (`cb_tree` name, `cb_tree` delimiter, `cb_tree` count)
- void `cb_emit_write` (`cb_tree` record, `cb_tree` from, `cb_tree` opt, `cb_tree` lockopt)
- `cb_tree` `cb_build_write_advancing_lines` (`cb_tree` pos, `cb_tree` lines)
- `cb_tree` `cb_build_write_advancing_mnemonic` (`cb_tree` pos, `cb_tree` mnemonic)
- `cb_tree` `cb_build_write_advancing_page` (`cb_tree` pos)
- void `cobc_tree_cast_error` (`cb_tree` x, const char \*filen, const int linenum, const int tagnum)
- void `codegen` (struct `cb_program` \*prog, int nested)
- void `cb_set_in_procedure` (void)
- void `cb_reset_in_procedure` (void)

- void `cb_add_78` (struct `cb_field` \*f)
- void `cb_reset_78` (void)
- struct `cb_field` \* `check_level_78` (const char \*name)

## Variables

- `cb_tree` `cb_any`
- `cb_tree` `cb_true`
- `cb_tree` `cb_false`
- `cb_tree` `cb_null`
- `cb_tree` `cb_zero`
- `cb_tree` `cb_one`
- `cb_tree` `cb_space`
- `cb_tree` `cb_low`
- `cb_tree` `cb_high`
- `cb_tree` `cb_norm_low`
- `cb_tree` `cb_norm_high`
- `cb_tree` `cb_quote`
- `cb_tree` `cb_int0`
- `cb_tree` `cb_int1`
- `cb_tree` `cb_int2`
- `cb_tree` `cb_int3`
- `cb_tree` `cb_int4`
- `cb_tree` `cb_int5`
- `cb_tree` `cb_i` [8]
- `cb_tree` `cb_error_node`
- `cb_tree` `cb_intr_whencomp`
- `cb_tree` `cb_intr_pi`
- `cb_tree` `cb_intr_e`
- `cb_tree` `cb_standard_error_handler`
- `size_t` `gen_screen_ptr`
- `int` `non_const_word`
- `size_t` `cb_needs_01`

### 7.16.1 Define Documentation

#### 7.16.1.1 `#define CB_AFTER cb_int1`

Definition at line 27 of file tree.h.

#### 7.16.1.2 `#define CB_ALPHABET_NAME( x ) (CB_TREE_CAST (CB_TAG_ALPHABET_NAME, struct cb_alphabet_name, x))`

Definition at line 338 of file tree.h.

7.16.1.3 `#define CB_ALPHABET_NAME_P( x ) (CB_TREE_TAG (x) ==  
CB_TAG_ALPHABET_NAME)`

Definition at line 339 of file tree.h.

7.16.1.4 `#define CB_ASSIGN( x ) (CB_TREE_CAST (CB_TAG_ASSIGN, struct cb_assign, x))`

Definition at line 799 of file tree.h.

7.16.1.5 `#define CB_ASSIGN_P( x ) (CB_TREE_TAG (x) == CB_TAG_ASSIGN)`

Definition at line 800 of file tree.h.

7.16.1.6 `#define CB_BEFORE cb_int0`

Definition at line 26 of file tree.h.

7.16.1.7 `#define CB_BINARY_OP( x ) (CB_TREE_CAST (CB_TAG_BINARY_OP, struct  
cb_binary_op, x))`

Definition at line 711 of file tree.h.

7.16.1.8 `#define CB_BINARY_OP_P( x ) (CB_TREE_TAG (x) == CB_TAG_BINARY_OP)`

Definition at line 712 of file tree.h.

7.16.1.9 `#define cb_build_cast_addr_of_addr( x ) cb_build_cast (CB_CAST_ADDR_OF_ADDR, x)`

Definition at line 784 of file tree.h.

7.16.1.10 `#define cb_build_cast_address( x ) cb_build_cast (CB_CAST_ADDRESS, x)`

Definition at line 783 of file tree.h.

7.16.1.11 `#define cb_build_cast_integer( x ) cb_build_cast (CB_CAST_INTEGER, x)`

Definition at line 782 of file tree.h.

7.16.1.12 `#define cb_build_cast_length( x ) cb_build_cast (CB_CAST_LENGTH, x)`

Definition at line 785 of file tree.h.

7.16.1.13 `#define cb_build_cast_ppointer( x ) cb_build_cast (CB_CAST_PROGRAM_POINTER, x)`

Definition at line 786 of file tree.h.

7.16.1.14 `#define cb_build_funcall_0( f ) cb_build_funcall(f, 0, NULL, NULL, NULL, NULL, NULL, NULL)`

Definition at line 741 of file tree.h.

7.16.1.15 `#define cb_build_funcall_1( f, a1 ) cb_build_funcall(f, 1, a1, NULL, NULL, NULL, NULL, NULL)`

Definition at line 743 of file tree.h.

7.16.1.16 `#define cb_build_funcall_2( f, a1, a2 ) cb_build_funcall(f, 2, a1, a2, NULL, NULL, NULL, NULL)`

Definition at line 745 of file tree.h.

7.16.1.17 `#define cb_build_funcall_3( f, a1, a2, a3 ) cb_build_funcall(f, 3, a1, a2, a3, NULL, NULL, NULL)`

Definition at line 747 of file tree.h.

7.16.1.18 `#define cb_build_funcall_4( f, a1, a2, a3, a4 ) cb_build_funcall(f, 4, a1, a2, a3, a4, NULL, NULL, NULL)`

Definition at line 749 of file tree.h.

7.16.1.19 `#define cb_build_funcall_5( f, a1, a2, a3, a4, a5 ) cb_build_funcall(f, 5, a1, a2, a3, a4, a5, NULL, NULL)`

Definition at line 751 of file tree.h.

7.16.1.20 `#define cb_build_funcall_6( f, a1, a2, a3, a4, a5, a6 ) cb_build_funcall(f, 6, a1, a2, a3, a4, a5, a6, NULL)`

Definition at line 753 of file tree.h.

7.16.1.21 `#define cb_build_funcall_7( f, a1, a2, a3, a4, a5, a6, a7 ) cb_build_funcall(f, 7, a1, a2, a3, a4, a5, a6, a7)`

Definition at line 755 of file tree.h.



7.16.1.22 `#define cb_build_negation( x ) cb_build_binary_op( x, '!', NULL)`

Definition at line 715 of file tree.h.

7.16.1.23 `#define cb_build_pair( x, y ) cb_build_list( x, y, NULL)`

Definition at line 1147 of file tree.h.

7.16.1.24 `#define cb_build_parenthesis( x ) cb_build_binary_op( x, '@', NULL)`

Definition at line 714 of file tree.h.

7.16.1.25 `#define cb_build_string0( str ) cb_build_string( str, strlen( (char *)str)`

Definition at line 319 of file tree.h.

7.16.1.26 `#define CB_CALL( x ) (CB_TREE_CAST( CB_TAG_CALL, struct cb_call, x))`

Definition at line 984 of file tree.h.

7.16.1.27 `#define CB_CALL_BY_CONTENT 2`

Definition at line 971 of file tree.h.

7.16.1.28 `#define CB_CALL_BY_REFERENCE 1`

Definition at line 970 of file tree.h.

7.16.1.29 `#define CB_CALL_BY_VALUE 3`

Definition at line 972 of file tree.h.

7.16.1.30 `#define CB_CALL_P( x ) (CB_TREE_TAG( x) == CB_TAG_CALL)`

Definition at line 985 of file tree.h.

7.16.1.31 `#define CB_CAST( x ) (CB_TREE_CAST( CB_TAG_CAST, struct cb_cast, x))`

Definition at line 777 of file tree.h.

7.16.1.32 `#define CB_CAST_P( x ) (CB_TREE_TAG (x) == CB_TAG_CAST)`

Definition at line 778 of file tree.h.

7.16.1.33 `#define CB_CHAIN( x ) (CB_LIST (x)->chain)`

Definition at line 1117 of file tree.h.

7.16.1.34 `#define CB_CLASS_NAME( x ) (CB_TREE_CAST (CB_TAG_CLASS_NAME, struct  
cb_class_name, x))`

Definition at line 355 of file tree.h.

7.16.1.35 `#define CB_CLASS_NAME_P( x ) (CB_TREE_TAG (x) == CB_TAG_CLASS_NAME)`

Definition at line 356 of file tree.h.

7.16.1.36 `#define cb_cons( x, l ) cb_build_list (NULL, x, l)`

Definition at line 1139 of file tree.h.

7.16.1.37 `#define CB_CONST( x ) (CB_TREE_CAST (CB_TAG_CONST, struct cb_const, x))`

Definition at line 285 of file tree.h.

7.16.1.38 `#define CB_CONST_P( x ) (CB_TREE_TAG (x) == CB_TAG_CONST)`

Definition at line 286 of file tree.h.

7.16.1.39 `#define CB_CONTINUE( x ) (CB_TREE_CAST (CB_TAG_CONTINUE, struct  
cb_continue, x))`

Definition at line 1094 of file tree.h.

7.16.1.40 `#define CB_CONTINUE_P( x ) (CB_TREE_TAG (x) == CB_TAG_CONTINUE)`

Definition at line 1095 of file tree.h.

7.16.1.41 `#define CB_DECIMAL( x ) (CB_TREE_CAST (CB_TAG_DECIMAL, struct cb_decimal,  
x))`

Definition at line 426 of file tree.h.

7.16.1.42 `#define CB_DECIMAL_P( x )(CB_TREE_TAG(x) == CB_TAG_DECIMAL)`

Definition at line 427 of file tree.h.

7.16.1.43 `#define CB_FIELD( x )(CB_TREE_CAST(CB_TAG_FIELD, struct cb_field, x))`

Definition at line 534 of file tree.h.

7.16.1.44 `#define CB_FIELD_P( x )(CB_TREE_TAG(x) == CB_TAG_FIELD)`

Definition at line 535 of file tree.h.

7.16.1.45 `#define CB_FILE( x )(CB_TREE_CAST(CB_TAG_FILE, struct cb_file, x))`

Definition at line 634 of file tree.h.

7.16.1.46 `#define CB_FILE_P( x )(CB_TREE_TAG(x) == CB_TAG_FILE)`

Definition at line 635 of file tree.h.

7.16.1.47 `#define CB_FUNCALL( x )(CB_TREE_CAST(CB_TAG_FUNCALL, struct cb_funcall, x))`

Definition at line 734 of file tree.h.

7.16.1.48 `#define CB_FUNCALL_P( x )(CB_TREE_TAG(x) == CB_TAG_FUNCALL)`

Definition at line 735 of file tree.h.

7.16.1.49 `#define CB_FUNCTION_TYPE 1`

Definition at line 42 of file tree.h.

7.16.1.50 `#define CB_GOTO( x )(CB_TREE_CAST(CB_TAG_GOTO, struct cb_goto, x))`

Definition at line 1001 of file tree.h.

7.16.1.51 `#define CB_GOTO_P( x )(CB_TREE_TAG(x) == CB_TAG_GOTO)`

Definition at line 1002 of file tree.h.

7.16.1.52 `#define CB_IF( x )(CB_TREE_CAST (CB_TAG_IF, struct cb_if, x))`

Definition at line 1018 of file tree.h.

7.16.1.53 `#define CB_IF_P( x )(CB_TREE_TAG (x) == CB_TAG_IF)`

Definition at line 1019 of file tree.h.

7.16.1.54 `#define CB_INDEX_P( x )`

**Value:**

```
((CB_FIELD_P (x) || CB_REFERENCE_P (x)) \
  && cb_field (x)->usage == CB_USAGE_INDEX)
```

Definition at line 553 of file tree.h.

7.16.1.55 `#define CB_INITIALIZE( x )(CB_TREE_CAST (CB_TAG_INITIALIZE, struct  
cb_initialize, x))`

Definition at line 941 of file tree.h.

7.16.1.56 `#define CB_INITIALIZE_P( x )(CB_TREE_TAG (x) == CB_TAG_INITIALIZE)`

Definition at line 942 of file tree.h.

7.16.1.57 `#define CB_INTEGER( x )(CB_TREE_CAST (CB_TAG_INTEGER, struct cb_integer,  
x))`

Definition at line 300 of file tree.h.

7.16.1.58 `#define CB_INTEGER_P( x )(CB_TREE_TAG (x) == CB_TAG_INTEGER)`

Definition at line 301 of file tree.h.

7.16.1.59 `#define CB_INTRINSIC( x )(CB_TREE_CAST (CB_TAG_INTRINSIC, struct  
cb_intrinsic, x))`

Definition at line 917 of file tree.h.

7.16.1.60 `#define CB_INTRINSIC_P( x )(CB_TREE_TAG (x) == CB_TAG_INTRINSIC)`

Definition at line 918 of file tree.h.

7.16.1.61 `#define CB_LABEL( x ) (CB_TREE_CAST (CB_TAG_LABEL, struct cb_label, x))`

Definition at line 578 of file tree.h.

7.16.1.62 `#define CB_LABEL_P( x ) (CB_TREE_TAG (x) == CB_TAG_LABEL)`

Definition at line 579 of file tree.h.

7.16.1.63 `#define CB_LIST( x ) (CB_TREE_CAST (CB_TAG_LIST, struct cb_list, x))`

Definition at line 1112 of file tree.h.

7.16.1.64 `#define cb_list_init( x ) cb_build_list (NULL, x, NULL)`

Definition at line 1138 of file tree.h.

7.16.1.65 `#define CB_LIST_P( x ) (CB_TREE_TAG (x) == CB_TAG_LIST)`

Definition at line 1113 of file tree.h.

7.16.1.66 `#define CB_LITERAL( x ) (CB_TREE_CAST (CB_TAG_LITERAL, struct cb_literal, x))`

Definition at line 407 of file tree.h.

7.16.1.67 `#define CB_LITERAL_P( x ) (CB_TREE_TAG (x) == CB_TAG_LITERAL)`

Definition at line 408 of file tree.h.

7.16.1.68 `#define CB_LOCALE_NAME( x ) (CB_TREE_CAST (CB_TAG_LOCALE_NAME, struct  
cb_locale_name, x))`

Definition at line 372 of file tree.h.

7.16.1.69 `#define CB_LOCALE_NAME_P( x ) (CB_TREE_TAG (x) == CB_TAG_LOCALE_NAME)`

Definition at line 373 of file tree.h.

7.16.1.70 `#define CB_NAME( x ) (CB_REFERENCE (x)->word->name)`

Definition at line 672 of file tree.h.

7.16.1.71 `#define CB_NUMERIC_LITERAL_P( x ) (CB_LITERAL_P (x) && CB_TREE_CATEGORY  
(x) == CB_CATEGORY_NUMERIC)`

Definition at line 409 of file tree.h.

7.16.1.72 `#define CB_PAIR_P( x ) (CB_LIST_P (x) && CB_PAIR_X (x))`

Definition at line 1143 of file tree.h.

7.16.1.73 `#define CB_PAIR_X( x ) CB_PURPOSE (x)`

Definition at line 1144 of file tree.h.

7.16.1.74 `#define CB_PAIR_Y( x ) CB_VALUE (x)`

Definition at line 1145 of file tree.h.

7.16.1.75 `#define CB_PERFORM( x ) (CB_TREE_CAST (CB_TAG_PERFORM, struct  
cb_perform, x))`

Definition at line 1057 of file tree.h.

7.16.1.76 `#define CB_PERFORM_P( x ) (CB_TREE_TAG (x) == CB_TAG_PERFORM)`

Definition at line 1058 of file tree.h.

7.16.1.77 `#define CB_PERFORM_VARYING( x ) (CB_TREE_CAST (CB_TAG_PERFORM_VARYING,  
struct cb_perform_varying, x))`

Definition at line 1055 of file tree.h.

7.16.1.78 `#define CB_PICTURE( x ) (CB_TREE_CAST (CB_TAG_PICTURE, struct cb_picture,  
x))`

Definition at line 449 of file tree.h.

7.16.1.79 `#define CB_PICTURE_P( x ) (CB_TREE_TAG (x) == CB_TAG_PICTURE)`

Definition at line 450 of file tree.h.

7.16.1.80 `#define CB_PREFIX_ATTR "a_"`

Definition at line 31 of file tree.h.

7.16.1.81 `#define CB_PREFIX_BASE "b_"`

Definition at line 32 of file tree.h.

7.16.1.82 `#define CB_PREFIX_CONST "c_"`

Definition at line 33 of file tree.h.

7.16.1.83 `#define CB_PREFIX_DECIMAL "d_"`

Definition at line 34 of file tree.h.

7.16.1.84 `#define CB_PREFIX_FIELD "f_"`

Definition at line 35 of file tree.h.

7.16.1.85 `#define CB_PREFIX_FILE "h_"`

Definition at line 36 of file tree.h.

7.16.1.86 `#define CB_PREFIX_KEYS "k_"`

Definition at line 37 of file tree.h.

7.16.1.87 `#define CB_PREFIX_LABEL "l_"`

Definition at line 38 of file tree.h.

7.16.1.88 `#define CB_PREFIX_SEQUENCE "s_"`

Definition at line 39 of file tree.h.

7.16.1.89 `#define CB_PROGRAM_TYPE 0`

Definition at line 41 of file tree.h.

7.16.1.90 `#define CB_PURPOSE( x ) (CB_LIST(x)->purpose)`

Definition at line 1115 of file tree.h.

7.16.1.91 `#define CB_PURPOSE_INT( x ) (CB_INTEGER (CB_PURPOSE (x))->val)`

Definition at line 1120 of file tree.h.

7.16.1.92 `#define CB_REF_OR_FIELD_P( x ) ((CB_FIELD_P (x) || CB_REFERENCE_P (x)))`

Definition at line 549 of file tree.h.

7.16.1.93 `#define CB_REFERENCE( x ) (CB_TREE_CAST (CB_TAG_REFERENCE, struct  
cb_reference, x))`

Definition at line 669 of file tree.h.

7.16.1.94 `#define CB_REFERENCE_P( x ) (CB_TREE_TAG (x) == CB_TAG_REFERENCE)`

Definition at line 670 of file tree.h.

7.16.1.95 `#define CB_SEARCH( x ) (CB_TREE_CAST (CB_TAG_SEARCH, struct cb_search, x))`

Definition at line 960 of file tree.h.

7.16.1.96 `#define CB_SEARCH_P( x ) (CB_TREE_TAG (x) == CB_TAG_SEARCH)`

Definition at line 961 of file tree.h.

7.16.1.97 `#define CB_SIZE_1 1`

Definition at line 1123 of file tree.h.

7.16.1.98 `#define CB_SIZE_2 2`

Definition at line 1124 of file tree.h.

7.16.1.99 `#define CB_SIZE_4 3`

Definition at line 1125 of file tree.h.

7.16.1.100 `#define CB_SIZE_8 4`

Definition at line 1126 of file tree.h.



7.16.1.101 `#define CB_SIZE_AUTO 0`

Definition at line 1122 of file tree.h.

7.16.1.102 `#define CB_SIZE_UNSIGNED 8`

Definition at line 1127 of file tree.h.

7.16.1.103 `#define CB_SIZES( x ) (CB_LIST (x)->sizes)`

Definition at line 1118 of file tree.h.

7.16.1.104 `#define CB_SIZES_INT( x ) ((CB_LIST (x)->sizes) & 0x07)`

Definition at line 1129 of file tree.h.

7.16.1.105 `#define CB_SIZES_INT_UNSIGNED( x ) ((CB_LIST (x)->sizes) & CB_SIZE_UNSIGNED)`

Definition at line 1130 of file tree.h.

7.16.1.106 `#define CB_STATEMENT( x ) (CB_TREE_CAST (CB_TAG_STATEMENT, struct  
cb_statement, x))`

Definition at line 1080 of file tree.h.

7.16.1.107 `#define CB_STATEMENT_P( x ) (CB_TREE_TAG (x) == CB_TAG_STATEMENT)`

Definition at line 1081 of file tree.h.

7.16.1.108 `#define CB_STRING( x ) (CB_TREE_CAST (CB_TAG_STRING, struct cb_string, x))`

Definition at line 316 of file tree.h.

7.16.1.109 `#define CB_STRING_P( x ) (CB_TREE_TAG (x) == CB_TAG_STRING)`

Definition at line 317 of file tree.h.

7.16.1.110 `#define CB_SYSTEM_NAME( x ) (CB_TREE_CAST (CB_TAG_SYSTEM_NAME, struct  
cb_system_name, x))`

Definition at line 387 of file tree.h.

7.16.1.111 `#define CB_SYSTEM_NAME_P( x ) (CB_TREE_TAG(x) == CB_TAG_SYSTEM_NAME)`

Definition at line 388 of file tree.h.

7.16.1.112 `#define CB_TREE( x ) ((struct cb_tree_common *) (x))`

Definition at line 219 of file tree.h.

7.16.1.113 `#define CB_TREE_CAST( tg, ty, x ) ((ty *) (x))`

Definition at line 235 of file tree.h.

7.16.1.114 `#define CB_TREE_CATEGORY( x ) cb_tree_category (CB_TREE(x))`

Definition at line 222 of file tree.h.

7.16.1.115 `#define CB_TREE_CLASS( x ) cb_tree_class (CB_TREE(x))`

Definition at line 221 of file tree.h.

7.16.1.116 `#define CB_TREE_TAG( x ) (CB_TREE(x)->tag)`

Definition at line 220 of file tree.h.

7.16.1.117 `#define CB_VALUE( x ) (CB_LIST(x)->value)`

Definition at line 1116 of file tree.h.

7.16.1.118 `#define CB_WORD_HASH_SIZE 133`

Definition at line 646 of file tree.h.

7.16.1.119 `#define COB_MAX_SUBSCRIPTS 16`

Definition at line 29 of file tree.h.

7.16.1.120 `#define YYSTYPE cb_tree`

Definition at line 24 of file tree.h.

## 7.16.2 Typedef Documentation

### 7.16.2.1 typedef struct cb\_tree\_common\* cb\_tree

Definition at line 217 of file tree.h.

## 7.16.3 Enumeration Type Documentation

### 7.16.3.1 enum cb\_alphabet\_name\_type

Enumerator:

```
CB_ALPHABET_NATIVE  
CB_ALPHABET_STANDARD_1  
CB_ALPHABET_STANDARD_2  
CB_ALPHABET_EBCDIC  
CB_ALPHABET_CUSTOM
```

Definition at line 80 of file tree.h.

```
        {  
        CB_ALPHABET_NATIVE,  
        CB_ALPHABET_STANDARD_1,  
        CB_ALPHABET_STANDARD_2,  
        CB_ALPHABET_EBCDIC,  
        CB_ALPHABET_CUSTOM  
};
```

### 7.16.3.2 enum cb\_cast\_type

Enumerator:

```
CB_CAST_INTEGER  
CB_CAST_ADDRESS  
CB_CAST_ADDR_OF_ADDR  
CB_CAST_LENGTH  
CB_CAST_PROGRAM_POINTER
```

Definition at line 763 of file tree.h.

```
        {  
        CB_CAST_INTEGER,  
        CB_CAST_ADDRESS,  
        CB_CAST_ADDR_OF_ADDR,  
        CB_CAST_LENGTH,  
        CB_CAST_PROGRAM_POINTER  
};
```

7.16.3.3 enum `cb_category`

Enumerator:

**`CB_CATEGORY_UNKNOWN`**  
**`CB_CATEGORY_ALPHABETIC`**  
**`CB_CATEGORY_ALPHANUMERIC`**  
**`CB_CATEGORY_ALPHANUMERIC_EDITED`**  
**`CB_CATEGORY_BOOLEAN`**  
**`CB_CATEGORY_INDEX`**  
**`CB_CATEGORY_NATIONAL`**  
**`CB_CATEGORY_NATIONAL_EDITED`**  
**`CB_CATEGORY_NUMERIC`**  
**`CB_CATEGORY_NUMERIC_EDITED`**  
**`CB_CATEGORY_OBJECT_REFERENCE`**  
**`CB_CATEGORY_DATA_POINTER`**  
**`CB_CATEGORY_PROGRAM_POINTER`**

Definition at line 147 of file `tree.h`.

```

{
    CB_CATEGORY_UNKNOWN,          /* 0 */
    CB_CATEGORY_ALPHABETIC,      /* 1 */
    CB_CATEGORY_ALPHANUMERIC,    /* 2 */
    CB_CATEGORY_ALPHANUMERIC_EDITED, /* 3 */
    CB_CATEGORY_BOOLEAN,        /* 4 */
    CB_CATEGORY_INDEX,          /* 5 */
    CB_CATEGORY_NATIONAL,       /* 6 */
    CB_CATEGORY_NATIONAL_EDITED, /* 7 */
    CB_CATEGORY_NUMERIC,        /* 8 */
    CB_CATEGORY_NUMERIC_EDITED,  /* 9 */
    CB_CATEGORY_OBJECT_REFERENCE, /* 10 */
    CB_CATEGORY_DATA_POINTER,    /* 11 */
    CB_CATEGORY_PROGRAM_POINTER  /* 12 */
};

```

7.16.3.4 enum `cb_class`

Enumerator:

**`CB_CLASS_UNKNOWN`**  
**`CB_CLASS_ALPHABETIC`**  
**`CB_CLASS_ALPHANUMERIC`**  
**`CB_CLASS_BOOLEAN`**  
**`CB_CLASS_INDEX`**  
**`CB_CLASS_NATIONAL`**  
**`CB_CLASS_NUMERIC`**

**CB\_CLASS\_OBJECT**  
**CB\_CLASS\_POINTER**

Definition at line 135 of file tree.h.

```
    {  
    CB_CLASS_UNKNOWN,          /* 0 */  
    CB_CLASS_ALPHABETIC,     /* 1 */  
    CB_CLASS_ALPHANUMERIC,   /* 2 */  
    CB_CLASS_BOOLEAN,       /* 3 */  
    CB_CLASS_INDEX,         /* 4 */  
    CB_CLASS_NATIONAL,      /* 5 */  
    CB_CLASS_NUMERIC,       /* 6 */  
    CB_CLASS_OBJECT,        /* 7 */  
    CB_CLASS_POINTER        /* 8 */  
};
```

## 7.16.3.5 enum cb\_device\_name

Enumerator:

**CB\_DEVICE\_SYSIN**  
**CB\_DEVICE\_SYSOUT**  
**CB\_DEVICE\_SYSERR**  
**CB\_DEVICE\_CONSOLE**

Definition at line 101 of file tree.h.

```
    {  
    CB_DEVICE_SYSIN,  
    CB_DEVICE_SYSOUT,  
    CB_DEVICE_SYSERR,  
    CB_DEVICE_CONSOLE  
};
```

## 7.16.3.6 enum cb\_feature\_name

Enumerator:

**CB\_FEATURE\_FORMFEED**  
**CB\_FEATURE\_C01**  
**CB\_FEATURE\_C02**  
**CB\_FEATURE\_C03**  
**CB\_FEATURE\_C04**  
**CB\_FEATURE\_C05**  
**CB\_FEATURE\_C06**  
**CB\_FEATURE\_C07**

***CB\_FEATURE\_C08***  
***CB\_FEATURE\_C09***  
***CB\_FEATURE\_C10***  
***CB\_FEATURE\_C11***  
***CB\_FEATURE\_C12***

Definition at line 108 of file tree.h.

```
        {  
        CB_FEATURE_FORMFEED,  
        CB_FEATURE_C01,  
        CB_FEATURE_C02,  
        CB_FEATURE_C03,  
        CB_FEATURE_C04,  
        CB_FEATURE_C05,  
        CB_FEATURE_C06,  
        CB_FEATURE_C07,  
        CB_FEATURE_C08,  
        CB_FEATURE_C09,  
        CB_FEATURE_C10,  
        CB_FEATURE_C11,  
        CB_FEATURE_C12  
};
```

#### 7.16.3.7 enum cb\_intr\_enum

Enumerator:

***CB\_INTR\_ABS***  
***CB\_INTR\_ACOS***  
***CB\_INTR\_ANNUITY***  
***CB\_INTR\_ASIN***  
***CB\_INTR\_ATAN***  
***CB\_INTR\_BOOLEAN\_OF\_INTEGER***  
***CB\_INTR\_BYTE\_LENGTH***  
***CB\_INTR\_CHAR***  
***CB\_INTR\_CHAR\_NATIONAL***  
***CB\_INTR\_COMBINED\_DATETIME***  
***CB\_INTR\_CONCATENATE***  
***CB\_INTR\_COS***  
***CB\_INTR\_CURRENT\_DATE***  
***CB\_INTR\_DATE\_OF\_INTEGER***  
***CB\_INTR\_DATE\_TO\_YYYYMMDD***  
***CB\_INTR\_DAY\_OF\_INTEGER***  
***CB\_INTR\_DAY\_TO\_YYYYDDD***

*CB\_INTR\_DISPLAY\_OF*  
*CB\_INTR\_E*  
*CB\_INTR\_EXCEPTION\_FILE*  
*CB\_INTR\_EXCEPTION\_FILE\_N*  
*CB\_INTR\_EXCEPTION\_LOCATION*  
*CB\_INTR\_EXCEPTION\_LOCATION\_N*  
*CB\_INTR\_EXCEPTION\_STATEMENT*  
*CB\_INTR\_EXCEPTION\_STATUS*  
*CB\_INTR\_EXP*  
*CB\_INTR\_EXP10*  
*CB\_INTR\_FACTORIAL*  
*CB\_INTR\_FRACTION\_PART*  
*CB\_INTR\_HIGHEST\_ALGEBRAIC*  
*CB\_INTR\_INTEGER*  
*CB\_INTR\_INTEGER\_OF\_BOOLEAN*  
*CB\_INTR\_INTEGER\_OF\_DATE*  
*CB\_INTR\_INTEGER\_OF\_DAY*  
*CB\_INTR\_INTEGER\_PART*  
*CB\_INTR\_LENGTH*  
*CB\_INTR\_LOCALE\_COMPARE*  
*CB\_INTR\_LOCALE\_DATE*  
*CB\_INTR\_LOCALE\_TIME*  
*CB\_INTR\_LOCALE\_TIME\_FROM\_SECS*  
*CB\_INTR\_LOG*  
*CB\_INTR\_LOG10*  
*CB\_INTR\_LOWER\_CASE*  
*CB\_INTR\_LOWEST\_ALGEBRAIC*  
*CB\_INTR\_MAX*  
*CB\_INTR\_MEAN*  
*CB\_INTR\_MEDIAN*  
*CB\_INTR\_MIDRANGE*  
*CB\_INTR\_MIN*  
*CB\_INTR\_MOD*  
*CB\_INTR\_NATIONAL\_OF*  
*CB\_INTR\_NUMVAL*  
*CB\_INTR\_NUMVAL\_C*  
*CB\_INTR\_NUMVAL\_F*  
*CB\_INTR\_ORD*

**CB\_INTR\_ORD\_MAX**  
**CB\_INTR\_ORD\_MIN**  
**CB\_INTR\_PI**  
**CB\_INTR\_PRESENT\_VALUE**  
**CB\_INTR\_RANDOM**  
**CB\_INTR\_RANGE**  
**CB\_INTR\_REM**  
**CB\_INTR\_REVERSE**  
**CB\_INTR\_SECONDS\_FROM\_FORMATTED\_TIME**  
**CB\_INTR\_SECONDS\_PAST\_MIDNIGHT**  
**CB\_INTR\_SIGN**  
**CB\_INTR\_SIN**  
**CB\_INTR\_SQRT**  
**CB\_INTR\_STANDARD\_COMPARE**  
**CB\_INTR\_STANDARD\_DEVIATION**  
**CB\_INTR\_STORED\_CHAR\_LENGTH**  
**CB\_INTR\_SUBSTITUTE**  
**CB\_INTR\_SUBSTITUTE\_CASE**  
**CB\_INTR\_SUM**  
**CB\_INTR\_TAN**  
**CB\_INTR\_TEST\_DATE\_YYYYMMDD**  
**CB\_INTR\_TEST\_DAY\_YYYYDDD**  
**CB\_INTR\_TEST\_NUMVAL**  
**CB\_INTR\_TEST\_NUMVAL\_C**  
**CB\_INTR\_TEST\_NUMVAL\_F**  
**CB\_INTR\_TRIM**  
**CB\_INTR\_UPPER\_CASE**  
**CB\_INTR\_VARIANCE**  
**CB\_INTR\_WHEN\_COMPILED**  
**CB\_INTR\_YEAR\_TO\_YYYY**

Definition at line 809 of file tree.h.

```
{  
    CB_INTR_ABS = 1,  
    CB_INTR_ACOS,  
    CB_INTR_ANNUITY,  
    CB_INTR_ASIN,  
    CB_INTR_ATAN,  
    CB_INTR_BOOLEAN_OF_INTEGER,  
    CB_INTR_BYTE_LENGTH,  
    CB_INTR_CHAR,
```



```
CB_INTR_CHAR_NATIONAL,  
CB_INTR_COMBINED_DATETIME,  
CB_INTR_CONCATENATE,  
CB_INTR_COS,  
CB_INTR_CURRENT_DATE,  
CB_INTR_DATE_OF_INTEGER,  
CB_INTR_DATE_TO_YYYYMMDD,  
CB_INTR_DAY_OF_INTEGER,  
CB_INTR_DAY_TO_YYYYDDD,  
CB_INTR_DISPLAY_OF,  
CB_INTR_E,  
CB_INTR_EXCEPTION_FILE,  
CB_INTR_EXCEPTION_FILE_N,  
CB_INTR_EXCEPTION_LOCATION,  
CB_INTR_EXCEPTION_LOCATION_N,  
CB_INTR_EXCEPTION_STATEMENT,  
CB_INTR_EXCEPTION_STATUS,  
CB_INTR_EXP,  
CB_INTR_EXP10,  
CB_INTR_FACTORIAL,  
CB_INTR_FRACTION_PART,  
CB_INTR_HIGHEST_ALGEBRAIC,  
CB_INTR_INTEGER,  
CB_INTR_INTEGER_OF_BOOLEAN,  
CB_INTR_INTEGER_OF_DATE,  
CB_INTR_INTEGER_OF_DAY,  
CB_INTR_INTEGER_PART,  
CB_INTR_LENGTH,  
CB_INTR_LOCALE_COMPARE,  
CB_INTR_LOCALE_DATE,  
CB_INTR_LOCALE_TIME,  
CB_INTR_LOCALE_TIME_FROM_SECS,  
CB_INTR_LOG,  
CB_INTR_LOG10,  
CB_INTR_LOWER_CASE,  
CB_INTR_LOWEST_ALGEBRAIC,  
CB_INTR_MAX,  
CB_INTR_MEAN,  
CB_INTR_MEDIAN,  
CB_INTR_MIDRANGE,  
CB_INTR_MIN,  
CB_INTR_MOD,  
CB_INTR_NATIONAL_OF,  
CB_INTR_NUMVAL,  
CB_INTR_NUMVAL_C,  
CB_INTR_NUMVAL_F,  
CB_INTR_ORD,  
CB_INTR_ORD_MAX,  
CB_INTR_ORD_MIN,  
CB_INTR_PI,  
CB_INTR_PRESENT_VALUE,  
CB_INTR_RANDOM,  
CB_INTR_RANGE,  
CB_INTR_REM,  
CB_INTR_REVERSE,  
CB_INTR_SECONDS_FROM_FORMATTED_TIME,  
CB_INTR_SECONDS_PAST_MIDNIGHT,  
CB_INTR_SIGN,  
CB_INTR_SIN,  
CB_INTR_SQRT,  
CB_INTR_STANDARD_COMPARE,  
CB_INTR_STANDARD_DEVIATION,
```

```
    CB_INTR_STORED_CHAR_LENGTH,  
    CB_INTR_SUBSTITUTE,  
    CB_INTR_SUBSTITUTE_CASE,  
    CB_INTR_SUM,  
    CB_INTR_TAN,  
    CB_INTR_TEST_DATE_YYYYMMDD,  
    CB_INTR_TEST_DAY_YYYYDDD,  
    CB_INTR_TEST_NUMVAL,  
    CB_INTR_TEST_NUMVAL_C,  
    CB_INTR_TEST_NUMVAL_F,  
    CB_INTR_TRIM,  
    CB_INTR_UPPER_CASE,  
    CB_INTR_VARIANCE,  
    CB_INTR_WHEN_COMPILED,  
    CB_INTR_YEAR_TO_YYYY  
};
```

#### 7.16.3.8 enum `cb_operand_type`

Enumerator:

```
CB_SENDING_OPERAND  
CB_RECEIVING_OPERAND
```

Definition at line 200 of file tree.h.

```
    {  
    CB_SENDING_OPERAND,  
    CB_RECEIVING_OPERAND  
};
```

#### 7.16.3.9 enum `cb_perform_type`

Enumerator:

```
CB_PERFORM_EXIT  
CB_PERFORM_ONCE  
CB_PERFORM_TIMES  
CB_PERFORM_UNTIL  
CB_PERFORM_FOREVER
```

Definition at line 1028 of file tree.h.

```
    {  
    CB_PERFORM_EXIT,  
    CB_PERFORM_ONCE,  
    CB_PERFORM_TIMES,  
    CB_PERFORM_UNTIL,  
    CB_PERFORM_FOREVER  
};
```

7.16.3.10 enum `cb_storage`

## Enumerator:

**`CB_STORAGE_CONSTANT`**  
**`CB_STORAGE_FILE`**  
**`CB_STORAGE_WORKING`**  
**`CB_STORAGE_LOCAL`**  
**`CB_STORAGE_LINKAGE`**  
**`CB_STORAGE_SCREEN`**  
**`CB_STORAGE_REPORT`**  
**`CB_STORAGE_COMMUNICATION`**

Definition at line 163 of file tree.h.

```
    {  
        CB_STORAGE_CONSTANT,      /* Constants */  
        CB_STORAGE_FILE,         /* FILE SECTION */  
        CB_STORAGE_WORKING,      /* WORKING-STORAGE SECTION */  
        CB_STORAGE_LOCAL,        /* LOCAL-STORAGE SECTION */  
        CB_STORAGE_LINKAGE,      /* LINKAGE SECTION */  
        CB_STORAGE_SCREEN,       /* SCREEN SECTION */  
        CB_STORAGE_REPORT,       /* REPORT SECTION */  
        CB_STORAGE_COMMUNICATION /* COMMUNICATION SECTION */  
    };
```

7.16.3.11 enum `cb_switch_name`

## Enumerator:

**`CB_SWITCH_1`**  
**`CB_SWITCH_2`**  
**`CB_SWITCH_3`**  
**`CB_SWITCH_4`**  
**`CB_SWITCH_5`**  
**`CB_SWITCH_6`**  
**`CB_SWITCH_7`**  
**`CB_SWITCH_8`**

Definition at line 124 of file tree.h.

```
    {  
        CB_SWITCH_1,  
        CB_SWITCH_2,  
        CB_SWITCH_3,  
        CB_SWITCH_4,  
        CB_SWITCH_5,  
        CB_SWITCH_6,  
        CB_SWITCH_7,  
        CB_SWITCH_8  
    };
```

7.16.3.12 enum `cb_system_name_category`

## Enumerator:

***CB\_CALL\_CONVENTION\_NAME***  
***CB\_CODE\_NAME***  
***CB\_COMPUTER\_NAME***  
***CB\_DEVICE\_NAME***  
***CB\_ENTRY\_CONVENTION\_NAME***  
***CB\_EXTERNAL\_LOCALE\_NAME***  
***CB\_FEATURE\_NAME***  
***CB\_LIBRARY\_NAME***  
***CB\_SWITCH\_NAME***  
***CB\_TEXT\_NAME***

Definition at line 88 of file `tree.h`.

```
        {  
        CB_CALL_CONVENTION_NAME,  
        CB_CODE_NAME,  
        CB_COMPUTER_NAME,  
        CB_DEVICE_NAME,  
        CB_ENTRY_CONVENTION_NAME,  
        CB_EXTERNAL_LOCALE_NAME,  
        CB_FEATURE_NAME,  
        CB_LIBRARY_NAME,  
        CB_SWITCH_NAME,  
        CB_TEXT_NAME  
};
```

7.16.3.13 enum `cb_tag`

## Enumerator:

***CB\_TAG\_CONST***  
***CB\_TAG\_INTEGER***  
***CB\_TAG\_STRING***  
***CB\_TAG\_ALPHABET\_NAME***  
***CB\_TAG\_CLASS\_NAME***  
***CB\_TAG\_LOCALE\_NAME***  
***CB\_TAG\_SYSTEM\_NAME***  
***CB\_TAG\_LITERAL***  
***CB\_TAG\_DECIMAL***  
***CB\_TAG\_FIELD***  
***CB\_TAG\_FILE***

**CB\_TAG\_REFERENCE**  
**CB\_TAG\_BINARY\_OP**  
**CB\_TAG\_FUNCALL**  
**CB\_TAG\_CAST**  
**CB\_TAG\_INTRINSIC**  
**CB\_TAG\_LABEL**  
**CB\_TAG\_ASSIGN**  
**CB\_TAG\_INITIALIZE**  
**CB\_TAG\_SEARCH**  
**CB\_TAG\_CALL**  
**CB\_TAG\_GOTO**  
**CB\_TAG\_IF**  
**CB\_TAG\_PERFORM**  
**CB\_TAG\_STATEMENT**  
**CB\_TAG\_CONTINUE**  
**CB\_TAG\_PERFORM\_VARYING**  
**CB\_TAG\_PICTURE**  
**CB\_TAG\_LIST**

Definition at line 44 of file tree.h.

```

{
/* primitives */
CB_TAG_CONST,          /* 0 constant value */
CB_TAG_INTEGER,       /* 1 integer constant */
CB_TAG_STRING,        /* 2 string constant */
CB_TAG_ALPHABET_NAME, /* 3 alphabet-name */
CB_TAG_CLASS_NAME,   /* 4 class-name */
CB_TAG_LOCALE_NAME,  /* 5 locale-name */
CB_TAG_SYSTEM_NAME,  /* 6 system-name */
CB_TAG_LITERAL,      /* 7 numeric/alphanumeric literal */
CB_TAG_DECIMAL,      /* 8 decimal number */
CB_TAG_FIELD,        /* 9 user-defined variable */
CB_TAG_FILE,         /* 10 file description */
/* expressions */
CB_TAG_REFERENCE,    /* 11 reference to a field, file, or label */
CB_TAG_BINARY_OP,   /* 12 binary operation */
CB_TAG_FUNCALL,     /* 13 run-time function call */
CB_TAG_CAST,        /* 14 type cast */
CB_TAG_INTRINSIC,   /* 15 intrinsic function */
/* statements */
CB_TAG_LABEL,       /* 16 label statement */
CB_TAG_ASSIGN,      /* 17 assignment statement */
CB_TAG_INITIALIZE,  /* 18 INITIALIZE statement */
CB_TAG_SEARCH,      /* 19 SEARCH statement */
CB_TAG_CALL,        /* 20 CALL statement */
CB_TAG_GOTO,        /* 21 GO TO statement */
CB_TAG_IF,          /* 22 IF statement */
CB_TAG_PERFORM,     /* 23 PERFORM statement */
CB_TAG_STATEMENT,   /* 24 general statement */

```

```

    CB_TAG_CONTINUE,          /* 25 CONTINUE statement */
    /* miscellaneous */
    CB_TAG_PERFORM_VARYING, /* 26 PERFORM VARYING parameter */
    CB_TAG_PICTURE,         /* 27 PICTURE clause */
    CB_TAG_LIST             /* 28 list */
};

```

#### 7.16.3.14 enum cb\_usage

##### Enumerator:

```

CB_USAGE_BINARY
CB_USAGE_BIT
CB_USAGE_COMP_5
CB_USAGE_COMP_X
CB_USAGE_DISPLAY
CB_USAGE_FLOAT
CB_USAGE_DOUBLE
CB_USAGE_INDEX
CB_USAGE_NATIONAL
CB_USAGE_OBJECT
CB_USAGE_PACKED
CB_USAGE_POINTER
CB_USAGE_PROGRAM
CB_USAGE_LENGTH
CB_USAGE_PROGRAM_POINTER
CB_USAGE_UNSIGNED_CHAR
CB_USAGE_SIGNED_CHAR
CB_USAGE_UNSIGNED_SHORT
CB_USAGE_SIGNED_SHORT
CB_USAGE_UNSIGNED_INT
CB_USAGE_SIGNED_INT
CB_USAGE_UNSIGNED_LONG
CB_USAGE_SIGNED_LONG

```

Definition at line 174 of file tree.h.

```

{
    CB_USAGE_BINARY,          /* 0 */
    CB_USAGE_BIT,            /* 1 */
    CB_USAGE_COMP_5,         /* 2 */
    CB_USAGE_COMP_X,         /* 3 */
    CB_USAGE_DISPLAY,        /* 4 */
    CB_USAGE_FLOAT,          /* 5 */

```

```

CB_USAGE_DOUBLE,          /* 6 */
CB_USAGE_INDEX,          /* 7 */
CB_USAGE_NATIONAL,       /* 8 */
CB_USAGE_OBJECT,         /* 9 */
CB_USAGE_PACKED,         /* 10 */
CB_USAGE_POINTER,        /* 11 */
CB_USAGE_PROGRAM,        /* 12 */
CB_USAGE_LENGTH,         /* 13 */
CB_USAGE_PROGRAM_POINTER, /* 14 */
CB_USAGE_UNSIGNED_CHAR,  /* 15 */
CB_USAGE_SIGNED_CHAR,    /* 16 */
CB_USAGE_UNSIGNED_SHORT, /* 17 */
CB_USAGE_SIGNED_SHORT,   /* 18 */
CB_USAGE_UNSIGNED_INT,   /* 19 */
CB_USAGE_SIGNED_INT,     /* 20 */
CB_USAGE_UNSIGNED_LONG,  /* 21 */
CB_USAGE_SIGNED_LONG     /* 22 */
};

```

## 7.16.4 Function Documentation

### 7.16.4.1 void ambiguous\_error ( cb\_tree x )

Definition at line 197 of file error.c.

```

{
    struct cb_word *w;
    struct cb_field *p;
    struct cb_label *l2;
    cb_tree l;
    cb_tree y;

    w = CB_REFERENCE (x)->word;
    if (w->error == 0) {
        if (!errnamebuff) {
            errnamebuff = cobc_malloc (COB_NORMAL_BUFFER);
        }
        /* display error on the first time */
        snprintf (errnamebuff, COB_NORMAL_MAX, "%s", CB_NAME (x));
        for (l = CB_REFERENCE (x)->chain; l; l = CB_REFERENCE (l)->chain)
        {
            strcat (errnamebuff, " in ");
            strcat (errnamebuff, CB_NAME (l));
            strcat (errnamebuff, " ");
        }
        cb_error_x (x, _("%s ambiguous; need qualification"), errnamebuff);
    };

    w->error = 1;

    /* display all fields with the same name */
    for (l = w->items; l; l = CB_CHAIN (l)) {
        y = CB_VALUE (l);
        snprintf (errnamebuff, COB_NORMAL_MAX, "%s' ", w->name);

        switch (CB_TREE_TAG (y)) {
        case CB_TAG_FIELD:
            for (p = CB_FIELD (y)->parent; p; p = p->parent)
            {
                strcat (errnamebuff, "in ");

```

```

        strcat (errnamebuff, p->name);
        strcat (errnamebuff, " " );
    }
    break;
case CB_TAG_LABEL:
    l2 = CB_LABEL (y);
    if (l2->section) {
        strcat (errnamebuff, "in '");
        strcat (errnamebuff, (const char *) (l2->
section->name));
        strcat (errnamebuff, " " );
    }
    break;
default:
    break;
}
strcat (errnamebuff, _("defined here"));
cb_error_x (y, errnamebuff);
}
}
}

```

#### 7.16.4.2 struct cb\_file\* build.file ( cb\_tree name ) [read]

Definition at line 1577 of file tree.c.

```

{
    struct cb_file *p;

    p = make_tree (CB_TAG_FILE, CB_CATEGORY_UNKNOWN, sizeof (struct cb_file))
;
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);

    p->organization = COB_ORG_SEQUENTIAL;
    p->access_mode = COB_ACCESS_SEQUENTIAL;
    p->handler = CB_LABEL (cb_standard_error_handler);
    p->handler_prog = current_program;
    return p;
}

```

#### 7.16.4.3 struct cb\_literal\* build.literal ( enum cb\_category category, const unsigned char \* data, size\_t size ) [read]

Definition at line 426 of file tree.c.

```

{
    struct cb_literal *p;

    p = make_tree (CB_TAG_LITERAL, category, sizeof (struct cb_literal));
    p->data = cobc_malloc ((size_t) (size + 1));
    p->size = size;
    memcpy (p->data, data, (size_t) size);
    /* RXW - malloc zeroes
    p->data[size] = 0;

```



```

    */
    return p;
}

```

#### 7.16.4.4 void cb\_add\_78 ( struct cb\_field \* f )

#### 7.16.4.5 cb\_tree cb\_build\_add ( cb\_tree v, cb\_tree n, cb\_tree round\_opt )

Definition at line 2591 of file typeck.c.

```

{
    cb_tree      opt;
    struct cb_field *f;

#ifdef COB_NON_ALIGNED
    if (CB_INDEX_P (v)) {
        return cb_build_move (cb_build_binary_op (v, '+', n), v);
    }
    if (CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        current_program->gen_ptrmanip = 1;
        return cb_build_funcall_3 ("cob_pointer_manip", v, n, cb_int0);
    }
#else
    if (CB_INDEX_P (v) || CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        return cb_build_move (cb_build_binary_op (v, '+', n), v);
    }
#endif

    if (CB_REF_OR_FIELD_P (v)) {
        f = cb_field (v);
        f->count++;
    }
    if (CB_REF_OR_FIELD_P (n)) {
        f = cb_field (n);
        f->count++;
    }
    if (round_opt == cb_high) {
        if (cb_fits_int (n)) {
            return cb_build_optim_add (v, n);
        } else {
            return cb_build_funcall_3 ("cob_add", v, n, cb_int0);
        }
    }
    opt = build_store_option (v, round_opt);
    if (opt == cb_int0 && cb_fits_int (n)) {
        return cb_build_optim_add (v, n);
    }
    return cb_build_funcall_3 ("cob_add", v, n, opt);
}

```

#### 7.16.4.6 cb\_tree cb\_build\_address ( cb\_tree x )

Definition at line 1027 of file typeck.c.

```

{

```

```

    if (x == cb_error_node ||
        (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node)) {
        return cb_error_node;
    }

    return cb_build_cast_address (x);
}

```

#### 7.16.4.7 **cb\_tree** **cb\_build\_alphabet\_name** ( *cb\_tree name*, enum *cb\_alphabet\_name\_type type* )

Definition at line 923 of file tree.c.

```

{
    struct cb_alphabet_name *p;

    p = make_tree (CB_TAG_ALPHABET_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_alphabet_name));
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);
    p->type = type;
    return CB_TREE (p);
}

```

#### 7.16.4.8 **cb\_tree** **cb\_build\_alphanumeric\_literal** ( const unsigned char \* *data*, size\_t *size* )

Definition at line 999 of file tree.c.

```

{
    return CB_TREE (build_literal (CB_CATEGORY_ALPHANUMERIC, data, size));
}

```

#### 7.16.4.9 **cb\_tree** **cb\_build\_any\_intrinsic** ( *cb\_tree args* )

Definition at line 2253 of file tree.c.

```

{
    struct cb_intrinsic_table *cbp;

    cbp = lookup_intrinsic ("LENGTH", 0);
    return make_intrinsic (NULL, cbp, args, NULL, NULL);
}

```

#### 7.16.4.10 **cb\_tree** **cb\_build\_assign** ( *cb\_tree var*, *cb\_tree val* )

Definition at line 2098 of file tree.c.

```

{
    struct cb_assign *p;

    p = make_tree (CB_TAG_ASSIGN, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_assign));
    p->var = var;
    p->val = val;
    return CB_TREE (p);
}

```

#### 7.16.4.11 `cb_tree cb_build_assignment_name ( struct cb_file * curfile, cb_tree name )`

Definition at line 677 of file typeck.c.

```

{
    const char    *s;
    const char    *p;

    if (name == cb_error_node) {
        return cb_error_node;
    }

    switch (CB_TREE_TAG (name)) {
    case CB_TAG_LITERAL:
        if (strcmp ((char *) (CB_LITERAL (name)->data), "$#@DUMMY@#$") == 0
) {
            cfile->special = 2;
        }
        return name;

    case CB_TAG_REFERENCE:
        s = CB_REFERENCE (name)->word->name;
        if (strcascmp (s, "KEYBOARD") == 0) {
            s = "#DUMMY#";
            cfile->special = 1;
            return cb_build_alphanumeric_literal ((ucharptr)s, strlen
(s));
        }
        switch (cb_assign_clause) {
        case CB_ASSIGN_COBOL2002:
            /* TODO */
            return cb_error_node;

        case CB_ASSIGN_MF:
            if (cfile->external_assign) {
                p = strrchr (s, '-');
                if (p) {
                    s = p + 1;
                }
                return cb_build_alphanumeric_literal ((ucharptr)s
, strlen (s));
            }
            current_program->reference_list =
                cb_list_add (current_program->reference_list, name);
            return name;

        case CB_ASSIGN_IBM:
            /* check organization */
            if (strncmp (s, "S-", 2) == 0 ||

```

```

        strcmp (s, "AS-", 3) == 0) {
            goto org;
        }
        /* skip the device label if exists */
        if ((p = strchr (s, '-')) != NULL) {
            s = p + 1;
        }
        /* check organization again */
        if (strcmp (s, "S-", 2) == 0 ||
            strcmp (s, "AS-", 3) == 0) {
org:
            /* skip it for now */
            s = strchr (s, '-') + 1;
        }
        /* convert the name into literal */
        return cb_build_alphanumeric_literal ((ucharptr)s, strlen
(s));
    }

    default:
        return cb_error_node;
    }
}

```

#### 7.16.4.12 `cb_tree cb_build_binary_list ( cb_tree l, int op )`

Definition at line 2019 of file tree.c.

```

{
    cb_tree e;

    e = CB_VALUE (l);
    for (l = CB_CHAIN (l); l; l = CB_CHAIN (l)) {
        e = cb_build_binary_op (e, op, CB_VALUE (l));
    }
    return e;
}

```

#### 7.16.4.13 `cb_tree cb_build_binary_op ( cb_tree x, int op, cb_tree y )`

Definition at line 1954 of file tree.c.

```

{
    struct cb_binary_op *p;
    enum cb_category category = CB_CATEGORY_UNKNOWN;

    switch (op) {
    case '+':
    case '-':
    case '*':
    case '/':
    case '^':
        /* arithmetic operators */
        if (CB_TREE_CLASS (x) == CB_CLASS_POINTER ||
            CB_TREE_CLASS (y) == CB_CLASS_POINTER) {

```

```

        category = CB_CATEGORY_DATA_POINTER;
        break;
    }
    x = cb_check_numeric_value (x);
    y = cb_check_numeric_value (y);
    if (x == cb_error_node || y == cb_error_node) {
        return cb_error_node;
    }
    category = CB_CATEGORY_NUMERIC;
    break;

case '=':
case '~':
case '<':
case '>':
case '[':
case ']':
    /* relational operators */
    category = CB_CATEGORY_BOOLEAN;
    break;

case '!':
case '&':
case '|':
    /* logical operators */
    if (CB_TREE_CLASS (x) != CB_CLASS_BOOLEAN ||
        (y && CB_TREE_CLASS (y) != CB_CLASS_BOOLEAN)) {
        cb_error (_("Invalid expression"));
        return cb_error_node;
    }
    category = CB_CATEGORY_BOOLEAN;
    break;

case '@':
    /* parentheses */
    category = CB_TREE_CATEGORY (x);
    break;

default:
    fprintf (stderr, "Unexpected operator -> %d\n", op);
    ABORT ();
}

p = make_tree (CB_TAG_BINARY_OP, category, sizeof (struct cb_binary_op));

p->op = op;
p->x = x;
p->y = y;
return CB_TREE (p);
}

```

#### 7.16.4.14 `cb_tree cb_build_call ( cb_tree name, cb_tree args, cb_tree stmt1, cb_tree stmt2, cb_tree returning, int is_system_call )`

Definition at line 2149 of file tree.c.

```

{
    struct cb_call *p;

```

```

    p = make_tree (CB_TAG_CALL, CB_CATEGORY_UNKNOWN, sizeof (struct cb_call))
;
    p->name = name;
    p->args = args;
    p->stmt1 = stmt1;
    p->stmt2 = stmt2;
    p->returning = returning;
    p->is_system = is_system_call;
    return CB_TREE (p);
}

```

#### 7.16.4.15 `cb_tree cb_build_cast ( enum cb_cast_type type, cb_tree val )`

Definition at line 2060 of file tree.c.

```

{
    struct cb_cast      *p;
    enum cb_category    category;

    if (type == CB_CAST_INTEGER) {
        category = CB_CATEGORY_NUMERIC;
    } else {
        category = CB_CATEGORY_UNKNOWN;
    }
    p = make_tree (CB_TAG_CAST, category, sizeof (struct cb_cast));
    p->type = type;
    p->val = val;
    return CB_TREE (p);
}

```

#### 7.16.4.16 `cb_tree cb_build_class_name ( cb_tree name, cb_tree list )`

Definition at line 939 of file tree.c.

```

{
    struct cb_class_name *p;
    char buff[COB_MINI_BUFF];

    p = make_tree (CB_TAG_CLASS_NAME, CB_CATEGORY_BOOLEAN, sizeof (struct
cb_class_name));
    p->name = cb_define (name, CB_TREE (p));
    snprintf (buff, COB_MINI_MAX, "is_%s", to_cname (p->name));
    p->cname = strdup (buff);
    p->list = list;
    return CB_TREE (p);
}

```

#### 7.16.4.17 `cb_tree cb_build_cond ( cb_tree x )`

Definition at line 2354 of file typeck.c.

```

{
    int                size1;
    int                size2;
    struct cb_field    *f;
    struct cb_binary_op *p;
    cb_tree            d1;
    cb_tree            d2;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_CONST:
    case CB_TAG_FUNCALL:
        return x;
    case CB_TAG_REFERENCE:
        if (!CB_FIELD_P (cb_ref (x))) {
            return cb_build_cond (cb_ref (x));
        }

        f = cb_field (x);

        /* level 88 condition */
        if (f->level == 88) {
            /* We need to build a 88 condition at every occurrence
               instead of once at the beginning because a 88 item
               may be subscripted (i.e., it is not a constant tree).
            */
            return cb_build_cond (build_cond_88 (x));
        }

        cb_error_x (x, _("Invalid expression"));
        return cb_error_node;
    case CB_TAG_BINARY_OP:
        p = CB_BINARY_OP (x);
        switch (p->op) {
        case '!':
            return cb_build_negation (cb_build_cond (p->x));
        case '&':
        case '|':
            return cb_build_binary_op (cb_build_cond (p->x), p->op,
cb_build_cond (p->y));
        default:
            if (CB_INDEX_P (p->x) || CB_INDEX_P (p->y)
                || CB_TREE_CLASS (p->x) == CB_CLASS_POINTER
                || CB_TREE_CLASS (p->y) == CB_CLASS_POINTER) {
                x = cb_build_binary_op (p->x, '-', p->y);
            } else if (CB_BINARY_OP_P (p->x) || CB_BINARY_OP_P (p->y)
) {
                /* decimal comparison */
                d1 = decimal_alloc ();
                d2 = decimal_alloc ();

                decimal_expand (d1, p->x);
                decimal_expand (d2, p->y);
                dpush (cb_build_funcall_2 ("cob_decimal_cmp", d1,
d2));

                decimal_free ();
                decimal_free ();
                x = cb_list_reverse (decimal_stack);
                decimal_stack = NULL;
            } else {
                if (cb_chk_num_cond (p->x, p->y)) {
                    size1 = cb_field_size (p->x);
                    x = cb_build_funcall_3 ("memcmp",

```

```

        cb_build_cast_address (p->x),
        cb_build_cast_address (p->y),
        cb_int (size1));
        break;
    }
    if (CB_TREE_CLASS (p->x) == CB_CLASS_NUMERIC
        && CB_TREE_CLASS (p->y) == CB_CLASS_NUMERIC
        && cb_fits_int (p->y)) {
        x = cb_build_optim_cond (p);
        break;
    }

    /* field comparison */
    if ((CB_REF_OR_FIELD_P (p->x))
        && (CB_TREE_CATEGORY (p->x) ==
CB_CATEGORY_ALPHANUMERIC ||
CB_CATEGORY_ALPHABETIC)
        && (CB_TREE_CATEGORY (p->x) ==
        && (cb_field_size (p->x) == 1)
        && (!current_program->alphabet_name_list)
        && (p->y == cb_space || p->y == cb_low ||
        p->y == cb_high || p->y == cb_zero)) {
        x = cb_build_funcall_2 ("SG", p->x, p->y)
;
        break;
    }
    if (cb_chk_alpha_cond (p->x) && cb_chk_alpha_cond
(p->y)) {
        size1 = cb_field_size (p->x);
        size2 = cb_field_size (p->y);
    } else {
        size1 = 0;
        size2 = 0;
    }
    if (size1 == 1 && size2 == 1) {
        x = cb_build_funcall_2 ("SG", p->x, p->y)
;
    } else if (size1 != 0 && size1 == size2) {
        x = cb_build_funcall_3 ("memcmp",
        cb_build_cast_address (p->x),
        cb_build_cast_address (p->y),
        cb_int (size1));
    } else {
        if (CB_TREE_CLASS (p->x) ==
CB_CLASS_NUMERIC && p->y == cb_zero) {
            x = cb_build_optim_cond (p);
        } else {
            x = cb_build_funcall_2 ("cob_cmp"
, p->x, p->y);
        }
    }
}
return cb_build_binary_op (x, p->op, p->y);
default:
    cb_error_x (x, _("Invalid expression"));
    return cb_error_node;
}
/* NOT REACHED */
return x;
}

```



7.16.4.18 `cb_tree cb_build_const_length ( cb_tree x )`

Definition at line 956 of file typeck.c.

```
{
    struct cb_field      *f;
    char                 buff[64];

    if (x == cb_error_node) {
        return cb_error_node;
    }
    if (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node) {
        return cb_error_node;
    }

    memset (buff, 0, sizeof (buff));
    f = CB_FIELD (cb_ref (x));
    if (f->flag_any_length) {
        cb_error (_("ANY LENGTH item not allowed here"));
        return cb_error_node;
    }
    if (f->level == 88) {
        cb_error (_("88 level item not allowed here"));
        return cb_error_node;
    }
    if (!f->flag_is_verified) {
        cb_validate_field (f);
    }
    sprintf (buff, "%d", f->memory_size);
    return cb_build_numeric_literal (0, (ucharptr)buff, 0);
}
```

7.16.4.19 `cb_tree cb_build_constant ( cb_tree name, cb_tree value )`

Definition at line 1446 of file tree.c.

```
{
    cb_tree x;

    x = cb_build_field (name);
    x->category = cb_tree_category (value);
    CB_FIELD (x)->storage = CB_STORAGE_CONSTANT;
    CB_FIELD (x)->values = cb_list_init (value);
    return x;
}
```

7.16.4.20 `cb_tree cb_build_continue ( void )`

Definition at line 2240 of file tree.c.

```
{
    struct cb_continue *p;

    p = make_tree (CB_TAG_CONTINUE, CB_CATEGORY_UNKNOWN, sizeof (struct
```

```
        cb_continue));
        return CB_TREE (p);
    }
```

#### 7.16.4.21 `cb_tree cb_build_converting ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3847 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_converting", x, y
));
}
```

#### 7.16.4.22 `cb_tree cb_build_decimal ( int id )`

Definition at line 1076 of file tree.c.

```
{
    struct cb_decimal *p;

    p = make_tree (CB_TAG_DECIMAL, CB_CATEGORY_NUMERIC, sizeof (struct
cb_decimal));
    p->id = id;
    return CB_TREE (p);
}
```

#### 7.16.4.23 `cb_tree cb_build_display_upon ( cb_tree x )`

Definition at line 3406 of file typeck.c.

```
{
    if (x == cb_error_node) {
        return cb_error_node;
    }

    switch (CB_SYSTEM_NAME (cb_ref (x))->token) {
    case CB_DEVICE_CONSOLE:
    case CB_DEVICE_SYSOUT:
        return cb_int0;
    case CB_DEVICE_SYSERR:
        return cb_int1;
    default:
        cb_error_x (x, _("Invalid output stream"));
        return cb_error_node;
    }
}
```

7.16.4.24 `cb_tree cb_build_display_upon_direct ( cb_tree x )`

Definition at line 3425 of file typeck.c.

```

{
    const char    *name;
    cb_tree      sys;

    if (x == cb_error_node) {
        return cb_error_node;
    }
    name = CB_NAME (x);
    if (CB_REFERENCE (x)->word->count == 0) {
        sys = lookup_system_name (CB_NAME (x));
        if (sys != cb_error_node) {
            switch (CB_SYSTEM_NAME (sys)->token) {
            case CB_DEVICE_CONSOLE:
            case CB_DEVICE_SYSOUT:
                cb_warning_x (x, _("%s' undefined in SPECIAL-NAM
ES"), name);
                return cb_int0;
            case CB_DEVICE_SYSERR:
                cb_warning_x (x, _("%s' undefined in SPECIAL-NAM
ES"), name);
                return cb_int1;
            default:
                break;
            }
        }
    }

    cb_error_x (x, _("%s' undefined in SPECIAL-NAMES"), name);
    return cb_error_node;
}

```

7.16.4.25 `cb_tree cb_build_expr ( cb_tree list )`

Definition at line 1820 of file typeck.c.

```

{
    cb_tree l;
    /* RXW
    cb_tree x;
    */
    int    op;

    cb_expr_init ();

    for (l = list; l; l = CB_CHAIN (l)) {
        op = CB_PURPOSE_INT (l);
        switch (op) {
            case '9': /* NUMERIC */
                cb_expr_shift_class ("cob_is_numeric");
                break;
            case 'A': /* ALPHABETIC */
                cb_expr_shift_class ("cob_is_alpha");
                break;
            case 'L': /* ALPHABETIC_LOWER */

```

```

        cb_expr_shift_class ("cob_is_lower");
        break;
    case 'U': /* ALPHABETIC_UPPER */
        cb_expr_shift_class ("cob_is_upper");
        break;
    case 'P': /* POSITIVE */
        cb_expr_shift_sign ('>');
        break;
    case 'N': /* NEGATIVE */
        cb_expr_shift_sign ('<');
        break;
    case 'O': /* OMITTED */
        current_statement->>null_check = NULL;
        cb_expr_shift_class ("cob_is_omitted");
        break;
/* RXW
    case 'x':
        if (CB_VALUE (1) && CB_REFERENCE_P (CB_VALUE (1))) {
            x = CB_CHAIN (1);
            if (x && cb_field (CB_VALUE (1))->level == 88) {
                switch (CB_PURPOSE_INT (x)) {
                    case '&':
                    case '|':
                    case '(':
                    case ')':
                        break;
                    default:
                        cb_error (_("Invalid condition"));
                }
                break;
            }
        }
        cb_expr_shift (op, CB_VALUE (1));
        break;
*/
    default:
        cb_expr_shift (op, CB_VALUE (1));
        break;
}

return cb_expr_finish ();
}

```

#### 7.16.4.26 `cb_tree cb_build_field ( cb_tree name )`

Definition at line 1417 of file tree.c.

```

{
    struct cb_field *p;

    p = make_tree (CB_TAG_FIELD, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_field));
    p->id = cb_field_id++;
    p->name = cb_define (name, CB_TREE (p));
    p->ename = NULL;
    p->usage = CB_USAGE_DISPLAY;
    p->storage = CB_STORAGE_WORKING;
}

```

```

    p->occurs_max = 1;
    return CB_TREE (p);
}

```

#### 7.16.4.27 `cb_tree cb_build_field_reference ( struct cb_field * f, cb_tree ref )`

Definition at line 1752 of file tree.c.

```

{
    cb_tree      x;
    struct cb_word *word;

    x = cb_build_reference (f->name);
    word = CB_REFERENCE (x)->word;
    if (ref) {
        memcpy (x, ref, sizeof (struct cb_reference));
    }
    x->category = CB_CATEGORY_UNKNOWN;
    CB_REFERENCE (x)->word = word;
    CB_REFERENCE (x)->value = CB_TREE (f);
    return x;
}

```

#### 7.16.4.28 `cb_tree cb_build_field_tree ( cb_tree level, cb_tree name, struct cb_field * last_field, enum cb_storage storage, struct cb_file * fn )`

Definition at line 78 of file field.c.

```

{
    struct cb_reference *r;
    struct cb_field *f;
    struct cb_field *p;
    struct cb_field *field_fill;
    cb_tree dummy_fill;
    cb_tree l;
    cb_tree x;
    int lv;

    if (level == cb_error_node || name == cb_error_node) {
        return cb_error_node;
    }

    /* check the level number */
    lv = cb_get_level (level);
    if (!lv) {
        return cb_error_node;
    }

    /* build the field */
    r = CB_REFERENCE (name);
    f = CB_FIELD (cb_build_field (name));
    f->storage = storage;
    last_real_field = last_field;
    if (lv == 78) {
        f->level = 01;
    }
}

```

```

        f->flag_item_78 = 1;
        return CB_TREE (f);
    } else {
        f->level = lv;
    }
}
if (f->level == 01 && storage == CB_STORAGE_FILE) {
    if (fn->external) {
        f->flag_external = 1;
        has_external = 1;
    } else if (fn->global) {
        f->flag_is_global = 1;
    }
}
if (last_field) {
    if (last_field->level == 77 && f->level != 01 &&
        f->level != 77 && f->level != 66 && f->level != 88) {
        cb_error_x (name, _("Level number must begin with 01 or 7
7"));
        return cb_error_node;
    }
}

/* checks for redefinition */
if (cb_warn_redefinition) {
    if (r->word->count > 1) {
        if (f->level == 01 || f->level == 77) {
            redefinition_warning (name, NULL);
        } else {
            for (l = r->word->items; l; l = CB_CHAIN (l)) {
                x = CB_VALUE (l);
                if (!CB_FIELD_P (x)
                    || CB_FIELD (x)->level == 01
                    || CB_FIELD (x)->level == 77
                    || (f->level == last_field->level
                        && CB_FIELD (x)->parent == last_f
ield->parent)) {
                    redefinition_warning (name, x);
                    break;
                }
            }
        }
    }
}

if (last_field && last_field->level == 88) {
    last_field = last_field->parent;
}

/* link the field into the tree */
if (f->level == 01 || f->level == 77) {
    /* top level */
    cb_needs_01 = 0;
    if (last_field) {
        /*
            cb_field_add (cb_field_founder (last_field), f);
        */
        cb_field_founder (last_field)->sister = f;
    }
} else if (!last_field || cb_needs_01) {
    /* invalid top level */
    cb_error_x (name, _("Level number must begin with 01 or 77"));
    return cb_error_node;
}

```

```

    } else if (f->level == 66) {
        /* level 66 */
        f->parent = cb_field_founder (last_field);
        for (p = f->parent->children; p && p->sister; p = p->sister) ;
        if (p) {
            p->sister = f;
        }
    } else if (f->level == 88) {
        /* level 88 */
        f->parent = last_field;
    } else if (f->level > last_field->level) {
        /* lower level */
        last_field->children = f;
        f->parent = last_field;
    } else if (f->level == last_field->level) {
        /* same level */
same_level:
        last_field->sister = f;
        f->parent = last_field->parent;
    } else {
        /* upper level */
        for (p = last_field->parent; p; p = p->parent) {
            if (p->level == f->level) {
                last_field = p;
                goto same_level;
            }
            if (cb_relax_level_hierarchy && p->level < f->level) {
                break;
            }
        }
        if (cb_relax_level_hierarchy) {
            dummy_fill = cb_build_filler ();
            field_fill = CB_FIELD (cb_build_field (dummy_fill));
            cb_warning_x (name, _("No previous data item of level %02
d"), f->level);
            field_fill->level = f->level;
            field_fill->storage = storage;
            field_fill->children = p->children;
            field_fill->parent = p;
            for (p = p->children; p != NULL; p = p->sister) {
                p->parent = field_fill;
            }
            field_fill->parent->children = field_fill;
            field_fill->sister = f;
            f->parent = field_fill->parent;
            last_field = field_fill;
        } else {
            cb_error_x (name, _("No previous data item of level %02d"
), f->level);
            return cb_error_node;
        }
    }
}

/* inherit parent's properties */
if (f->parent) {
    f->usage = f->parent->usage;
    f->indexes = f->parent->indexes;
    f->flag_sign_leading = f->parent->flag_sign_leading;
    f->flag_sign_separate = f->parent->flag_sign_separate;
    f->flag_is_global = f->parent->flag_is_global;
}
return CB_TREE (f);

```

```
}

```

#### 7.16.4.29 `cb_tree cb_build_filler ( void )`

Definition at line 1740 of file tree.c.

```
{
    cb_tree      x;
    char         name[16];

    sprintf (name, "WORK$%d", filler_id++);
    x = cb_build_reference (name);
    x->source_line = cb_source_line;
    return x;
}
```

#### 7.16.4.30 `cb_tree cb_build_funcall ( const char * name, int argc, cb_tree a1, cb_tree a2, cb_tree a3, cb_tree a4, cb_tree a5, cb_tree a6, cb_tree a7 )`

Definition at line 2035 of file tree.c.

```
{
    struct cb_funcall *p;

    p = make_tree (CB_TAG_FUNCALL, CB_CATEGORY_BOOLEAN, sizeof (struct
cb_funcall));
    p->name = name;
    p->argc = argc;
    p->varcnt = 0;
    p->screenptr = gen_screen_ptr;
    p->argv[0] = a1;
    p->argv[1] = a2;
    p->argv[2] = a3;
    p->argv[3] = a4;
    p->argv[4] = a5;
    p->argv[5] = a6;
    p->argv[6] = a7;
    return CB_TREE (p);
}
```

#### 7.16.4.31 `cb_tree cb_build_goto ( cb_tree target, cb_tree depending )`

Definition at line 2168 of file tree.c.

```
{
    struct cb_goto *p;

    p = make_tree (CB_TAG_GOTO, CB_CATEGORY_UNKNOWN, sizeof (struct cb_goto)
);
    p->target = target;
    p->depending = depending;
    return CB_TREE (p);
}
```



7.16.4.32 `cb_tree cb_build_identifier ( cb_tree x )`

Definition at line 763 of file typeck.c.

```

{
    struct cb_reference    *r;
    struct cb_field       *f;
    struct cb_field       *p;
    const char            *name;
    cb_tree               v;
    cb_tree               e1;
    cb_tree               e2;
    cb_tree               l;
    cb_tree               sub;
    int                   offset;
    int                   length;
    int                   n;

    if (x == cb_error_node) {
        return cb_error_node;
    }

    r = CB_REFERENCE (x);
    name = r->word->name;

    /* resolve reference */
    v = cb_ref (x);
    if (v == cb_error_node) {
        return cb_error_node;
    }

    /* check if it is a data name */
    if (!CB_FIELD_P (v)) {
        if (r->subs) {
            cb_error_x (x, _("%s' cannot be subscripted"), name);
            return cb_error_node;
        }
        if (r->offset) {
            cb_error_x (x, _("%s' cannot be reference modified"), name);
            return cb_error_node;
        }
        return x;
    }
    f = CB_FIELD (v);

    /* BASED check */
    if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_PTR)) {
        for (p = f; p->parent; p = p->parent) {
            ;
        }
        if (current_statement) {
            if (p->flag_item_based ||
                (f->storage == CB_STORAGE_LINKAGE &&
                 !p->flag_is_pdiv_parm)) {
                current_statement->null_check =
cb_build_funcall_2 (
                                "cob_check_based",
                                cb_build_address (
cb_build_field_reference (p, NULL)),
                                cb_build_string0 ((ucharptr)name));
            }
        }
    }
}

```

```

    }
}

/* check the number of subscripts */
if (!r->all && cb_list_length (r->subs) != f->indexes) {
    switch (f->indexes) {
        case 0:
            cb_error_x (x, _("%s' cannot be subscripted"), name);
            return cb_error_node;
        case 1:
            cb_error_x (x, _("%s' requires 1 subscript"), name);
            return cb_error_node;
        default:
            cb_error_x (x, _("%s' requires %d subscripts"), name, f-
>indexes);
            return cb_error_node;
    }
}

/* subscript check */
if (!r->all && r->subs) {
    l = r->subs;
    for (p = f; p; p = p->parent) {
        if (p->flag_occurs) {
            sub = cb_check_integer_value (CB_VALUE (l));

            l = CB_CHAIN (l);

            if (sub == cb_error_node) {
                continue;
            }

            /* compile-time check */
            if (CB_LITERAL_P (sub)) {
                n = cb_get_int (sub);
                if (n < 1 || n > p->occurs_max) {
                    cb_error_x (x, _("Subscript of '%s
s' out of bounds: %d"),
                                name, n);
                }
            }

            /* run-time check */
            if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_SUBSCRIPT))
            {
                if (p->occursDepending) {
                    e1 = cb_build_funcall_4 ("cob_che
ck_odo",
                                cb_build_cast_integer (p
->occursDepending),
                                cb_int (p->occurs_min),
                                cb_int (p->occurs_max),
                                cb_build_string0
                                ((ucharptr) (cb_field (p-
>occursDepending)->name)));
                    e2 = cb_build_funcall_4 ("cob_che
ck_subscript",
                                cb_build_cast_integer (s
ub),
                                cb_int1,
                                cb_build_cast_integer (p
->occursDepending),

```

```

                                cb_build_string0 ((
ucharptr)name));
                                r->check = cb_list_add (r->check,
e1);
                                r->check = cb_list_add (r->check,
e2);
                                } else {
                                    if (!CB_LITERAL_P (sub)) {
                                        e1 = cb_build_funcall_4 (
"cob_check_subscript",
                                cb_build_cast_integer (sub),
                                                cb_int1,
                                                cb_int (p->
occurs_max),
                                                cb_build_string0
((ucharptr)name));
                                r->check = cb_list_add (r
->check, e1);
                                    }
                                }
                            }
                    }
}

/* reference modification check */
if (r->offset) {
    /* compile-time check */
    if (CB_LITERAL_P (r->offset)) {
        offset = cb_get_int (r->offset);
        if (offset < 1 || offset > f->size) {
            cb_error_x (x, _("Offset of '%s' out of bounds: %
d"), name, offset);
        } else if (r->length && CB_LITERAL_P (r->length)) {
            length = cb_get_int (r->length);
            if (length < 1 || length > f->size - offset + 1)
{
                cb_error_x (x, _("Length of '%s' out of b
ounds: %d"),
                            name, length);
            }
        }
    }

    /* run-time check */
    if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_REF_MOD)) {
        if (!CB_LITERAL_P (r->offset)
            || (r->length && !CB_LITERAL_P (r->length))) {
            e1 = cb_build_funcall_4 ("cob_check_ref_mod",
                cb_build_cast_integer (r
->offset),
                r->length ?
cb_build_cast_integer (r->length) :
                cb_int1, cb_int (f->
size),
                cb_build_string0 ((
ucharptr)f->name));
            r->check = cb_list_add (r->check, e1);
        }
    }
}

```

```

    if (f->storage == CB_STORAGE_CONSTANT) {
        return CB_VALUE (f->values);
    }

    return x;
}

```

#### 7.16.4.33 **cb\_tree** **cb\_build\_if** ( **cb\_tree** *test*, **cb\_tree** *stmt1*, **cb\_tree** *stmt2* )

Definition at line 2183 of file tree.c.

```

{
    struct cb_if *p;

    p = make_tree (CB_TAG_IF, CB_CATEGORY_UNKNOWN, sizeof (struct cb_if));
    p->test = test;
    p->stmt1 = stmt1;
    p->stmt2 = stmt2;
    return CB_TREE (p);
}

```

#### 7.16.4.34 **cb\_tree** **cb\_build\_implicit\_field** ( **cb\_tree** *name*, **int** *len* )

Definition at line 1432 of file tree.c.

```

{
    cb_tree x;
    char pic[32];

    x = cb_build_field (name);
    memset (pic, 0, sizeof(pic));
    sprintf (pic, "X(%d)", len);
    CB_FIELD (x)->pic = CB_PICTURE (cb_build_picture (pic));
    cb_validate_field (CB_FIELD (x));
    return x;
}

```

#### 7.16.4.35 **cb\_tree** **cb\_build\_index** ( **cb\_tree** *name*, **cb\_tree** *values*, **int** *indexed\_by*, **struct** **cb\_field** \* *qual* )

Definition at line 744 of file typeck.c.

```

{
    struct cb_field *f;

    f = CB_FIELD (cb_build_field (x));
    f->usage = CB_USAGE_INDEX;
    cb_validate_field (f);
    if (values) {
        f->values = cb_list_init (values);
    }
}

```

```

    }
    if (qual) {
        f->index_qual = qual;
    }
    f->flag_indexed_by = indexed_by;
    current_program->working_storage = cb_field_add (current_program->
working_storage, f);
    return x;
}

```

#### 7.16.4.36 `cb_tree cb_build_initialize ( cb_tree var, cb_tree val, cb_tree rep, cb_tree def, int flag )`

Definition at line 2113 of file tree.c.

```

{
    struct cb_initialize *p;

    p = make_tree (CB_TAG_INITIALIZE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_initialize));
    p->var = var;
    p->val = val;
    p->rep = rep;
    p->def = def;
    p->flag_statement = flag;
    return CB_TREE (p);
}

```

#### 7.16.4.37 `cb_tree cb_build_inspect_region ( cb_tree l, cb_tree pos, cb_tree x )`

Definition at line 3859 of file typeck.c.

```

{
    if (pos == CB_BEFORE) {
        return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_before",
x));
    } else {
        return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_after", x
));
    }
}

```

#### 7.16.4.38 `cb_tree cb_build_inspect_region_start ( void )`

Definition at line 3853 of file typeck.c.

```

{
    return cb_list_init (cb_build_funcall_0 ("cob_inspect_start"));
}

```

7.16.4.39 `cb_tree cb_build_intrinsic ( cb_tree name, cb_tree args, cb_tree refmod )`

Definition at line 2262 of file tree.c.

```

{
    struct cb_intrinsic_table      *cbp;
    cb_tree                        x;
    int                            numargs;

    numargs = cb_list_length (args);

    cbp = lookup_intrinsic (CB_NAME (name), 0);
    if (cbp) {
        if ((cbp->args != -1 && numargs != cbp->args) ||
            (cbp->args == -1 && cbp->intr_enum != CB_INTR_RANDOM && numar
gs < 1)) {
            cb_error_x (name, _("FUNCTION %s has wrong number of argu
ments"), cbp->name);
            return cb_error_node;
        }
        if (refmod) {
            if (!cbp->refmod) {
                cb_error_x (name, _("FUNCTION %s can not have ref
erence modification"), cbp->name);
                return cb_error_node;
            }
            if (CB_LITERAL_P(CB_PAIR_X(refmod)) &&
                cb_get_int (CB_PAIR_X(refmod)) < 1) {
                cb_error_x (name, _("FUNCTION %s has invalid refe
rence modification"), cbp->name);
                return cb_error_node;
            }
            if (CB_PAIR_Y(refmod) && CB_LITERAL_P(CB_PAIR_Y(refmod))
&&
                cb_get_int (CB_PAIR_Y(refmod)) < 1) {
                cb_error_x (name, _("FUNCTION %s has invalid refe
rence modification"), cbp->name);
                return cb_error_node;
            }
        }
        /* cb_tree      x; */
        switch (cbp->intr_enum) {
        case CB_INTR_LENGTH:
        case CB_INTR_BYTE_LENGTH:
            x = CB_VALUE (args);
            if (CB_INTRINSIC_P (x)) {
                return make_intrinsic (name, cbp, args, NULL,
NULL);
            }
            } else if ((CB_FIELD_P (x) || CB_REFERENCE_P (x)) &&
                cb_field(x)->flag_any_length) {
                return make_intrinsic (name, cbp, args, NULL,
NULL);
            }
            } else {
                return cb_build_length (CB_VALUE (args));
            }
        }
        case CB_INTR_WHEN_COMPILED:
            if (refmod) {
                return make_intrinsic (name, cbp,
                    cb_list_init (cb_intr_whencomp), NULL, re
fmod);
            }
            } else {

```

```

        return cb_intr_whencomp;
    }
    case CB_INTR_PI:
        return cb_intr_pi;
    case CB_INTR_E:
        return cb_intr_e;

    case CB_INTR_LOWER_CASE:
    case CB_INTR_UPPER_CASE:
    case CB_INTR_REVERSE:
/* RXW Why did I do this ? - still do not know
        if (CB_INTRINSIC_P (CB_VALUE (args))) {
            return make_intrinsic (name, cbp, args, cb_int0);

        } else {
            return make_intrinsic (name, cbp, args,
                                   cb_build_length (CB_VALUE
(Args)));
        }
RXW */

    case CB_INTR_ABS:
    case CB_INTR_ACOS:
    case CB_INTR_ANNUIITY:
    case CB_INTR_ASIN:
    case CB_INTR_ATAN:
    case CB_INTR_CHAR:
    case CB_INTR_COMBINED_DATETIME:
    case CB_INTR_COS:
    case CB_INTR_CURRENT_DATE:
    case CB_INTR_DATE_OF_INTEGER:
    case CB_INTR_DAY_OF_INTEGER:
    case CB_INTR_EXCEPTION_FILE:
    case CB_INTR_EXCEPTION_LOCATION:
    case CB_INTR_EXCEPTION_STATUS:
    case CB_INTR_EXCEPTION_STATEMENT:
    case CB_INTR_EXP:
    case CB_INTR_EXP10:
    case CB_INTR_FACTORIAL:
    case CB_INTR_FRACTION_PART:
    case CB_INTR_INTEGER:
    case CB_INTR_INTEGER_OF_DATE:
    case CB_INTR_INTEGER_OF_DAY:
    case CB_INTR_INTEGER_PART:
    case CB_INTR_LOCALE_DATE:
    case CB_INTR_LOCALE_TIME:
    case CB_INTR_LOCALE_TIME_FROM_SECS:
    case CB_INTR_LOG:
    case CB_INTR_LOG10:
    case CB_INTR_MOD:
    case CB_INTR_NUMVAL:
    case CB_INTR_NUMVAL_C:
    case CB_INTR_ORD:
    case CB_INTR_REM:
    case CB_INTR_SECONDS_FROM_FORMATTED_TIME:
    case CB_INTR_SECONDS_PAST_MIDNIGHT:
    case CB_INTR_SIGN:
    case CB_INTR_SIN:
    case CB_INTR_SQRT:
    case CB_INTR_STORED_CHAR_LENGTH:
    case CB_INTR_TAN:
    case CB_INTR_TEST_DATE_YYYYMMDD:

```

```

case CB_INTR_TEST_DAY_YYYYDDD:
case CB_INTR_TRIM:
    return make_intrinsic (name, cbp, args, NULL, refmod);

case CB_INTR_CONCATENATE:
    return make_intrinsic (name, cbp, args, cb_intl, refmod);

case CB_INTR_DATE_TO_YYYYMMDD:
case CB_INTR_DAY_TO_YYYYDDD:
case CB_INTR_MAX:
case CB_INTR_MEAN:
case CB_INTR_MEDIAN:
case CB_INTR_MIDRANGE:
case CB_INTR_MIN:
case CB_INTR_ORD_MAX:
case CB_INTR_ORD_MIN:
case CB_INTR_PRESENT_VALUE:
case CB_INTR_RANDOM:
case CB_INTR_RANGE:
case CB_INTR_STANDARD_DEVIATION:
case CB_INTR_SUM:
case CB_INTR_VARIANCE:
case CB_INTR_YEAR_TO_YYYY:
    return make_intrinsic (name, cbp, args, cb_intl, NULL);
case CB_INTR_SUBSTITUTE:
case CB_INTR_SUBSTITUTE_CASE:
    if (numargs < 3 || (numargs % 2) == 0) {
        cb_error_x (name, _("FUNCTION %s has wrong number
of arguments"), cbp->name);
        return cb_error_node;
    }
    return make_intrinsic (name, cbp, args, cb_intl, refmod);

default:
    break;
}
}
cb_error_x (name, _("FUNCTION %s not implemented"), CB_NAME (name));
return cb_error_node;
}

```

#### 7.16.4.40 `cb_tree cb_build_label ( cb_tree name, struct cb_label * section )`

Definition at line 2081 of file tree.c.

```

{
    struct cb_label *p;

    p = make_tree (CB_TAG_LABEL, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_label));
    p->id = cb_id++;
    p->name = (const unsigned char *)cb_define (name, CB_TREE (p));
    p->orig_name = p->name;
    p->section = section;
    return CB_TREE (p);
}

```



7.16.4.41 `cb_tree cb_build_length ( cb_tree x )`

Definition at line 986 of file typeck.c.

```

{
    struct cb_field      *f;
    struct cb_literal    *l;
    cb_tree              temp;
    char                 buff[64];

    if (x == cb_error_node) {
        return cb_error_node;
    }
    if (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node) {
        return cb_error_node;
    }

    memset (buff, 0, sizeof (buff));
    if (CB_LITERAL_P (x)) {
        l = CB_LITERAL (x);
        sprintf (buff, "%d", (int)l->size);
        return cb_build_numeric_literal (0, (ucharptr)buff, 0);
    }
    if (CB_REF_OR_FIELD_P (x)) {
        f = CB_FIELD (cb_ref (x));
        if (f->flag_any_length) {
            return cb_build_any_intrinsic (cb_list_init (x));
        }
        if (cb_field_variable_size (f) == NULL) {
            sprintf (buff, "%d", cb_field_size (x));
            return cb_build_numeric_literal (0, (ucharptr)buff, 0);
        }
    }
    if (CB_INTRINSIC_P (x)) {
        return cb_build_any_intrinsic (cb_list_init (x));
    }
    temp = cb_build_index (cb_build_filler (), NULL, 0, NULL);
    CB_FIELD (cb_ref (temp))->usage = CB_USAGE_LENGTH;
    CB_FIELD (cb_ref (temp))->count++;
    cb_emit (cb_build_assign (temp, cb_build_length_1 (x)));
    return temp;
}

```

7.16.4.42 `cb_tree cb_build_list ( cb_tree purpose, cb_tree value, cb_tree rest )`

Definition at line 769 of file tree.c.

```

{
    struct cb_list *p;

    p = make_tree (CB_TAG_LIST, CB_CATEGORY_UNKNOWN, sizeof (struct cb_list))
;
    p->purpose = purpose;
    p->value = value;
    p->chain = rest;
    return CB_TREE (p);
}

```

7.16.4.43 `cb_tree cb_build_locale_name ( cb_tree name, cb_tree list )`

Definition at line 957 of file tree.c.

```

{
    struct cb_class_name    *p;

    p = make_tree (CB_TAG_LOCALE_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_locale_name));
    p->name = cb_define (name, CB_TREE (p));
    p->cname = to_cname (p->name);
    p->list = list;
    return CB_TREE (p);
}

```

7.16.4.44 `cb_tree cb_build_move ( cb_tree src, cb_tree dst )`

Definition at line 4964 of file typeck.c.

```

{
    struct cb_field *f;
    struct cb_field *p;

    if (src == cb_error_node || dst == cb_error_node) {
        return cb_error_node;
    }

    if (validate_move (src, dst, 0) < 0) {
        return cb_error_node;
    }

    if (CB_REFERENCE_P (src)) {
        CB_REFERENCE (src)->type = CB_SENDING_OPERAND;
    }
    if (CB_REFERENCE_P (dst)) {
        CB_REFERENCE (dst)->type = CB_RECEIVING_OPERAND;
    }

    if (CB_TREE_CLASS (dst) == CB_CLASS_POINTER) {
        return cb_build_assign (dst, src);
    }

    if (CB_REFERENCE_P (src) && CB_ALPHABET_NAME_P (CB_REFERENCE (src)->value))
    {
        return cb_build_move_call (src, dst);
    }
    if (CB_INDEX_P (dst)) {
        if (src == cb_null) {
            return cb_build_assign (dst, cb_zero);
        }
        return cb_build_assign (dst, src);
    }

    if (CB_INDEX_P (src)) {
        return cb_build_funcall_2 ("cob_set_int", dst,
cb_build_cast_integer (src));
    }
}

```

```

    if (CB_INTRINSIC_P (src) || CB_INTRINSIC_P (dst)) {
        return cb_build_move_call (src, dst);
    }

    f = cb_field (dst);

    if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_SUBSCRIPT)) {
        for (p = f; p; p = p->parent) {
            if (p->flag_occurs) {
                return cb_build_move_call (src, dst);
            }
        }
        if (CB_REF_OR_FIELD_P (src)) {
            for (p = cb_field (src); p; p = p->parent) {
                if (p->flag_occurs) {
                    return cb_build_move_call (src, dst);
                }
            }
        }
    }

    /* output optimal code */
    if (src == cb_zero) {
        return cb_build_move_zero (dst);
    } else if (src == cb_space) {
        return cb_build_move_space (dst);
    } else if (src == cb_high) {
        return cb_build_move_high (dst);
    } else if (src == cb_low) {
        return cb_build_move_low (dst);
    } else if (src == cb_quote) {
        return cb_build_move_quote (dst);
    } else if (CB_LITERAL_P (src)) {
        return cb_build_move_literal (src, dst);
    }
    return cb_build_move_field (src, dst);
}

```

#### 7.16.4.45 `cb_tree cb_build_numeric_literal ( int sign, const unsigned char * data, int scale )`

Definition at line 988 of file tree.c.

```

{
    struct cb_literal *p;

    p = build_literal (CB_CATEGORY_NUMERIC, data, strlen ((char *)data));
    p->sign = (char)sign;
    p->scale = (char)scale;
    return CB_TREE (p);
}

```

#### 7.16.4.46 `cb_tree cb_build_perform ( int type )`

Definition at line 2199 of file tree.c.

```

{

```

```

    struct cb_perform *p;

    p = make_tree (CB_TAG_PERFORM, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_perform));
    p->type = type;
    return CB_TREE (p);
}

```

#### 7.16.4.47 `cb_tree cb_build_perform_exit ( struct cb_label * label )`

Definition at line 5156 of file typeck.c.

```

{
    cb_tree x;

    x = cb_build_perform (CB_PERFORM_EXIT);
    CB_PERFORM (x)->data = CB_TREE (label);
    return x;
}

```

#### 7.16.4.48 `cb_tree cb_build_perform_forever ( cb_tree body )`

Definition at line 5143 of file typeck.c.

```

{
    cb_tree x;

    if (body == cb_error_node) {
        return cb_error_node;
    }
    x = cb_build_perform (CB_PERFORM_FOREVER);
    CB_PERFORM (x)->body = body;
    return x;
}

```

#### 7.16.4.49 `cb_tree cb_build_perform_once ( cb_tree body )`

Definition at line 5105 of file typeck.c.

```

{
    cb_tree x;

    if (body == cb_error_node) {
        return cb_error_node;
    }
    x = cb_build_perform (CB_PERFORM_ONCE);
    CB_PERFORM (x)->body = body;
    return x;
}

```

**7.16.4.50 `cb_tree` `cb_build_perform_times` ( `cb_tree count` )**

Definition at line 5118 of file typeck.c.

```
{
    cb_tree x;

    if (cb_check_integer_value (times) == cb_error_node) {
        return cb_error_node;
    }

    x = cb_build_perform (CB_PERFORM_TIMES);
    CB_PERFORM (x)->data = times;
    return x;
}
```

**7.16.4.51 `cb_tree` `cb_build_perform_until` ( `cb_tree condition`, `cb_tree varying` )**

Definition at line 5132 of file typeck.c.

```
{
    cb_tree x;

    x = cb_build_perform (CB_PERFORM_UNTIL);
    CB_PERFORM (x)->test = condition;
    CB_PERFORM (x)->varying = varying;
    return x;
}
```

**7.16.4.52 `cb_tree` `cb_build_perform_varying` ( `cb_tree name`, `cb_tree from`, `cb_tree step`, `cb_tree until` )**

Definition at line 2209 of file tree.c.

```
{
    struct cb_perform_varying *p;

    p = make_tree (CB_TAG_PERFORM_VARYING, CB_CATEGORY_UNKNOWN, sizeof (struct
    cb_perform_varying));
    p->name = name;
    p->from = from;
    p->step = name ? cb_build_add (name, by, cb_high) : NULL;
    p->until = until;
    return CB_TREE (p);
}
```

**7.16.4.53 `cb_tree` `cb_build_picture` ( `const char * str` )**

Definition at line 1090 of file tree.c.

```

{
    struct cb_picture      *pic;
    const char             *p;
    size_t                 idx = 0;
    size_t                 buffcnt = 0;
    size_t                 at_beginning;
    size_t                 at_end;
    size_t                 p_char_seen;
    size_t                 s_char_seen;
    int                    category = 0;
    int                    size = 0;
    int                    allocated = 0;
    int                    digits = 0;
    int                    scale = 0;
    int                    s_count = 0;
    int                    v_count = 0;
    int                    i;
    int                    n;
    unsigned char          c;
    unsigned char          lastonechar = 0;
    unsigned char          lasttwochar = 0;
    unsigned char          buff[COB_SMALL_BUFF];

    pic = make_tree (CB_TAG_PICTURE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_picture));
    if (strlen (str) > 50) {
        goto error;
    }
    memset (buff, 0, sizeof (buff));
    p_char_seen = 0;
    s_char_seen = 0;
    for (p = str; *p; p++) {
repeat:
        n = 1;
        c = *p;

        /* count the number of repeated chars */
        while (p[1] == c) {
            p++, n++;
        }

        /* add parenthesized numbers */
        if (p[1] == '(') {
            i = 0;
            p += 2;
            for (; *p == '0'; p++) {
                ;
            }
            for (; *p != ')'; p++) {
                if (!isdigit (*p)) {
                    goto error;
                } else {
                    allocated++;
                    if (allocated > 9) {
                        goto error;
                    }
                    i = i * 10 + (*p - '0');
                }
            }
            if (i == 0) {
                goto error;
            }
            n += i - 1;

```

```
        goto repeat;
    }

    /* check grammar and category */
    /* FIXME: need more error check */
    switch (c) {
    case 'A':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_ALPHABETIC;
        break;

    case 'X':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_ALPHANUMERIC;
        break;

    case '9':
        category |= PIC_NUMERIC;
        digits += n;
        if (v_count) {
            scale += n;
        }
        break;

    case 'N':
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        category |= PIC_NATIONAL;
        break;

    case 'S':
        category |= PIC_NUMERIC;
        if (category & PIC_ALPHABETIC) {
            goto error;
        }
        s_count += n;
        if (s_count > 1 || idx != 0) {
            goto error;
        }
        s_char_seen = 1;
        continue;

    case ',':
    case '.':
        category |= PIC_NUMERIC_EDITED;
        if (s_char_seen || p_char_seen) {
            goto error;
        }
        if (c != current_program->decimal_point) {
            break;
        }
        /* fall through */
    case 'V':
        category |= PIC_NUMERIC;
        if (category & PIC_ALPHABETIC) {
            goto error;
        }
    }
```

```

        v_count += n;
        if (v_count > 1) {
            goto error;
        }
        break;

case 'P':
    category |= PIC_NUMERIC;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (p_char_seen) {
        goto error;
    }
    at_beginning = 0;
    at_end = 0;
    switch (buffcnt) {
    case 0:
        /* P..... */
        at_beginning = 1;
        break;
    case 1:
        /* VP.... */
        /* SP.... */
        if (lastonechar == 'V' || lastonechar == 'S') {
            at_beginning = 1;
        }
        break;
    case 2:
        /* SVP... */
        if (lasttwochar == 'S' && lastonechar == 'V') {
            at_beginning = 1;
        }
        break;
    }
    if (p[1] == 0 || (p[1] == 'V' && p[2] == 0)) {
        /* .....P */
        /* ....PV */
        at_end = 1;
    }
    if (!at_beginning && !at_end) {
        goto error;
    }
    p_char_seen = 1;
    if (at_beginning) {
        v_count++;      /* implicit V */
    }
    digits += n;
    if (v_count) {
        scale += n;
    } else {
        scale -= n;
    }
    break;

case '0':
case 'B':
case '/':
    category |= PIC_EDITED;
    if (s_char_seen || p_char_seen) {
        goto error;
    }
}

```



```
        break;

case '*':
case 'Z':
    category |= PIC_NUMERIC_EDITED;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    digits += n;
    if (v_count) {
        scale += n;
    }
    break;

case '+':
case '-':
    category |= PIC_NUMERIC_EDITED;
    if (category & PIC_ALPHABETIC) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    digits += n - 1;
    s_count++;
    /* FIXME: need more check */
    break;

case 'C':
    category |= PIC_NUMERIC_EDITED;
    if (!(p[1] == 'R' && p[2] == 0)) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    p++;
    s_count++;
    break;

case 'D':
    category |= PIC_NUMERIC_EDITED;
    if (!(p[1] == 'B' && p[2] == 0)) {
        goto error;
    }
    if (s_char_seen || p_char_seen) {
        goto error;
    }
    p++;
    s_count++;
    break;

default:
    if (c == current_program->currency_symbol) {
        category |= PIC_NUMERIC_EDITED;
        digits += n - 1;
        /* FIXME: need more check */
        break;
    }
}
```

```

        goto error;
    }

    /* calculate size */
    if (c != 'V' && c != 'P') {
        size += n;
    }
    if (c == 'C' || c == 'D' || c == 'N') {
        size += n;
    }

    /* store in the buffer */
    buff[idx++] = c;
    lasttwochar = lastonechar;
    lastonechar = c;
    memcpy (&buff[idx], (unsigned char *)&n, sizeof(int));
    idx += sizeof(int);
    ++buffcnt;
}
buff[idx] = 0;

if (size == 0 && v_count) {
    goto error;
}
/* set picture */
pic->orig = strdup (str);
pic->size = size;
pic->digits = (unsigned char)digits;
pic->scale = (signed char)scale;
pic->have_sign = (unsigned char)s_count;

/* set picture category */
switch (category) {
case PIC_ALPHABETIC:
    pic->category = CB_CATEGORY_ALPHABETIC;
    break;
case PIC_NUMERIC:
    pic->category = CB_CATEGORY_NUMERIC;
    if (digits > 36) {
        cb_error (_("Numeric field cannot be larger than 36 digit
s"));
    }
    break;
case PIC_ALPHANUMERIC:
case PIC_NATIONAL:
    pic->category = CB_CATEGORY_ALPHANUMERIC;
    break;
case PIC_NUMERIC_EDITED:
    pic->str = cobc_malloc (idx + 1);
    memcpy (pic->str, buff, idx);
    pic->category = CB_CATEGORY_NUMERIC_EDITED;
    pic->lenstr = idx;
    break;
case PIC_EDITED:
case PIC_ALPHABETIC_EDITED:
case PIC_ALPHANUMERIC_EDITED:
case PIC_NATIONAL_EDITED:
    pic->str = cobc_malloc (idx + 1);
    memcpy (pic->str, buff, idx);
    pic->category = CB_CATEGORY_ALPHANUMERIC_EDITED;
    pic->lenstr = idx;

```

```

        break;
    default:
        goto error;
    }
    goto end;

error:
    cb_error (_("Invalid picture string - '%s'"), str);

end:
    return CB_TREE (pic);
}

```

#### 7.16.4.54 `cb_tree cb_build_ppointer ( cb_tree x )`

Definition at line 1038 of file typeck.c.

```

{
    struct cb_field *f;

    if (x == cb_error_node ||
        (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node)) {
        return cb_error_node;
    }

    if (CB_REFERENCE_P (x)) {
        f = cb_field (cb_ref(x));
        f->count++;
    }
    return cb_build_cast_ppointer (x);
}

```

#### 7.16.4.55 `struct cb_program* cb_build_program ( struct cb_program * last_program, int nest_level )` [read]

Definition at line 841 of file tree.c.

```

{
    struct cb_program *p;

    cb_reset_78 ();
    cb_reset_in_procedure ();
    cb_clear_real_field ();
    p = cobc_malloc (sizeof (struct cb_program));
    p->next_program = last_program;
    p->nested_level = nest_level;
    p->decimal_point = '.';
    p->currency_symbol = '$';
    p->numeric_separator = ',';
    if (nest_level) {
        p->global_file_list = last_program->global_file_list;
        p->collating_sequence = last_program->collating_sequence;
        p->function_spec_list = last_program->function_spec_list;
        p->class_spec_list = last_program->class_spec_list;
        p->interface_spec_list = last_program->interface_spec_list;
    }
}

```

```

        p->program_spec_list = last_program->program_spec_list;
        p->property_spec_list = last_program->property_spec_list;
        p->alphabet_name_list = last_program->alphabet_name_list;
        p->class_name_list = last_program->class_name_list;
        p->locale_list = last_program->locale_list;
        p->symbolic_list = last_program->symbolic_list;
        p->decimal_point = last_program->decimal_point;
        p->numeric_separator = last_program->numeric_separator;
        p->currency_symbol = last_program->currency_symbol;
        p->cb_return_code = last_program->cb_return_code;
    } else {
        functions_are_all = cb_flag_functions_all;
    }
    return p;
}

```

#### 7.16.4.56 `const char* cb_build_program_id ( cb_tree name, cb_tree alt_name )`

Definition at line 591 of file typeck.c.

```

{
    const char    *s;

    /* This needs some more thought, should we generate an entry
    point per program source name ?
    if (alt_name) {
        s = (char *)CB_LITERAL (alt_name->data);
    } else if (CB_LITERAL_P (name)) {
        s = (char *)CB_LITERAL (name->data);
    } else {
        s = (char *)CB_NAME (name);
    }

    if (!cb_flag_main && strcmp (s, source_name)) {
        cb_warning (_("Source name '%s' differs from PROGRAM-ID '%s'"),
            source_name, s);
        current_program->source_name = strdup (source_name);
    }
    End comment out */

    if (alt_name) {
        current_program->orig_source_name = strdup ((char *)CB_LITERAL (a
lt_name->data);
        s = (char *)CB_LITERAL (alt_name->data);
    } else if (CB_LITERAL_P (name)) {
        current_program->orig_source_name = strdup ((char *)CB_LITERAL (n
ame->data);
        s = cb_encode_program_id ((char *)CB_LITERAL (name->data);
    } else {
        current_program->orig_source_name = strdup (CB_NAME (name));
        s = cb_encode_program_id (CB_NAME (name));
    }
    if (cobc_check_valid_name (current_program->orig_source_name)) {
        cb_error (_("PROGRAM-ID '%s' invalid"), current_program->
orig_source_name);
    }
    return s;
}

```

7.16.4.57 `cb_tree cb_build_reference ( const char * name )`

Definition at line 1730 of file tree.c.

```

{
    struct cb_reference *p;

    p = make_tree (CB_TAG_REFERENCE, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_reference));
    p->word = lookup_word (name);
    return CB_TREE (p);
}

```

7.16.4.58 `void cb_build_registers ( void )`

Definition at line 494 of file typeck.c.

```

{
#if !defined(__linux__) && !defined(__CYGWIN__) && defined(HAVE_TIMEZONE)
    long    contz;
#endif
    time_t  t;
    char    buff[48];

    /* RETURN-CODE */
    if (!current_program->nested_level) {
        current_program->cb_return_code =
            cb_build_index (cb_build_reference ("RETURN-CODE"),
                cb_zero, 0, NULL);
        cb_field (current_program->cb_return_code)->flag_is_global = 1;
    }

    /* SORT-RETURN */
    current_program->cb_sort_return =
        cb_build_index (cb_build_reference ("SORT-RETURN"), cb_zero, 0,
NULL);
    cb_field (current_program->cb_sort_return)->flag_no_init = 1;

    /* NUMBER-OF-CALL-PARAMETERS */
    current_program->cb_call_params =
        cb_build_index (cb_build_reference ("NUMBER-OF-CALL-PARAMETERS"),
cb_zero, 0, NULL);
    cb_field (current_program->cb_call_params)->flag_no_init = 1;

    /* TALLY */
    /* 01 TALLY GLOBAL PICTURE 9(9) USAGE COMP-5 VALUE ZERO. */
    /* TALLY/EXAMINE not standard/supported */

    t = time (NULL);

    /* WHEN-COMPILED */
    memset (buff, 0, sizeof (buff));
    strftime (buff, 17, "%m/%d/%y%H.%M.%S", localtime (&t));
    cb_build_constant (cb_build_reference ("WHEN-COMPILED"),
        cb_build_alphanumeric_literal ((ucharptr)buff, 16));

    /* FUNCTION WHEN-COMPILED */
    memset (buff, 0, sizeof (buff));

```

```

#if defined(__linux__) || defined(__CYGWIN__)
    strftime (buff, 22, "%Y%m%d%H%M%S00%z", localtime (&t));
#elif defined(HAVE_TIMEZONE)
    strftime (buff, 17, "%Y%m%d%H%M%S00", localtime (&t));
    if (timezone <= 0) {
        contz = -timezone;
        buff[16] = '+';
    } else {
        contz = timezone;
        buff[16] = '-';
    }
    sprintf (&buff[17], "%2.2ld%2.2ld", contz / 3600, contz % 60);
#else
    strftime (buff, 22, "%Y%m%d%H%M%S0000000", localtime (&t));
#endif
cb_intr_whencomp = cb_build_alphanumeric_literal ((ucharptr)buff, 21);

/* FUNCTION PI */
memset (buff, 0, sizeof (buff));
strcpy (buff, "31415926535897932384626433832795029");
cb_intr_pi = cb_build_numeric_literal (0, (ucharptr)buff, 34);

/* FUNCTION E */
memset (buff, 0, sizeof (buff));
strcpy (buff, "27182818284590452353602874713526625");
cb_intr_e = cb_build_numeric_literal (0, (ucharptr)buff, 34);
}

```

#### 7.16.4.59 `cb_tree cb_build_replacing_all ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3823 of file typeck.c.

```

{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_all", y, x));
}

```

#### 7.16.4.60 `cb_tree cb_build_replacing_characters ( cb_tree x, cb_tree l )`

Definition at line 3817 of file typeck.c.

```

{
    return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_characters", x));
}

```

#### 7.16.4.61 `cb_tree cb_build_replacing_first ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3835 of file typeck.c.

```

{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_first", y, x));
}

```

**7.16.4.62 `cb_tree` `cb_build_replacing_leading` ( `cb_tree` *x*, `cb_tree` *y*, `cb_tree` *l* )**

Definition at line 3829 of file `typeck.c`.

```
{
    return cb_list_add (1, cb_build_funcall_2 ("cob_inspect_leading", y, x));
}
```

**7.16.4.63 `cb_tree` `cb_build_replacing_trailing` ( `cb_tree` *x*, `cb_tree` *y*, `cb_tree` *l* )**

Definition at line 3841 of file `typeck.c`.

```
{
    return cb_list_add (1, cb_build_funcall_2 ("cob_inspect_trailing", y, x)
;
}
```

**7.16.4.64 `cb_tree` `cb_build_search` ( `int` *flag\_all*, `cb_tree` *table*, `cb_tree` *var*, `cb_tree` *end\_stmt*, `cb_tree` *whens* )**

Definition at line 2131 of file `tree.c`.

```
{
    struct cb_search *p;

    p = make_tree (CB_TAG_SEARCH, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_search));
    p->flag_all = flag_all;
    p->table = table;
    p->var = var;
    p->end_stmt = end_stmt;
    p->whens = whens;
    return CB_TREE (p);
}
```

**7.16.4.65 `cb_tree` `cb_build_section_name` ( `cb_tree` *name*, `int` *sect\_or\_para* )**

Definition at line 653 of file `typeck.c`.

```
{
    cb_tree x;

    if (name == cb_error_node) {
        return cb_error_node;
    }

    if (CB_REFERENCE (name)->word->count > 0) {
        x = CB_VALUE (CB_REFERENCE (name)->word->items);
        /* Used as a non-label name or used as a section name.

```

```

        Duplicate paragraphs are allowed if not referenced;
        Checked in typeck.c */
        if (!CB_LABEL_P (x) || sect_or_para == 0
            || (sect_or_para && CB_LABEL_P (x) && CB_LABEL (x)->is_sectio
n)) {
                redefinition_error (name);
                return cb_error_node;
            }
        }
    return name;
}

```

#### 7.16.4.66 `struct cb_statement* cb_build_statement ( const char * name )` [read]

Definition at line 2226 of file tree.c.

```

{
    struct cb_statement *p;

    p = make_tree (CB_TAG_STATEMENT, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_statement));
    p->name = name;
    return p;
}

```

#### 7.16.4.67 `cb_tree cb_build_string ( const unsigned char * data, size_t size )`

Definition at line 908 of file tree.c.

```

{
    struct cb_string *p;

    p = make_tree (CB_TAG_STRING, CB_CATEGORY_ALPHANUMERIC, sizeof (struct
cb_string));
    p->size = size;
    p->data = data;
    return CB_TREE (p);
}

```

#### 7.16.4.68 `cb_tree cb_build_sub ( cb_tree v, cb_tree n, cb_tree round_opt )`

Definition at line 2633 of file typeck.c.

```

{
    cb_tree      opt;
    struct cb_field *f;

#ifdef COB_NON_ALIGNED
    if (CB_INDEX_P (v)) {
        return cb_build_move (cb_build_binary_op (v, '-', n), v);
    }
}

```



```

    if (CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        current_program->gen_ptrmanip = 1;
        return cb_build_funcall_3 ("cob_pointer_manip", v, n, cb_int1);
    }
#else
    if (CB_INDEX_P (v) || CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        return cb_build_move (cb_build_binary_op (v, '-', n), v);
    }
#endif

    if (CB_REF_OR_FIELD_P (v)) {
        f = cb_field (v);
        f->count++;
    }
    if (CB_REF_OR_FIELD_P (n)) {
        f = cb_field (n);
        f->count++;
    }
    opt = build_store_option (v, round_opt);
    if (opt == cb_int0 && cb_fits_int (n)) {
        return cb_build_optim_sub (v, n);
    }
    return cb_build_funcall_3 ("cob_sub", v, n, opt);
}

```

#### 7.16.4.69 `cb_tree cb_build_system_name ( enum cb_system_name_category category, int token )`

Definition at line 973 of file tree.c.

```

{
    struct cb_system_name *p;

    p = make_tree (CB_TAG_SYSTEM_NAME, CB_CATEGORY_UNKNOWN, sizeof (struct
cb_system_name));
    p->category = category;
    p->token = token;
    return CB_TREE (p);
}

```

#### 7.16.4.70 `cb_tree cb_build_tarrying_all ( void )`

Definition at line 3778 of file typeck.c.

```

{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before ALL"));
    }
    inspect_func = "cob_inspect_all";
    return NULL;
}

```

**7.16.4.71   cb\_tree cb\_build\_tarrying\_characters ( cb\_tree l )**

Definition at line 3768 of file typeck.c.

```
{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before CHARACTERS"));
    }
    inspect_func = NULL;
    return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_characters", inspect_data));
}
```

**7.16.4.72   cb\_tree cb\_build\_tarrying\_data ( cb\_tree x )**

Definition at line 3761 of file typeck.c.

```
{
    inspect_data = x;
    return NULL;
}
```

**7.16.4.73   cb\_tree cb\_build\_tarrying\_leading ( void )**

Definition at line 3788 of file typeck.c.

```
{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before LEADING"));
    }
    inspect_func = "cob_inspect_leading";
    return NULL;
}
```

**7.16.4.74   cb\_tree cb\_build\_tarrying\_trailing ( void )**

Definition at line 3798 of file typeck.c.

```
{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before TRAILING"));
    }
    inspect_func = "cob_inspect_trailing";
    return NULL;
}
```

**7.16.4.75** `cb_tree cb_build_tarrying_value ( cb_tree x, cb_tree l )`

Definition at line 3808 of file typeck.c.

```
{
    if (inspect_func == NULL) {
        cb_error_x (x, _("ALL, LEADING or TRAILING expected before '%s'")
, cb_name (x));
    }
    return cb_list_add (l, cb_build_funcall_2 (inspect_func, inspect_data, x)
);
}
```

**7.16.4.76** `cb_tree cb_build_unstring_delimited ( cb_tree all, cb_tree value )`

Definition at line 5866 of file typeck.c.

```
{
    if (cb_validate_one (value)) {
        return cb_error_node;
    }
    return cb_build_funcall_2 ("cob_unstring_delimited", value, all);
}
```

**7.16.4.77** `cb_tree cb_build_unstring_into ( cb_tree name, cb_tree delimiter, cb_tree count )`

Definition at line 5875 of file typeck.c.

```
{
    if (cb_validate_one (name)) {
        return cb_error_node;
    }
    if (delimiter == NULL) {
        delimiter = cb_int0;
    }
    if (count == NULL) {
        count = cb_int0;
    }
    return cb_build_funcall_3 ("cob_unstring_into", name, delimiter, count);
}
```

**7.16.4.78** `cb_tree cb_build_write_advancing_lines ( cb_tree pos, cb_tree lines )`

Definition at line 5957 of file typeck.c.

```
{
    cb_tree e;
    int opt;
```

```

    opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
    e = cb_build_binary_op (cb_int (opt | COB_WRITE_LINES), '+', lines);
    return cb_build_cast_integer (e);
}

```

#### 7.16.4.79 `cb_tree cb_build_write_advancing_mnemonic ( cb_tree pos, cb_tree mnemonic )`

Definition at line 5968 of file typeck.c.

```

{
    int    opt;
    int    token;

    token = CB_SYSTEM_NAME (cb_ref (mnemonic))->token;
    switch (token) {
    case CB_FEATURE_FORMFEED:
        opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
        return cb_int (opt | COB_WRITE_PAGE);
    case CB_FEATURE_C01:
    case CB_FEATURE_C02:
    case CB_FEATURE_C03:
    case CB_FEATURE_C04:
    case CB_FEATURE_C05:
    case CB_FEATURE_C06:
    case CB_FEATURE_C07:
    case CB_FEATURE_C08:
    case CB_FEATURE_C09:
    case CB_FEATURE_C10:
    case CB_FEATURE_C11:
    case CB_FEATURE_C12:
        opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
        return cb_int (opt | COB_WRITE_CHANNEL | COB_WRITE_PAGE | token);
    default:
        cb_error_x (mnemonic, _("Invalid mnemonic name"));
        return cb_error_node;
    }
}

```

#### 7.16.4.80 `cb_tree cb_build_write_advancing_page ( cb_tree pos )`

Definition at line 5999 of file typeck.c.

```

{
    int opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;

    return cb_int (opt | COB_WRITE_PAGE);
}

```

#### 7.16.4.81 `cb_tree cb_check_numeric_value ( cb_tree x )`

Definition at line 426 of file typeck.c.

```

{
    if (x == cb_error_node) {
        return cb_error_node;
    }

    if (CB_TREE_CATEGORY (x) == CB_CATEGORY_NUMERIC) {
        return x;
    }

    cb_error_x (x, _("%s' is not a numeric value"), cb_name (x));
    return cb_error_node;
}

```

#### 7.16.4.82 void cb\_clear\_real\_field ( void )

Definition at line 1050 of file field.c.

```

{
    last_real_field = NULL;
}

```

#### 7.16.4.83 cb\_tree cb\_concat\_literals ( cb\_tree x1, cb\_tree x2 )

Definition at line 1005 of file tree.c.

```

{
    unsigned char      *buff;
    cb_tree            x;
    unsigned char      *data1;
    unsigned char      *data2;
    size_t             size1;
    size_t             size2;

    if (x1 == cb_error_node || x2 == cb_error_node) {
        return cb_error_node;
    }
    if (CB_LITERAL_P (x1)) {
        data1 = CB_LITERAL (x1)->data;
        size1 = CB_LITERAL (x1)->size;
    } else if (CB_CONST_P (x1)) {
        size1 = 1;
        if (x1 == cb_space) {
            data1 = (unsigned char *) " ";
        } else if (x1 == cb_zero) {
            data1 = (unsigned char *) "0";
        } else if (x1 == cb_quote) {
            data1 = (unsigned char *) "\"";
        } else if (x1 == cb_norm_low) {
            data1 = (unsigned char *) "\0";
        } else if (x1 == cb_norm_high) {
            data1 = (unsigned char *) "\255";
        } else if (x1 == cb_null) {
            data1 = (unsigned char *) "\0";
        } else {
            return cb_error_node;
        }
    }
}

```

```

    }
  } else {
    return cb_error_node;
  }
  if (CB_LITERAL_P (x2)) {
    data2 = CB_LITERAL (x2)->data;
    size2 = CB_LITERAL (x2)->size;
  } else if (CB_CONST_P (x2)) {
    size2 = 1;
    if (x2 == cb_space) {
      data2 = (unsigned char *) " ";
    } else if (x2 == cb_zero) {
      data2 = (unsigned char *) "0";
    } else if (x2 == cb_quote) {
      data2 = (unsigned char *) "\"";
    } else if (x2 == cb_norm_low) {
      data2 = (unsigned char *) "\0";
    } else if (x2 == cb_norm_high) {
      data2 = (unsigned char *) "\255";
    } else if (x2 == cb_null) {
      data2 = (unsigned char *) "\0";
    } else {
      return cb_error_node;
    }
  } else {
    return cb_error_node;
  }
  buff = ccbc_malloc (size1 + size2 + 3);
  memcpy (buff, data1, size1);
  memcpy (buff + size1, data2, size2);
  x = cb_build_alphanumeric_literal (buff, size1 + size2);
  free (buff);
  return x;
}

```

#### 7.16.4.84 `const char* cb_define ( cb_tree name, cb_tree val )`

Definition at line 1769 of file tree.c.

```

{
  struct cb_word *w;

  w = CB_REFERENCE (name)->word;
  w->items = cb_list_add (w->items, val);
  w->count++;
  val->source_file = name->source_file;
  val->source_line = name->source_line;
  CB_REFERENCE (name)->value = val;
  return w->name;
}

```

#### 7.16.4.85 `void cb_define_switch_name ( cb_tree name, cb_tree sname, cb_tree flag, cb_tree ref )`

Definition at line 629 of file typeck.c.

```

{
    cb_tree switch_id;
    cb_tree value;

    if (name == cb_error_node) {
        return;
    }
    if (sname == cb_error_node) {
        return;
    }
    if (CB_SYSTEM_NAME (sname)->category != CB_SWITCH_NAME) {
        cb_error_x (ref, _("Switch-name is expected '%s'"), CB_NAME (ref)
    );
    } else {
        switch_id = cb_int (CB_SYSTEM_NAME (sname)->token);
        value = cb_build_funcall_1 ("cob_get_switch", switch_id);
        if (flag == cb_int0) {
            value = cb_build_negation (value);
        }
        cb_build_constant (name, value);
    }
}

```

#### 7.16.4.86 void cb\_define\_system\_name ( const char \* name )

Definition at line 1783 of file tree.c.

```

{
    cb_tree x;

    x = cb_build_reference (name);
    if (CB_REFERENCE (x)->word->count == 0) {
        cb_define (x, lookup_system_name (name));
    }
}

```

#### 7.16.4.87 void cb\_emit\_accept ( cb\_tree var, cb\_tree pos, cb\_tree fgc, cb\_tree bgc, cb\_tree scroll, int dispatrs )

Definition at line 2802 of file typeck.c.

```

{
    cb_tree line;
    cb_tree column;

    if (cb_validate_one (var)) {
        return;
    }
    if (cb_validate_one (pos)) {
        return;
    }
    if (cb_validate_one (fgc)) {
        return;
    }
    if (cb_validate_one (bgc)) {

```

```

        return;
    }
    if (cb_validate_one (scroll)) {
        return;
    }
    if (current_program->flag_screen) {
        /* Bump ref count to force CRT STATUS field generation */
        cb_field (current_program->crt_status)->count++;
        if ((CB_REF_OR_FIELD_P (var)) &&
            CB_FIELD (cb_ref (var))->storage == CB_STORAGE_SCREEN) {
            output_screen_from (CB_FIELD (cb_ref (var)), 0);
            gen_screen_ptr = 1;
            if (pos) {
                if (CB_PAIR_P (pos)) {
                    line = CB_PAIR_X (pos);
                    column = CB_PAIR_Y (pos);
                    cb_emit (cb_build_funcall_3 ("cob_screen_
accept",
                                                var, line, column));
                } else {
                    cb_emit (cb_build_funcall_3 ("cob_screen_
accept",
                                                var, pos, NULL));
                }
            } else {
                cb_emit (cb_build_funcall_3 ("cob_screen_accept",
                                            var, NULL, NULL));
            }
            gen_screen_ptr = 0;
            output_screen_to (CB_FIELD (cb_ref (var)), 0);
        } else {
            if (pos || fgc || bgc) {
                if (!pos) {
                    cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                                var, NULL, NULL, fgc, bgc,
                                                scroll, cb_int (dispattrs)));
                } else if (CB_PAIR_P (pos)) {
                    line = CB_PAIR_X (pos);
                    column = CB_PAIR_Y (pos);
                    cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                                var, line, column, fgc, bgc,
                                                scroll, cb_int (dispattrs)));
                } else {
                    cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                                var, pos, NULL, fgc, bgc,
                                                scroll, cb_int (dispattrs)));
                }
            } else {
                cb_emit (cb_build_funcall_7 ("cob_field_accept",
                                            var, NULL, NULL, fgc, bgc,
                                            scroll, cb_int (dispattrs)));
            }
        }
    }
    } else if (pos || fgc || bgc || scroll) {
        /* Bump ref count to force CRT STATUS field generation */
        cb_field (current_program->crt_status)->count++;
        if (!pos) {
            cb_emit (cb_build_funcall_7 ("cob_field_accept",

```



```
        var, NULL, NULL, fgC, bgC, scroll,
        cb_int (dispattrs));
    } else if (CB_PAIR_P (pos)) {
        line = CB_PAIR_X (pos);
        column = CB_PAIR_Y (pos);
        cb_emit (cb_build_funcall_7 ("cob_field_accept",
        var, line, column, fgC, bgC, scroll,
        cb_int (dispattrs)));
    } else {
        cb_emit (cb_build_funcall_7 ("cob_field_accept",
        var, pos, NULL, fgC, bgC, scroll,
        cb_int (dispattrs)));
    }
} else {
    cb_emit (cb_build_funcall_1 ("cob_accept", var));
}
}
```

#### 7.16.4.88 void cb\_emit\_accept\_arg\_number ( cb\_tree var )

Definition at line 2986 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_arg_number", var));
}
```

#### 7.16.4.89 void cb\_emit\_accept\_arg\_value ( cb\_tree var )

Definition at line 2995 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_arg_value", var));
}
```

#### 7.16.4.90 void cb\_emit\_accept\_command\_line ( cb\_tree var )

Definition at line 2956 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_command_line", var));
}
```

**7.16.4.91 void cb\_emit\_accept\_date ( cb\_tree var )**

Definition at line 2902 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_date", var));
}
```

**7.16.4.92 void cb\_emit\_accept\_date\_yyyymmdd ( cb\_tree var )**

Definition at line 2911 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_date_yyyymmdd", var));
}
```

**7.16.4.93 void cb\_emit\_accept\_day ( cb\_tree var )**

Definition at line 2920 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_day", var));
}
```

**7.16.4.94 void cb\_emit\_accept\_day\_of\_week ( cb\_tree var )**

Definition at line 2938 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_day_of_week", var));
}
```

**7.16.4.95 void cb\_emit\_accept\_day\_yyyyddd ( cb\_tree var )**

Definition at line 2929 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funccall_1 ("cob_accept_day_yyyyddd", var));
}
```

#### 7.16.4.96 void cb\_emit\_accept\_environment ( cb\_tree var )

Definition at line 2977 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funccall_1 ("cob_accept_environment", var));
}
```

#### 7.16.4.97 void cb\_emit\_accept\_line\_or\_col ( cb\_tree var, const int l\_or\_c )

Definition at line 2893 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funccall_2 ("cob_screen_line_col", var, cb_int (l_or_c))
);
}
```

#### 7.16.4.98 void cb\_emit\_accept\_mnemonic ( cb\_tree var, cb\_tree mnemonic )

Definition at line 3004 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    switch (CB_SYSTEM_NAME (cb_ref (mnemonic))->token) {
    case CB_DEVICE_CONSOLE:
    case CB_DEVICE_SYSIN:
        cb_emit (cb_build_funccall_1 ("cob_accept", var));
        break;
    default:
        cb_error_x (mnemonic, _("Invalid input stream '%s'"),
                    cb_name (mnemonic));
        break;
    }
}
```

**7.16.4.99 void cb\_emit\_accept\_name ( cb\_tree var, cb\_tree name )**

Definition at line 3022 of file typeck.c.

```

{
    cb_tree sys;

    if (cb_validate_one (var)) {
        return;
    }
    if (CB_REFERENCE (name)->word->count == 0) {
        sys = lookup_system_name (CB_NAME (name));

        if (sys != cb_error_node) {
            switch (CB_SYSTEM_NAME (sys)->token) {
            case CB_DEVICE_CONSOLE:
            case CB_DEVICE_SYSIN:
                cb_warning_x (name, _("%s' undefined in SPECIAL-
NAMES"), CB_NAME (name));
                cb_emit (cb_build_funcall_1 ("cob_accept", var));
                return;
            default:
                break;
            }
        }

        cb_error_x (name, _("%s' undefined in SPECIAL-NAMES"), CB_NAME (name));
    }
}

```

**7.16.4.100 void cb\_emit\_accept\_time ( cb\_tree var )**

Definition at line 2947 of file typeck.c.

```

{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_time", var));
}

```

**7.16.4.101 void cb\_emit\_allocate ( cb\_tree target1, cb\_tree target2, cb\_tree size, cb\_tree initialize )**

Definition at line 3053 of file typeck.c.

```

{
    cb_tree x;
    char buff[32];

    if (cb_validate_one (target1)) {
        return;
    }
}

```

```

    if (cb_validate_one (target2)) {
        return;
    }
    if (cb_validate_one (size)) {
        return;
    }
    if (target1) {
        if (!(CB_REFERENCE_P(target1) &&
            cb_field (target1)->flag_item_based)) {
            cb_error_x (CB_TREE(current_statement),
                _("Target of ALLOCATE is not a BASED item"));
        }
    }
    if (target2) {
        if (!(CB_REFERENCE_P(target2) &&
            CB_TREE_CLASS (target2) == CB_CLASS_POINTER)) {
            cb_error_x (CB_TREE(current_statement),
                _("Target of RETURNING is not a data pointer"));
        }
    }
    if (size) {
        if (CB_TREE_CLASS (size) != CB_CLASS_NUMERIC) {
            cb_error_x (CB_TREE(current_statement),
                _("The CHARACTERS field of ALLOCATE must be numeric"));
        }
    }
    if (target1) {
        sprintf (buff, "%d", cb_field (target1)->memory_size);
        x = cb_build_numeric_literal (0, (ucharptr)buff, 0);
        cb_emit (cb_build_funcall_3 ("cob_allocate",
            cb_build_cast_addr_of_addr (target1), target2, x));
    } else {
        cb_emit (cb_build_funcall_3 ("cob_allocate",
            NULL, target2, size));
    }
    if (initialize && target1) {
        current_statement->handler2 =
            cb_build_initialize (target1, cb_true, NULL, cb_true, 0);
    }
}

```

#### 7.16.4.102 void cb\_emit\_arg\_number ( cb\_tree value )

Definition at line 3273 of file typeck.c.

```

{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_arg_number", value));
}

```

#### 7.16.4.103 void cb\_emit\_arithmetic ( cb\_tree vars, int op, cb\_tree val )

Definition at line 2094 of file typeck.c.

```

{
    cb_tree      l;
    struct cb_field *f;

    val = cb_check_numeric_value (val);
    if (op) {
        cb_list_map (cb_check_numeric_name, vars);
    } else {
        cb_list_map (cb_check_numeric_edited_name, vars);
    }

    if (cb_validate_one (val)) {
        return;
    }
    if (cb_validate_list (vars)) {
        return;
    }

    if (!CB_BINARY_OP_P (val)) {
        if (op == '+' || op == '-') {
            if (CB_EXCEPTION_ENABLE (COB_EC_DATA_INCOMPATIBLE) &&
                (CB_REF_OR_FIELD_P (val))) {
                f = cb_field (val);
                if (f->usage == CB_USAGE_DISPLAY ||
                    f->usage == CB_USAGE_PACKED) {
                    cb_emit (cb_build_funcall_2 ("cob_check_n
umeric",
                                                val,
                                                cb_build_string0 ((
ucharptr) (f->name))));
                }
            }
            for (l = vars; l; l = CB_CHAIN (l)) {
                if (CB_EXCEPTION_ENABLE (COB_EC_DATA_INCOMPATIBLE
) &&
                    (CB_REF_OR_FIELD_P (CB_VALUE(l)))) {
                    f = cb_field (CB_VALUE(l));
                    if (f->usage == CB_USAGE_DISPLAY ||
                        f->usage == CB_USAGE_PACKED) {
                        cb_emit (cb_build_funcall_2 ("cob
_check_numeric",
                                                    CB_VALUE(l),
                                                    cb_build_string0 ((
ucharptr) (f->name))));
                    }
                }
                if (op == '+') {
                    CB_VALUE (l) = cb_build_add (CB_VALUE (l)
, val, CB_PURPOSE (l));
                } else {
                    CB_VALUE (l) = cb_build_sub (CB_VALUE (l)
, val, CB_PURPOSE (l));
                }
            }
            cb_emit_list (vars);
            return;
        }
    }
    cb_emit (build_decimal_assign (vars, op, val));
}

```

#### 7.16.4.104 void cb\_emit\_call ( cb\_tree prog, cb\_tree using, cb\_tree returning, cb\_tree on\_exception, cb\_tree not\_on\_exception )

Definition at line 3108 of file typeck.c.

```

{
    cb_tree          l;
    cb_tree          x;
    const struct system_table *psyst;
    int              is_sys_call = 0;

    if (CB_INTRINSIC_P (prog)) {
        if (CB_INTRINSIC(prog)->intr_tab->category !=
CB_CATEGORY_ALPHANUMERIC) {
            cb_error (_("Only alphanumeric FUNCTION types are allowed
here"));
            return;
        }
    }
    if (returning) {
        if (CB_TREE_CLASS(returning) != CB_CLASS_NUMERIC &&
            CB_TREE_CLASS(returning) != CB_CLASS_POINTER) {
            cb_error (_("Invalid RETURNING field"));
            return;
        }
    }
    for (l = using; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            continue;
        }
        if (CB_CONST_P (x) && x != cb_null) {
            cb_error_x (x, _("Figurative constant invalid here"));
        }
        if ((CB_REFERENCE_P (x) && CB_FIELD_P(CB_REFERENCE(x)->value)
|| CB_FIELD_P (x)) {
            if (cb_field (x)->level == 88) {
                cb_error_x (x, _("'%s' Not a data name"),
CB_NAME (x));
                return;
            }
            if (cb_warn_call_params &&
                CB_PURPOSE_INT (l) == CB_CALL_BY_REFERENCE) {
                if (cb_field (x)->level != 01 &&
                    cb_field (x)->level != 77) {
                    cb_warning_x (x, _("'%s' is not 01 or 77
level item"), CB_NAME (x));
                }
            }
        }
    }

    if (CB_LITERAL_P(prog)) {
        for (psyst = (const struct system_table *)&system_tab[0]; psyst->
syst_name; psyst++) {
            if (!strcmp((const char *)CB_LITERAL(prog)->data,
                (const char *)psyst->syst_name)) {
                if (psyst->syst_params > cb_list_length (using))
{
                    cb_error (_("Wrong number of CALL paramet
ers for '%s'"),

```

```

                                                (char *)psyst->syst_name);
                                                return;
                                        }
                                        is_sys_call = 1;
                                        break;
                                }
        }
    }
    cb_emit (cb_build_call (prog, using, on_exception, not_on_exception,
        returning, is_sys_call));
}

```

#### 7.16.4.105 void cb\_emit\_cancel ( cb\_tree prog )

Definition at line 3177 of file typeck.c.

```

{
    if (cb_validate_one (prog)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_field_cancel", prog));
}

```

#### 7.16.4.106 void cb\_emit\_close ( cb\_tree file, cb\_tree opt )

Definition at line 3190 of file typeck.c.

```

{
    if (file == cb_error_node) {
        return;
    }
    file = cb_ref (file);
    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    cb_emit (cb_build_funcall_3 ("cob_close", file, opt,
        CB_FILE(file)->file_status));
}

```

#### 7.16.4.107 void cb\_emit\_command\_line ( cb\_tree value )

Definition at line 3282 of file typeck.c.

```

{
    if (cb_validate_one (value)) {
        return;
    }
}

```



```
    }  
    cb_emit (cb_build_funcall_1 ("cob_display_command_line", value));  
}
```

#### 7.16.4.108 void cb\_emit\_commit ( void )

Definition at line 3213 of file typeck.c.

```
{  
    cb_emit (cb_build_funcall_0 ("cob_commit"));  
}
```

#### 7.16.4.109 void cb\_emit\_continue ( void )

Definition at line 3223 of file typeck.c.

```
{  
    cb_emit (cb_build_continue ());  
}
```

#### 7.16.4.110 void cb\_emit\_corresponding ( cb\_tree\*(cb\_tree f1, cb\_tree f2, cb\_tree f3) func, cb\_tree x1, cb\_tree x2, cb\_tree opt )

Definition at line 2695 of file typeck.c.

```
{  
    x1 = cb_check_group_name (x1);  
    x2 = cb_check_group_name (x2);  
  
    if (cb_validate_one (x1)) {  
        return;  
    }  
    if (cb_validate_one (x2)) {  
        return;  
    }  
  
    emit_corresponding (func, x1, x2, opt);  
}
```

#### 7.16.4.111 void cb\_emit\_delete ( cb\_tree file )

Definition at line 3233 of file typeck.c.

```
{  
    if (file == cb_error_node) {  
        return;  
    }  
    file = cb_ref (file);  
}
```

```

    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    cb_emit (cb_build_funcall_2 ("cob_delete", file, CB_FILE(file)->file_stat
us));
}

```

**7.16.4.112 void cb.emit.display ( cb\_tree values, cb\_tree upon, cb\_tree no.adv, cb\_tree pos, cb\_tree fg, cb\_tree bg, cb\_tree scroll, int dispattrs )**

Definition at line 3291 of file typeck.c.

```

{
    cb_tree l;
    cb_tree x;
    cb_tree line;
    cb_tree column;
    cb_tree p;

    if (cb_validate_list (values)) {
        return;
    }
    if (cb_validate_one (pos)) {
        return;
    }
    if (cb_validate_one (fg)) {
        return;
    }
    if (cb_validate_one (bg)) {
        return;
    }
    if (cb_validate_one (scroll)) {
        return;
    }
    for (l = values; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            return;
        }

        switch (CB_TREE_TAG (x)) {
            case CB_TAG_LITERAL:
            case CB_TAG_INTRINSIC:
            case CB_TAG_CONST:
            case CB_TAG_STRING:
            case CB_TAG_INTEGER:
                break;
            case CB_TAG_REFERENCE:
                if (!CB_FIELD_P (CB_REFERENCE (x)->value)) {
                    cb_error_x (x, _("'%'s' is an invalid type for DIS
PLAY operand"), cb_name (x));
                    return;
                }
                break;
        }
    }
}

```

```

        default:
            cb_error_x (x, _("Invalid type for DISPLAY operand"));
            return;
        }
    }
    if (upon == cb_error_node) {
        return;
    }

    x = CB_VALUE (values);
    if ((CB_REF_OR_FIELD_P (x)) &&
        CB_FIELD (cb_ref (x))->storage == CB_STORAGE_SCREEN) {
        output_screen_from (CB_FIELD (cb_ref (x)), 0);
        gen_screen_ptr = 1;
        if (pos) {
            if (CB_PAIR_P (pos)) {
                line = CB_PAIR_X (pos);
                column = CB_PAIR_Y (pos);
                if (line == NULL) {
                    line = cb_one;
                }
                if (column == NULL) {
                    column = cb_one;
                }
                cb_emit (cb_build_funcall_3 ("cob_screen_display"
, x,
, x,
, x,
                line, column));
            } else {
                cb_emit (cb_build_funcall_3 ("cob_screen_display"
, x,
, x,
                pos, NULL));
            }
        } else {
            cb_emit (cb_build_funcall_3 ("cob_screen_display", x,
            NULL, NULL));
        }
        gen_screen_ptr = 0;
    } else if (pos || fgc || bgc || scroll || dispattrs) {
        if (!pos) {
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), NULL, NULL, fgc, bgc,
            scroll, cb_int (dispattrs)));
        } else if (CB_PAIR_P (pos)) {
            line = CB_PAIR_X (pos);
            column = CB_PAIR_Y (pos);
            if (line == NULL) {
                line = cb_one;
            }
            if (column == NULL) {
                column = cb_one;
            }
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), line, column, fgc, bgc,
            scroll, cb_int (dispattrs)));
        } else {
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), pos, NULL, fgc, bgc,
            scroll, cb_int (dispattrs)));
        }
    } else {
        /* DISPLAY x ... [UPON device-name] */
        p = cb_build_funcall_3 ("cob_display", upon, no_adv, values);
    }
}

```

```

        CB_FUNCALL(p)->varcnt = cb_list_length (values);
        cb_emit (p);
        for (l = values; l; l = CB_CHAIN (l)) {
            x = CB_VALUE (l);
            if (CB_FIELD_P (x)) {
                CB_FIELD (cb_ref (x))->count++;
            }
        }
    }
}

```

#### 7.16.4.113 void `cb_emit_divide` ( *cb\_tree dividend*, *cb\_tree divisor*, *cb\_tree quotient*, *cb\_tree remainder* )

Definition at line 3460 of file `typeck.c`.

```

{
    if (cb_validate_one (dividend)) {
        return;
    }
    if (cb_validate_one (divisor)) {
        return;
    }
    CB_VALUE (quotient) = cb_check_numeric_edited_name (CB_VALUE (quotient));
    CB_VALUE (remainder) = cb_check_numeric_edited_name (CB_VALUE (remainder)
);
    if (cb_validate_one (CB_VALUE (quotient))) {
        return;
    }
    if (cb_validate_one (CB_VALUE (remainder))) {
        return;
    }
    cb_emit (cb_build_funcall_4 ("cob_div_quotient", dividend, divisor,
                                CB_VALUE (quotient),
                                build_store_option (CB_VALUE (quotient),
CB_PURPOSE (quotient))));
    cb_emit (cb_build_funcall_2 ("cob_div_remainder", CB_VALUE (remainder),
                                build_store_option (CB_VALUE (remainder),
cb_int0)));
}

```

#### 7.16.4.114 void `cb_emit_env_name` ( *cb\_tree value* )

Definition at line 3255 of file `typeck.c`.

```

{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_environment", value));
}

```

**7.16.4.115 void cb\_emit\_env\_value ( cb\_tree value )**

Definition at line 3264 of file typeck.c.

```
{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_env_value", value));
}
```

**7.16.4.116 void cb\_emit\_evaluate ( cb\_tree subject\_list, cb\_tree case\_list )**

Definition at line 3601 of file typeck.c.

```
{
    cb_emit (build_evaluate (subject_list, case_list));
}
```

**7.16.4.117 void cb\_emit\_exit ( size\_t goback )**

Definition at line 3675 of file typeck.c.

```
{
    if (goback) {
        cb_emit (cb_build_goto (cb_int1, NULL));
    } else {
        cb_emit (cb_build_goto (NULL, NULL));
    }
}
```

**7.16.4.118 void cb\_emit\_free ( cb\_tree vars )**

Definition at line 3611 of file typeck.c.

```
{
    cb_tree l;
    struct cb_field *f;
    int i;

    if (cb_validate_list (vars)) {
        return;
    }
    for (l = vars, i = 1; l; l = CB_CHAIN (l), i++) {
        if (CB_TREE_CLASS (CB_VALUE (l)) == CB_CLASS_POINTER) {
            if (CB_CAST_P (CB_VALUE (l))) {
                f = cb_field (CB_CAST (CB_VALUE (l))->val);
                if (!f->flag_item_based) {
                    cb_error_x (CB_TREE (current_statement),
                                _("Target %d of FREE, a data addr
```

```

    ess identifier, must address a BASED data item"), i);
        }
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        cb_build_cast_address (CB_VALUE (1)),
NULL));
        } else {
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        NULL, cb_build_cast_address (CB_VALUE (1)
));
        }
        } else if (CB_REF_OR_FIELD_P (CB_VALUE (1))) {
        f = cb_field (CB_VALUE (1));
        if (!f->flag_item_based) {
        cb_error_x (CB_TREE (current_statement),
        _("Target %d of FREE, a data addr
ess identifier, must address a BASED data item"), i);
        }
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        cb_build_cast_addr_of_addr (CB_VALUE (1))
, NULL));
        } else {
        cb_error_x (CB_TREE (current_statement),
        _("Target %d of FREE must be a data pointer"), i)
;
        }
    }
}

```

#### 7.16.4.119 void cb\_emit\_get\_environment ( cb\_tree envvar, cb\_tree envval )

Definition at line 2965 of file typeck.c.

```

{
    if (cb_validate_one (envvar)) {
        return;
    }
    if (cb_validate_one (envval)) {
        return;
    }
    cb_emit (cb_build_funcall_2 ("cob_get_environment", envvar, envval));
}

```

#### 7.16.4.120 void cb\_emit\_goto ( cb\_tree target, cb\_tree depending )

Definition at line 3654 of file typeck.c.

```

{
    if (target == cb_error_node) {
        return;
    }
    if (depending) {
        /* GO TO procedure-name ... DEPENDING ON identifier */
        cb_emit (cb_build_goto (target, depending));
    } else {
        /* GO TO procedure-name */

```

```

        if (target == NULL) {
            cb_verify (cb_goto_statement_without_name, "GO TO without
procedure-name");
        } else if (CB_CHAIN (target)) {
            cb_error (_("GO TO with multiple procedure-names"));
        } else {
            cb_emit (cb_build_goto (CB_VALUE (target), NULL));
        }
    }
}

```

#### 7.16.4.121 void cb\_emit\_if ( cb\_tree cond, cb\_tree stmt1, cb\_tree stmt2 )

Definition at line 3689 of file typeck.c.

```

{
    cb_emit (cb_build_if (cond, stmt1, stmt2));
}

```

#### 7.16.4.122 void cb\_emit\_initialize ( cb\_tree vars, cb\_tree fillinit, cb\_tree value, cb\_tree replacing, cb\_tree def )

Definition at line 3699 of file typeck.c.

```

{
    cb_tree l;
    int fill_init = 1;

    if (cb_validate_list (vars)) {
        return;
    }
    if (value == NULL && replacing == NULL) {
        def = cb_true;
    }
    if (fillinit == cb_true) {
        fill_init = 0;
    }
    for (l = vars; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_initialize (CB_VALUE (l), value, replacing, def
, fill_init));
    }
}

```

#### 7.16.4.123 void cb\_emit\_inspect ( cb\_tree var, cb\_tree body, cb\_tree replacing, int replconv )

Definition at line 3723 of file typeck.c.

```

{
    switch (CB_TREE_TAG(var)) {
        case CB_TAG_REFERENCE:

```

```

        break;
    case CB_TAG_INTRINSIC:
        switch (CB_TREE_CATEGORY(var)) {
            case CB_CATEGORY_ALPHABETIC:
            case CB_CATEGORY_ALPHANUMERIC:
            case CB_CATEGORY_NATIONAL:
                break;
            default:
                cb_error (_("Invalid target for INSPECT"));
                return;
        }
        break;
    case CB_TAG_LITERAL:
        break;
    default:
        cb_error (_("Invalid target for REPLACING/CONVERTING"));
        return;
}
if (replconv && sending_id) {
    cb_error (_("Invalid target for REPLACING/CONVERTING"));
}
cb_emit (cb_build_funcall_2 ("cob_inspect_init", var, replacing));
cb_emit_list (body);
cb_emit (cb_build_funcall_0 ("cob_inspect_finish"));
}

```

#### 7.16.4.124 void cb\_emit\_move ( cb\_tree src, cb\_tree dsts )

Definition at line 5041 of file typeck.c.

```

{
    cb_tree l;

    if (cb_validate_one (src)) {
        return;
    }
    if (cb_validate_list (dsts)) {
        return;
    }

    for (l = dsts; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_move (src, CB_VALUE (l)));
    }
}

```

#### 7.16.4.125 void cb\_emit\_move\_corresponding ( cb\_tree x1, cb\_tree x2 )

Definition at line 2738 of file typeck.c.

```

{
    cb_tree      l;
    cb_tree      v;

    x1 = cb_check_group_name (x1);
    if (cb_validate_one (x1)) {

```



```

        return;
    }
    for (l = x2; l; l = CB_CHAIN(l)) {
        v = CB_VALUE(l);
        v = cb_check_group_name (v);
        if (cb_validate_one (v)) {
            return;
        }
        emit_move_corresponding (x1, v);
    }
}

```

#### 7.16.4.126 void cb\_emit\_open ( *cb\_tree file*, *cb\_tree mode*, *cb\_tree sharing* )

Definition at line 5062 of file typeck.c.

```

{
    if (file == cb_error_node) {
        return;
    }
    file = cb_ref (file);
    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;

    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    if (sharing == NULL) {
        sharing = CB_FILE (file)->sharing ? CB_FILE (file)->sharing :
cb_int0;
    }

    /* READ ONLY */
    if (sharing == cb_int0 && CB_INTEGER (mode)->val != COB_OPEN_INPUT) {
        sharing = cb_int1;
    }

    cb_emit (cb_build_funcall_4 ("cob_open", file, mode,
        sharing, CB_FILE(file)->file_status));
}

```

#### 7.16.4.127 void cb\_emit\_perform ( *cb\_tree perform*, *cb\_tree body* )

Definition at line 5095 of file typeck.c.

```

{
    if (perform == cb_error_node) {
        return;
    }
    CB_PERFORM (perform)->body = body;
    cb_emit (perform);
}

```

7.16.4.128 `void cb_emit_read ( cb_tree ref, cb_tree next, cb_tree into, cb_tree key, cb_tree lock_opts )`

Definition at line 5170 of file typeck.c.

```
{
    int      read_opts = 0;
    cb_tree file;
    cb_tree rec;

    if (lock_opts == cb_int1) {
        read_opts = COB_READ_LOCK;
    } else if (lock_opts == cb_int2) {
        read_opts = COB_READ_NO_LOCK;
    } else if (lock_opts == cb_int3) {
        read_opts = COB_READ_IGNORE_LOCK;
    } else if (lock_opts == cb_int4) {
        read_opts = COB_READ_WAIT_LOCK;
    }
    if (ref == cb_error_node) {
        return;
    }
    file = cb_ref (ref);
    if (file == cb_error_node) {
        return;
    }
    rec = cb_build_field_reference (CB_FILE (file)->record, ref);
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    if (next == cb_int1 || next == cb_int2 ||
        CB_FILE (file)->access_mode == COB_ACCESS_SEQUENTIAL) {
        /* READ NEXT/PREVIOUS */
        if (next == cb_int2) {
            if (CB_FILE (file)->organization != COB_ORG_INDEXED) {
                cb_error_x (CB_TREE (current_statement),
                    _("READ PREVIOUS only allowed for INDEXED SEQUENTIAL files"));
            }
            read_opts |= COB_READ_PREVIOUS;
        } else {
            read_opts |= COB_READ_NEXT;
        }
        if (key) {
            cb_warning (_("KEY ignored with sequential READ"));
        }
        cb_emit (cb_build_funcall_4 ("cob_read", file, cb_int0,
            CB_FILE(file)->file_status,
            cb_int (read_opts)));
    } else {
        /* READ */
        cb_emit (cb_build_funcall_4 ("cob_read",
            file, key ? key : CB_FILE (file)->key,
            CB_FILE(file)->file_status, cb_int (read_opts)));
    }
    if (into) {
        current_statement->handler3 = cb_build_move (rec, into);
    }
    current_statement->file = file;
}
```

**7.16.4.129 void cb\_emit\_release ( cb\_tree ref, cb\_tree from )**

Definition at line 5278 of file typeck.c.

```
{
    struct cb_field *f;
    cb_tree file;

    if (record == cb_error_node) {
        return;
    }
    if (from == cb_error_node) {
        return;
    }
    if (cb_ref (record) == cb_error_node) {
        return;
    }
    if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE requires a record name as subject"));
        return;
    }
    if (cb_field (record)->storage != CB_STORAGE_FILE) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE subject does not refer to a record name"));
        return;
    }
    f = CB_FIELD (cb_ref (record));
    file = CB_TREE (f->file);
    if (CB_FILE (file)->organization != COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE not allowed on this record item"));
        return;
    }
    current_statement->file = file;
    if (from) {
        cb_emit (cb_build_move (from, record));
    }
    cb_emit (cb_build_funcall_1 ("cob_file_release", file));
}
```

**7.16.4.130 void cb\_emit\_return ( cb\_tree ref, cb\_tree into )**

Definition at line 5321 of file typeck.c.

```
{
    cb_tree file;
    cb_tree rec;

    if (ref == cb_error_node) {
        return;
    }
    if (into == cb_error_node) {
        return;
    }
    file = cb_ref (ref);
    if (file == cb_error_node) {
        return;
    }
}
```

```

    }
    rec = cb_build_field_reference (CB_FILE (file)->record, ref);
    cb_emit (cb_build_funcall_1 ("cob_file_return", file));
    if (into) {
        current_statement->handler3 = cb_build_move (rec, into);
    }
    current_statement->file = file;
}

```

#### 7.16.4.131 void cb.emit.rewrite ( cb\_tree record, cb\_tree from, cb\_tree lockopt )

Definition at line 5232 of file typeck.c.

```

{
    cb_tree file;
    int     opts = 0;

    if (record == cb_error_node || cb_ref (record) == cb_error_node) {
        return;
    }
    if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
        cb_error_x (CB_TREE (current_statement),
            _("REWRITE requires a record name as subject"));
        return;
    }
    if (cb_field (record)->storage != CB_STORAGE_FILE) {
        cb_error_x (CB_TREE (current_statement),
            _("REWRITE subject does not refer to a record name"));
        return;
    }
    file = CB_TREE (CB_FIELD (cb_ref (record))->file);
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    } else if (current_statement->handler_id == COB_EC_I_O_INVALID_KEY &&
        (CB_FILE (file)->organization != COB_ORG_RELATIVE &&
        CB_FILE (file)->organization != COB_ORG_INDEXED)) {
        cb_error_x (CB_TREE (current_statement),
            _("INVALID KEY clause invalid with this file type"));
    } else if ((CB_FILE (file)->lock_mode & COB_LOCK_AUTOMATIC) && lockopt) {
        cb_error_x (CB_TREE (current_statement),
            _("LOCK clause invalid with file LOCK AUTOMATIC"));
    } else if (lockopt == cb_int1) {
        opts = COB_WRITE_LOCK;
    }
    if (from) {
        cb_emit (cb_build_move (from, record));
    }
    cb_emit (cb_build_funcall_4 ("cob_rewrite", file, record,
        cb_int (opts), CB_FILE (file)->file_status));
}

```

#### 7.16.4.132 void cb.emit.rollback ( void )

Definition at line 5349 of file typeck.c.

```
{
    cb_emit (cb_build_funcall_0 ("cob_rollback"));
}
```

#### 7.16.4.133 void `cb_emit_search` ( `cb_tree table`, `cb_tree varying`, `cb_tree at_end`, `cb_tree whens` )

Definition at line 5427 of file `typeck.c`.

```
{
    if (cb_validate_one (table)) {
        return;
    }
    if (cb_validate_one (varying)) {
        return;
    }
    if (table == cb_error_node) {
        return;
    }
    cb_emit (cb_build_search (0, table, varying, at_end, whens));
}
```

#### 7.16.4.134 void `cb_emit_search_all` ( `cb_tree table`, `cb_tree at_end`, `cb_tree when`, `cb_tree stmts` )

Definition at line 5442 of file `typeck.c`.

```
{
    if (cb_validate_one (table)) {
        return;
    }
    if (table == cb_error_node) {
        return;
    }
    cb_emit (cb_build_search (1, table, NULL, at_end,
                             cb_build_if (cb_build_search_all (table, when),
                             stmts, NULL)));
}
```

#### 7.16.4.135 void `cb_emit_set_false` ( `cb_tree l` )

Definition at line 5604 of file `typeck.c`.

```
{
    cb_tree      x;
    struct cb_field *f;
    cb_tree      ref;
    cb_tree      val;

    for (; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
```

```

        if (x == cb_error_node) {
            return;
        }
        if (!(CB_REFERENCE_P (x) && CB_FIELD_P(CB_REFERENCE(x)->value))
            && !CB_FIELD_P (x)) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        f = cb_field (x);
        if (f->level != 88) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        if (!f->>false_88) {
            cb_error_x (x, _("Field does not have FALSE clause"));
            return;
        }
        ref = cb_build_field_reference (f->parent, x);
        val = CB_VALUE (f->>false_88);
        if (CB_PAIR_P (val)) {
            val = CB_PAIR_X (val);
        }
        cb_emit (cb_build_move (val, ref));
    }
}

```

#### 7.16.4.136 void cb.emit.set.on.off ( cb\_tree l, cb\_tree flag )

Definition at line 5558 of file typeck.c.

```

{
    struct cb_system_name *s;

    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {
        s = CB_SYSTEM_NAME (cb_ref (CB_VALUE (l)));
        cb_emit (cb_build_funcall_2 ("cob_set_switch", cb_int (s->token),
flag));
    }
}

```

#### 7.16.4.137 void cb.emit.set.to ( cb\_tree l, cb\_tree x )

Definition at line 5465 of file typeck.c.

```

{
    cb_tree      l;
    cb_tree      v;
    struct cb_cast *p;

#ifdef 0
    enum cb_class class = CB_CLASS_UNKNOWN;
#endif
}

```

```

    if (cb_validate_one (x)) {
        return;
    }
    if (cb_validate_list (vars)) {
        return;
    }
#endif
    /* determine the class of targets */
    for (l = vars; l; l = CB_CHAIN (l)) {
        if (CB_TREE_CLASS (CB_VALUE (l)) != CB_CLASS_UNKNOWN) {
            if (class == CB_CLASS_UNKNOWN) {
                class = CB_TREE_CLASS (CB_VALUE (l));
            } else if (class != CB_TREE_CLASS (CB_VALUE (l))) {
                break;
            }
        }
    }
    if (l || (class != CB_CLASS_INDEX && class != CB_CLASS_POINTER)) {
        cb_error_x (CB_TREE (current_statement),
            _("The targets of SET must be either indexes or pointers"));
        return;
    }
#endif

    if (CB_CAST_P (x)) {
        p = CB_CAST (x);
        if (p->type == CB_CAST_PROGRAM_POINTER) {
            for (l = vars; l; l = CB_CHAIN (l)) {
                v = CB_VALUE (l);
                if (!CB_REFERENCE_P (v)) {
                    cb_error_x (CB_TREE (current_statement),
                        _("SET targets must be PROGRAM-POINTER"));
                }
                CB_VALUE (l) = cb_error_node;
            } else if (CB_FIELD(cb_ref(v))->usage !=
CB_USAGE_PROGRAM_POINTER) {
                cb_error_x (CB_TREE (current_statement),
                    _("SET targets must be PROGRAM-POINTER"));
            }
            CB_VALUE (l) = cb_error_node;
        }
    }
}
/* validate the targets */
for (l = vars; l; l = CB_CHAIN (l)) {
    v = CB_VALUE (l);
    if (CB_CAST_P (v)) {
        p = CB_CAST (v);
        if (p->type == CB_CAST_ADDRESS
            && !CB_FIELD (cb_ref (p->val))->flag_item_based
            && CB_FIELD (cb_ref (p->val))->storage !=
CB_STORAGE_LINKAGE) {
            cb_error_x (p->val, _("The address of '%s' cannot
be changed"),
                cb_name (p->val));
            CB_VALUE (l) = cb_error_node;
        }
    }
}

```

```

    if (cb_validate_list (vars)) {
        return;
    }

    for (l = vars; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_move (x, CB_VALUE (l)));
    }
}

```

#### 7.16.4.138 void cb\_emit\_set\_true ( cb\_tree l )

Definition at line 5572 of file typeck.c.

```

{
    cb_tree      x;
    struct cb_field *f;
    cb_tree      ref;
    cb_tree      val;

    for (; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            return;
        }
        if (!(CB_REFERENCE_P (x) && CB_FIELD_P (CB_REFERENCE (x)->value))
            && !CB_FIELD_P (x)) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        f = cb_field (x);
        if (f->level != 88) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        ref = cb_build_field_reference (f->parent, x);
        val = CB_VALUE (f->values);
        if (CB_PAIR_P (val)) {
            val = CB_PAIR_X (val);
        }
        cb_emit (cb_build_move (val, ref));
    }
}

```

#### 7.16.4.139 void cb\_emit\_set\_up\_down ( cb\_tree l, cb\_tree flag, cb\_tree x )

Definition at line 5540 of file typeck.c.

```

{
    if (cb_validate_one (x)) {
        return;
    }
    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {

```



```

        if (flag == cb_int0) {
            cb_emit (cb_build_add (CB_VALUE (1), x, cb_int0));
        } else {
            cb_emit (cb_build_sub (CB_VALUE (1), x, cb_int0));
        }
    }
}

```

#### 7.16.4.140 void cb\_emit\_setenv ( cb\_tree x, cb\_tree y )

Definition at line 5459 of file typeck.c.

```

{
    cb_emit (cb_build_funcall_2 ("cob_set_environment", x, y));
}

```

#### 7.16.4.141 void cb\_emit\_sort\_finish ( cb\_tree file )

Definition at line 5741 of file typeck.c.

```

{
    if (CB_FILE_P (cb_ref (file))) {
        cb_emit (cb_build_funcall_1 ("cob_file_sort_close", cb_ref (file)
    ));
    }
}

```

#### 7.16.4.142 void cb\_emit\_sort\_giving ( cb\_tree file, cb\_tree l )

Definition at line 5715 of file typeck.c.

```

{
    cb_tree      p;
    int          listlen;

    if (cb_validate_list (l)) {
        return;
    }
    for (p = l; p; p = CB_CHAIN (p)) {
        if (CB_FILE (cb_ref(CB_VALUE(p)))->organization == COB_ORG_SORT)
        {
            cb_error (_("Invalid SORT GIVING parameter"));
        }
    }
    listlen = cb_list_length (l);
    p = cb_build_funcall_2 ("cob_file_sort_giving", cb_ref (file), l);
    CB_FUNCALL(p)->varcnt = listlen;
    cb_emit (p);
}

```

## 7.16.4.143 void cb\_emit\_sort\_init ( cb\_tree name, cb\_tree keys, cb\_tree col )

Definition at line 5644 of file typeck.c.

```

{
    cb_tree      l;
    struct cb_field *f;

    if (cb_validate_list (keys)) {
        return;
    }
    for (l = keys; l; l = CB_CHAIN (l)) {
        if (CB_VALUE (l) == NULL) {
            CB_VALUE (l) = name;
        }
        cb_ref (CB_VALUE (l));
    }

    if (CB_FILE_P (cb_ref (name))) {
        if (CB_FILE (cb_ref (name))->organization != COB_ORG_SORT) {
            cb_error_x (name, _("Invalid SORT filename"));
        }
        cb_field (current_program->cb_sort_return)->count++;
        cb_emit (cb_build_funcall_5 ("cob_file_sort_init", cb_ref (name),
                                     cb_int (cb_list_length (keys)), col,
                                     cb_build_cast_address (
current_program->cb_sort_return),
                                     CB_FILE(cb_ref (name))->file_status)
);
        for (l = keys; l; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_4 ("cob_file_sort_init_key",
cb_ref (name),
                                     CB_PURPOSE (l),
                                     CB_VALUE (l),
                                     cb_int (cb_field (CB_VALUE(l))->offset)))
;
        }
    } else {
        f = CB_FIELD (cb_ref (name));
        if (keys == NULL) {
            cb_error_x (name, _("Table sort without keys not implemen
ted yet"));
        }
        cb_emit (cb_build_funcall_2 ("cob_table_sort_init", cb_int (
cb_list_length (keys)), col));
        for (l = keys; l; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_3 ("cob_table_sort_init_key",
                                     CB_PURPOSE (l),
                                     CB_VALUE (l),
                                     cb_int (cb_field (CB_VALUE(l))->offset)))
;
        }
        cb_emit (cb_build_funcall_2 ("cob_table_sort", name,
                                     (f->occursDepending
                                     ? cb_build_cast_integer (f->
occursDepending)
                                     : cb_int (f->occurs_max))));
    }
}

```

**7.16.4.144 void cb\_emit\_sort\_input ( cb\_tree proc )**

Definition at line 5709 of file typeck.c.

```
{
    cb_emit (cb_build_perform_once (proc));
}
```

**7.16.4.145 void cb\_emit\_sort\_output ( cb\_tree proc )**

Definition at line 5735 of file typeck.c.

```
{
    cb_emit (cb_build_perform_once (proc));
}
```

**7.16.4.146 void cb\_emit\_sort\_using ( cb\_tree file, cb\_tree l )**

Definition at line 5694 of file typeck.c.

```
{
    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {
        if (CB_FILE (cb_ref(CB_VALUE(l)))->organization == COB_ORG_SORT)
        {
            cb_error (_("Invalid SORT USING parameter"));
        }
        cb_emit (cb_build_funcall_2 ("cob_file_sort_using",
            cb_ref (file), cb_ref (CB_VALUE (l))));
    }
}
```

**7.16.4.147 void cb\_emit\_start ( cb\_tree file, cb\_tree op, cb\_tree key )**

Definition at line 5753 of file typeck.c.

```
{
    if (cb_validate_one (key)) {
        return;
    }
    if (file != cb_error_node) {
        current_statement->file = cb_ref (file);
        cb_emit (cb_build_funcall_4 ("cob_start", cb_ref (file), op,
            key ? key : CB_FILE (cb_ref (file))-
            >key,
            CB_FILE(cb_ref(file))->file_statu
            s));
    }
}
```

**7.16.4.148 void cb.emit.stop\_run ( cb\_tree x )**

Definition at line 5771 of file typeck.c.

```
{
    cb_emit (cb_build_funcall_1 ("cob_stop_run", cb_build_cast_integer (x)));
}
```

**7.16.4.149 void cb.emit.string ( cb\_tree items, cb\_tree into, cb\_tree pointer )**

Definition at line 5781 of file typeck.c.

```
{
    cb_tree start;
    cb_tree l;
    cb_tree end;
    cb_tree dlm;

    if (cb_validate_one (into)) {
        return;
    }
    if (cb_validate_one (pointer)) {
        return;
    }
    start = items;
    cb_emit (cb_build_funcall_2 ("cob_string_init", into, pointer));
    while (start) {

        /* find DELIMITED item */
        for (end = start; end; end = CB_CHAIN (end)) {
            if (CB_PAIR_P (CB_VALUE (end))) {
                break;
            }
        }

        /* cob_string_delimited */
        dlm = end ? CB_PAIR_X (CB_VALUE (end)) : cb_int0;
        cb_emit (cb_build_funcall_1 ("cob_string_delimited", dlm));

        /* cob_string_append */
        for (l = start; l != end; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_1 ("cob_string_append",
CB_VALUE (l)));
        }

        start = end ? CB_CHAIN (end) : NULL;
    }
    cb_emit (cb_build_funcall_0 ("cob_string_finish"));
}
```

**7.16.4.150 void cb.emit.unlock ( cb\_tree ref )**

Definition at line 5824 of file typeck.c.

```

{
    cb_tree file;

    if (ref != cb_error_node) {
        file = cb_ref (ref);
        cb_emit (cb_build_funcall_2 ("cob_unlock_file",
            file, CB_FILE(file)->file_status));
        current_statement->file = file;
    }
}

```

#### 7.16.4.151 void cb\_emit\_unstring ( *cb\_tree name*, *cb\_tree delimited*, *cb\_tree into*, *cb\_tree pointer*, *cb\_tree tallying* )

Definition at line 5841 of file typeck.c.

```

{
    if (cb_validate_one (name)) {
        return;
    }
    if (cb_validate_one (tallying)) {
        return;
    }
    if (cb_validate_list (delimited)) {
        return;
    }
    if (cb_validate_list (into)) {
        return;
    }
    cb_emit (cb_build_funcall_3 ("cob_unstring_init", name, pointer,
        cb_int (cb_list_length (delimited))));
    cb_emit_list (delimited);
    cb_emit_list (into);
    if (tallying) {
        cb_emit (cb_build_funcall_1 ("cob_unstring_tallying", tallying));
    }
    cb_emit (cb_build_funcall_0 ("cob_unstring_finish"));
}

```

#### 7.16.4.152 void cb\_emit\_write ( *cb\_tree record*, *cb\_tree from*, *cb\_tree opt*, *cb\_tree lockopt* )

Definition at line 5894 of file typeck.c.

```

{
    cb_tree    file;
    int        val;

    if (record != cb_error_node && cb_ref (record) != cb_error_node) {
        if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
            cb_error_x (CB_TREE (current_statement),
                _("WRITE requires a record name as subject"));
            return;
        }
    }
}

```

```

    if (cb_field (record)->storage != CB_STORAGE_FILE) {
        cb_error_x (CB_TREE (current_statement),
            _("WRITE subject does not refer to a record name"
));
        return;
    }
    file = CB_TREE (CB_FIELD (cb_ref (record))->file);
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    } else if (current_statement->handler_id == COB_EC_I_O_INVALID_KEY
Y &&
        (CB_FILE(file)->organization != COB_ORG_RELATIVE &&
        CB_FILE(file)->organization != COB_ORG_INDEXED)) {
        cb_error_x (CB_TREE(current_statement),
            _("INVALID KEY clause invalid with this file type
"));
    } else if (lockopt) {
        if ((CB_FILE (file)->lock_mode & COB_LOCK_AUTOMATIC)) {
            cb_error_x (CB_TREE (current_statement),
                _("LOCK clause invalid with file LOCK AUTOMATIC"
));
        } else if (opt != cb_int0) {
            cb_error_x (CB_TREE (current_statement),
                _("LOCK clause invalid here"));
        } else if (lockopt == cb_int1) {
            opt = cb_int (COB_WRITE_LOCK);
        }
    }
    if (from) {
        cb_emit (cb_build_move (from, record));
    }
    if (CB_FILE (file)->organization == COB_ORG_LINE_SEQUENTIAL &&
        opt == cb_int0) {
        opt = cb_int (COB_WRITE_BEFORE | COB_WRITE_LINES | 1);
    }
    /* RXW - This is horrible */
    if (current_statement->handler_id == COB_EC_I_O_EOP &&
        current_statement->handler1) {
        if (CB_CAST_P(opt)) {
            val = CB_INTEGER(CB_BINARY_OP(CB_CAST(opt)->val)-
>x)->val;
            val |= COB_WRITE_EOP;
            CB_BINARY_OP (CB_CAST(opt)->val)->x = cb_int (val)
;
        } else {
            val = CB_INTEGER(opt)->val;
            val |= COB_WRITE_EOP;
            opt = cb_int (val);
        }
    }
    cb_emit (cb_build_funcall_4 ("cob_write", file, record, opt,
        CB_FILE(file)->file_status));
}
}

```

#### 7.16.4.153 char\* cb\_encode\_program\_id ( const char \* name )

Definition at line 563 of file typeck.c.

```

{
    unsigned char      *p;
    const unsigned char *s;
    unsigned char      buff[COB_SMALL_BUFFER];

    p = buff;
    s = (const unsigned char *)name;
    /* encode the initial digit */
    if (isdigit (*s)) {
        p += sprintf ((char *)p, "%02X", *s++);
    }
    /* encode invalid letters */
    for (; *s; s++) {
        if (isalnum (*s) || *s == '_') {
            *p++ = *s;
        } else if (*s == '-') {
            *p++ = '_';
            *p++ = '_';
        } else {
            p += sprintf ((char *)p, "%02X", *s);
        }
    }
    *p = 0;
    return strdup ((char *)buff);
}

```

#### 7.16.4.154 void `cb_error_x`( `cb_tree x`, `const char *fmt`, ... )

Definition at line 110 of file `error.c`.

```

{
    va_list ap;

    va_start (ap, fmt);
    print_error ((char *) (x->source_file), x->source_line, "Error: ", fmt, ap);
    va_end (ap);

    errorcount++;
}

```

#### 7.16.4.155 struct `cb_field*` `cb_field`( `cb_tree x` ) [read]

Definition at line 1458 of file `tree.c`.

```

{
    if (CB_REFERENCE_P (x)) {
        return CB_FIELD (cb_ref (x));
    } else {
        return CB_FIELD (x);
    }
}

```

**7.16.4.156** `struct cb_field* cb_field_add ( struct cb_field * f, struct cb_field * p )`  
 [read]

Definition at line 1468 of file tree.c.

```
{
    struct cb_field *t;

    if (f == NULL) {
        return p;
    }
    for (t = f; t->sister; t = t->sister) {
        ;
    }
    t->sister = p;
    return f;
}
```

**7.16.4.157** `struct cb_field* cb_field_founder ( struct cb_field * f )` [read]

Definition at line 1521 of file tree.c.

```
{
    while (f->parent) {
        f = f->parent;
    }
    return f;
}
```

**7.16.4.158** `int cb_field_size ( cb_tree x )`

Definition at line 1483 of file tree.c.

```
{
    struct cb_reference *r;
    struct cb_field *f;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_LITERAL:
        return CB_LITERAL (x)->size;
    case CB_TAG_FIELD:
        return CB_FIELD (x)->size;
    case CB_TAG_REFERENCE:
        r = CB_REFERENCE (x);
        f = CB_FIELD (r->value);

        if (r->length) {
            if (CB_LITERAL_P (r->length)) {
                return cb_get_int (r->length);
            } else {
                return -1;
            }
        }
        } else if (r->offset) {
            if (CB_LITERAL_P (r->offset)) {
```



```

        return f->size - cb_get_int (r->offset) + 1;
    } else {
        return -1;
    }
} else {
    return f->size;
}
default:
    fprintf (stderr, "Unexpected tree tag %d\n", CB_TREE_TAG (x));
    ABORT ();
}
/* NOT REACHED */
return 0;
}

```

#### 7.16.4.159 int cb\_field\_subordinate ( struct cb\_field \* p, struct cb\_field \* f )

Definition at line 1562 of file tree.c.

```

{
    for (p = p->parent; p; p = p->parent) {
        if (p == f) {
            return 1;
        }
    }
    return 0;
}

```

#### 7.16.4.160 struct cb\_field\* cb\_field\_variable\_address ( struct cb\_field \* f ) [read]

Definition at line 1545 of file tree.c.

```

{
    struct cb_field *p;

    for (p = f->parent; p; f = f->parent, p = f->parent) {
        for (p = p->children; p != f; p = p->sister) {
            if (p->occursDepending || cb_field_variable_size (p)) {
                return p;
            }
        }
    }
    return NULL;
}

```

#### 7.16.4.161 struct cb\_field\* cb\_field\_variable\_size ( struct cb\_field \* f ) [read]

Definition at line 1530 of file tree.c.

```

{
    struct cb_field *p;

```

```

    for (f = f->children; f; f = f->sister) {
        if (f->occursDepending) {
            return f;
        } else if ((p = cb_field_variable_size (f)) != NULL) {
            return p;
        }
    }
    return NULL;
}

```

#### 7.16.4.162 int cb\_fits\_int ( cb\_tree x )

Definition at line 587 of file tree.c.

```

{
    struct cb_literal      *l;
    struct cb_field       *f;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_LITERAL:
        l = CB_LITERAL (x);
        if (l->scale <= 0 && l->size < 10) {
            return 1;
        }
        return 0;
    case CB_TAG_FIELD:
        f = CB_FIELD (x);
        switch (f->usage) {
        case CB_USAGE_INDEX:
        case CB_USAGE_LENGTH:
            return 1;
        case CB_USAGE_BINARY:
        case CB_USAGE_COMP_5:
        case CB_USAGE_COMP_X:
            if (f->pic->scale <= 0 && f->size <= (int)sizeof (int)) {
                return 1;
            }
            return 0;
        case CB_USAGE_DISPLAY:
            if (f->size < 10) {
                if (!f->pic || f->pic->scale <= 0) {
                    return 1;
                }
            }
            return 0;
        case CB_USAGE_PACKED:
            if (f->pic->scale <= 0 && f->pic->digits < 10) {
                return 1;
            }
            return 0;
        default:
            return 0;
        }
    case CB_TAG_REFERENCE:
        return cb_fits_int (CB_REFERENCE (x)->value);
    default:
        return 0;
    }
}

```

## 7.16.4.163 int cb\_fits\_long\_long ( cb\_tree x )

Definition at line 635 of file tree.c.

```

{
    struct cb_literal    *l;
    struct cb_field      *f;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_LITERAL:
        l = CB_LITERAL (x);
        if (l->scale <= 0 && l->size < 19) {
            return 1;
        }
        return 0;
    case CB_TAG_FIELD:
        f = CB_FIELD (x);
        switch (f->usage) {
        case CB_USAGE_INDEX:
        case CB_USAGE_LENGTH:
            return 1;
        case CB_USAGE_BINARY:
        case CB_USAGE_COMP_5:
        case CB_USAGE_COMP_X:
            if (f->pic->scale <= 0 && f->size <= (int)sizeof (long lo
ng)) {
                return 1;
            }
            return 0;
        case CB_USAGE_DISPLAY:
            if (f->pic->scale <= 0 && f->size < 19) {
                return 1;
            }
            return 0;
        default:
            return 0;
        }
    case CB_TAG_REFERENCE:
        return cb_fits_long_long (CB_REFERENCE (x)->value);
    default:
        return 0;
    }
}

```

## 7.16.4.164 int cb\_get\_int ( cb\_tree x )

Definition at line 676 of file tree.c.

```

{
    struct cb_literal    *l;
    size_t               i;
    int                  val = 0;

    l = CB_LITERAL (x);
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            break;
        }
    }
}

```

```

    }
/* RXWRXW
   if (l->size - i >= 10) {
       ABORT ();
   }
*/

   for (; i < l->size; i++) {
       val = val * 10 + l->data[i] - '0';
   }
   if (l->sign < 0) {
       val = -val;
   }
   return val;
}

```

#### 7.16.4.165 int cb\_get\_level ( cb\_tree x )

Definition at line 41 of file field.c.

```

{
    const char    *p;
    const char    *name;
    int           level = 0;

    name = CB_NAME (x);
    /* get level */
    for (p = name; *p; p++) {
        if (!isdigit (*p)) {
            goto level_error;
        }
        level = level * 10 + (*p - '0');
    }

    /* check level */
    switch (level) {
    case 66:
    case 77:
    case 78:
    case 88:
        break;
    default:
        if (level < 1 || level > 49) {
            goto level_error;
        }
        break;
    }

    return level;

level_error:
    cb_error_x (x, _("Invalid level number '%s'"), name);
    return 0;
}

```

## 7.16.4.166 long long cb\_get\_long\_long ( cb\_tree x )

Definition at line 705 of file tree.c.

```

{
    struct cb_literal    *l;
    size_t              i;
    long long           val = 0;

    l = CB_LITERAL (x);
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            break;
        }
    }

    if (l->size - i >= 19) {
        ABORT ();
    }

    for (; i < l->size; i++) {
        val = val * 10 + l->data[i] - '0';
    }
    if (l->sign < 0) {
        val = -val;
    }
    return val;
}

```

## 7.16.4.167 void cb\_init\_constants ( void )

Definition at line 732 of file tree.c.

```

{
    char    *s;
    int     i;

    cb_error_node = make_constant (CB_CATEGORY_UNKNOWN, NULL);
    cb_any = make_constant (CB_CATEGORY_UNKNOWN, NULL);
    cb_true = make_constant (CB_CATEGORY_BOOLEAN, "1");
    cb_false = make_constant (CB_CATEGORY_BOOLEAN, "0");
    cb_null = make_constant (CB_CATEGORY_DATA_POINTER, "0");
    cb_zero = make_constant (CB_CATEGORY_NUMERIC, "&cob_zero");
    cb_one = make_constant (CB_CATEGORY_NUMERIC, "&cob_one");
    cb_space = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_space");
    cb_low = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_low");
    cb_norm_low = cb_low;
    cb_high = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_high");
    cb_norm_high = cb_high;
    cb_quote = make_constant (CB_CATEGORY_ALPHANUMERIC, "&cob_quote");
    cb_int0 = cb_int (0);
    cb_int1 = cb_int (1);
    cb_int2 = cb_int (2);
    cb_int3 = cb_int (3);
    cb_int4 = cb_int (4);
    cb_int5 = cb_int (5);
    for (i = 1; i < 8; i++) {
        s = cobc_malloc (4);
    }
}

```

```

        sprintf (s, "i%d", i);
        cb_i[i] = make_constant (CB_CATEGORY_NUMERIC, s);
    }
    cb_standard_error_handler = make_constant_label ("Default Error Handler")
;
}

```

#### 7.16.4.168 void cb\_init\_reserved ( void )

Definition at line 1019 of file reserved.c.

```

{
    int    i;

    /* build system-name table */
    for (i = 0; system_table[i].name != NULL; ++i) {
        system_table[i].node =
            cb_build_system_name (system_table[i].category, system_table[i]
.token);
    }
}

```

#### 7.16.4.169 void cb\_init\_tarrying ( void )

Definition at line 3754 of file typeck.c.

```

{
    inspect_func = NULL;
    inspect_data = NULL;
}

```

#### 7.16.4.170 cb\_tree cb\_int ( int n )

Definition at line 881 of file tree.c.

```

{
    struct cb_integer    *x;
    struct int_node      *p;

    for (p = int_node_table; p; p = p->next) {
        if (p->n == n) {
            return p->node;
        }
    }

    x = make_tree (CB_TAG_INTEGER, CB_CATEGORY_NUMERIC, sizeof (struct
cb_integer));
    x->val = n;

    p = cobc_malloc (sizeof (struct int_node));
    p->n = n;
    p->node = CB_TREE (x);
}

```

```

    p->next = int_node_table;
    int_node_table = p;
    return p->node;
}

```

#### 7.16.4.171 `cb_tree cb_list.add ( cb_tree l, cb_tree x )`

Definition at line 798 of file tree.c.

```

{
    return cb_list_append (l, cb_list_init (x));
}

```

#### 7.16.4.172 `cb_tree cb_list.append ( cb_tree l1, cb_tree l2 )`

Definition at line 781 of file tree.c.

```

{
    cb_tree l;

    if (l1 == NULL) {
        return l2;
    } else {
        l = l1;
        while (CB_CHAIN (l)) {
            l = CB_CHAIN (l);
        }
        CB_CHAIN (l) = l2;
        return l;
    }
}

```

#### 7.16.4.173 `void cb_list.intrinsics ( void )`

Definition at line 979 of file reserved.c.

```

{
    const char    *s;
    size_t        i;
    size_t        n;

    printf ("Intrinsic Function (Implemented Y/N)\n\n");
    for (i = 0; i < NUM_INTRINSICS; ++i) {
        n = strlen (function_list[i].name);
        switch (n / 8) {
            case 0:
                s = "\t\t\t\t";
                break;
            case 1:
                s = "\t\t\t";
                break;
            case 2:

```

```

        s = "\t\t";
        break;
    default:
        s = "\t";
        break;
    }
    printf ("%s%s(%s)\n", function_list[i].name, s,
           function_list[i].implemented ? "Y" : "N");
}
}

```

#### 7.16.4.174 int cb.list\_length ( cb\_tree l )

Definition at line 818 of file tree.c.

```

{
    int n = 0;

    for (; l; l = CB_CHAIN (l)) {
        n++;
    }
    return n;
}

```

#### 7.16.4.175 void cb.list\_map ( cb\_tree\*(\*)(cb\_tree x) func, cb\_tree l )

Definition at line 829 of file tree.c.

```

{
    for (; l; l = CB_CHAIN (l)) {
        CB_VALUE (l) = func (CB_VALUE (l));
    }
}

```

#### 7.16.4.176 void cb.list\_mnemonics ( void )

Definition at line 1008 of file reserved.c.

```

{
    size_t      i;

    printf ("Mnemonic names\n\n");
    for (i = 0; system_table[i].name != NULL; ++i) {
        printf ("%s\n", system_table[i].name);
    }
}

```



**7.16.4.177 void cb\_list\_reserved ( void )**

Definition at line 950 of file reserved.c.

```
{
    const char    *s;
    size_t    i;
    size_t    n;

    printf ("Reserved Words (Parsed Y/N)\n\n");
    for (i = 0; i < NUM_RESERVED_WORDS; ++i) {
        n = strlen (reserved_words[i].name);
        switch (n / 8) {
            case 0:
                s = "\t\t\t\t";
                break;
            case 1:
                s = "\t\t\t";
                break;
            case 2:
                s = "\t\t";
                break;
            default:
                s = "\t";
                break;
        }
        printf ("%s%s(%s)\n", reserved_words[i].name, s,
            reserved_words[i].token != -1 ? "Y" : "N");
    }
}
```

**7.16.4.178 cb\_tree cb\_list\_reverse ( cb\_tree l )**

Definition at line 804 of file tree.c.

```
{
    cb_tree next;
    cb_tree last = NULL;

    for (; l; l = next) {
        next = CB_CHAIN (l);
        CB_CHAIN (l) = last;
        last = l;
    }
    return last;
}
```

**7.16.4.179 char\* cb\_name ( cb\_tree x )**

Definition at line 441 of file tree.c.

```
{
    if (!treenamebuff) {
        treenamebuff = cobc_malloc (COB_NORMAL_BUFF);
    }
}
```

```

    }
    cb_name_1 (treenamebuff, x);
    return treenamebuff;
}

```

#### 7.16.4.180 `cb_tree cb.ref ( cb_tree x )`

Definition at line 1794 of file tree.c.

```

{
    struct cb_reference    *r;
    struct cb_field       *p;
    struct cb_label       *s;
    cb_tree               candidate = NULL;
    cb_tree               items;
    cb_tree               cb1;
    cb_tree               cb2;
    cb_tree               v;
    cb_tree               c;
    struct cb_program     *prog;
    struct cb_word        *w;
    size_t                val;
    size_t                ambiguous = 0;

    r = CB_REFERENCE (x);
    /* if this reference has already been resolved (and the value
       has been cached), then just return the value */
    if (r->value) {
        return r->value;
    }
    /* resolve the value */

    items = r->word->items;
    for (; items; items = CB_CHAIN (items)) {
        /* find a candidate value by resolving qualification */
        v = CB_VALUE (items);
        c = r->chain;
        switch (CB_TREE_TAG (v)) {
            case CB_TAG_FIELD:
                /* in case the value is a field, it might be qualified
                   by its parent names and a file name */
                if (CB_FIELD (v)->flag_indexed_by) {
                    p = CB_FIELD (v)->index_qual;
                } else {
                    p = CB_FIELD (v)->parent;
                }
                /* resolve by parents */
                for (; p; p = p->parent) {
                    if (c && strcasecmp (CB_NAME (c), p->name) == 0)
                    {
                        c = CB_REFERENCE (c)->chain;
                    }
                }

                /* resolve by file */
                if (c && CB_REFERENCE (c)->chain == NULL) {
                    if (CB_REFERENCE (c)->word->count == 1 &&
                        CB_FILE_P (cb_ref (c))
                            && (CB_FILE (cb_ref (c)) == cb_field_founder

```

```

(CB_FIELD (v)->file)) {
    c = CB_REFERENCE (c)->chain;
    }
    }
    break;
case CB_TAG_LABEL:
    /* in case the value is a label, it might be qualified
       by its section name */
    s = CB_LABEL (v)->section;

    /* unqualified paragraph name referenced within the section
       is resolved without ambiguity check if not duplicated
       */
    if (c == NULL && r->offset && s == CB_LABEL (r->offset))
    {
        for (cb1 = CB_CHAIN (items); cb1; cb1 = CB_CHAIN
(cb1)) {
            cb2 = CB_VALUE (cb1);
            if (s == CB_LABEL (cb2)->section) {
                ambiguous_error (x);
                goto error;
            }
            candidate = v;
            goto end;
        }

        /* resolve by section name */
        if (c && s && strcmp (CB_NAME (c), (char *)s->name) =
= 0) {
            c = CB_REFERENCE (c)->chain;
        }

        break;
default:
    /* other values cannot be qualified */
    break;
}

/* a well qualified value is a good candidate */
if (c == NULL) {
    if (candidate == NULL) {
        /* keep the first candidate */
        candidate = v;
    } else {
        /* multiple candidates and possibly ambiguous */
        ambiguous = 1;
        /* continue search because the reference might not
           be ambiguous and exit loop by "goto end" later
           */
    }
}

/* there is no candidate */
if (candidate == NULL) {
    if (current_program->nested_level > 0) {
        /* Nested program - check parents for GLOBAL candidate */

```

```

        ambiguous = 0;
        val = hash ((const unsigned char *)r->word->name);
        prog = current_program->next_program;
        for (; prog; prog = prog->next_program) {
            if (prog->nested_level >= current_program->
nested_level) {
                continue;
            }
            for (w = prog->word_table[val]; w; w = w->next) {
                if (strcasemp (r->word->name, w->name) =
= 0) {
                    candidate = global_check (r, w->
items, &ambiguous);
                    if (candidate) {
                        if (ambiguous) {
                            ambiguous_error (
x);
                            goto error;
                        }
                        if (CB_FILE_P(candidate))
                            current_program->
gen_file_error = 1;
                        goto end;
                    }
                }
            }
            if (prog->nested_level == 0) {
                break;
            }
        }
        undefined_error (x);
        goto error;
    }
    /* the reference is ambiguous */
    if (ambiguous) {
        ambiguous_error (x);
        goto error;
    }
end:
    if (CB_FIELD_P (candidate)) {
        CB_FIELD (candidate)->count++;
        if (CB_FIELD (candidate)->flag_invalid) {
            goto error;
        }
    }
    r->value = candidate;
    return r->value;

error:
    r->value = cb_error_node;
    return cb_error_node;
}

```

7.16.4.181 void `cb_reset_78`( void )

7.16.4.182 void `cb_reset_in_procedure`( void )

7.16.4.183 struct `cb_field`\* `cb_resolve_redefines`( struct `cb_field` \* *field*, *cb\_tree redefines*  
) [read]

Definition at line 231 of file `field.c`.

```
{
    struct cb_field      *f;
    struct cb_reference  *r;
    const char          *name;
    cb_tree             x;

    r = CB_REFERENCE (redefines);
    name = CB_NAME (redefines);
    x = CB_TREE (field);

    /* check qualification */
    if (r->chain) {
        cb_error_x (x, _("%s' cannot be qualified here"), name);
        return NULL;
    }

    /* check subscripts */
    if (r->subs) {
        cb_error_x (x, _("%s' cannot be subscripted here"), name);
        return NULL;
    }

    /* resolve the name in the current group (if any) */
    if (field->parent && field->parent->children) {
        for (f = field->parent->children; f; f = f->sister) {
            if (strcasecmp (f->name, name) == 0) {
                break;
            }
        }
        if (f == NULL) {
            cb_error_x (x, _("%s' undefined in '%s'"), name, field->
parent->name);
            return NULL;
        }
    } else {
        if (cb_ref (redefines) == cb_error_node) {
            return NULL;
        }
        f = cb_field (redefines);
    }

    /* check level number */
    if (f->level != field->level) {
        cb_error_x (x, _("Level number of REDEFINES entries must be ident
ical"));
        return NULL;
    }
    if (f->level == 66 || f->level == 88) {
        cb_error_x (x, _("Level number of REDEFINES entry cannot be 66 or
88"));
        return NULL;
    }
}
```

```

    }

    if (!cb_indirect_redefines && f->redefines) {
        cb_error_x (x, _("%s' not the original definition"), f->name);
        return NULL;
    }

    /* return the original definition */
    while (f->redefines) {
        f = f->redefines;
    }
    return f;
}

```

**7.16.4.184 void cb\_set\_in\_procedure ( void )**

**7.16.4.185 enum cb\_category cb\_tree.category ( cb\_tree x )**

Definition at line 458 of file tree.c.

```

{
    struct cb_cast      *p;
    struct cb_reference *r;
    struct cb_field     *f;

    if (x == cb_error_node) {
        return 0;
    }
    if (x->category != CB_CATEGORY_UNKNOWN) {
        return x->category;
    }

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_CAST:
        p = CB_CAST (x);
        switch (p->type) {
        case CB_CAST_ADDRESS:
        case CB_CAST_ADDR_OF_ADDR:
            x->category = CB_CATEGORY_DATA_POINTER;
            break;
        case CB_CAST_PROGRAM_POINTER:
            x->category = CB_CATEGORY_PROGRAM_POINTER;
            break;
        default:
            fprintf (stderr, "Unexpected cast type -> %d\n", p->type)
;
            ABORT ();
        }
        break;
    case CB_TAG_REFERENCE:
        r = CB_REFERENCE (x);
        if (r->offset) {
            x->category = CB_CATEGORY_ALPHANUMERIC;
        } else {
            x->category = cb_tree_category (r->value);
        }
        break;
    case CB_TAG_FIELD:

```

```

    f = CB_FIELD (x);
    if (f->children) {
        x->category = CB_CATEGORY_ALPHANUMERIC;
    } else if (f->usage == CB_USAGE_POINTER && f->level != 88) {
        x->category = CB_CATEGORY_DATA_POINTER;
    } else if (f->usage == CB_USAGE_PROGRAM_POINTER && f->level != 88
) {
        x->category = CB_CATEGORY_PROGRAM_POINTER;
    } else {
        switch (f->level) {
            case 66:
                if (f->rename_thru) {
                    x->category = CB_CATEGORY_ALPHANUMERIC;
                } else {
                    x->category = cb_tree_category (CB_TREE (
f->redefines));
                }
                break;
            case 88:
                x->category = CB_CATEGORY_BOOLEAN;
                break;
            default:
                x->category = f->pic->category;
                break;
        }
        break;
    case CB_TAG_ALPHABET_NAME:
    case CB_TAG_LOCALE_NAME:
        x->category = CB_CATEGORY_ALPHANUMERIC;
        break;
    case CB_TAG_BINARY_OP:
        x->category = CB_CATEGORY_BOOLEAN;
        break;
    default:
        fprintf (stderr, "Unknown tree tag %d Category %d\n",
CB_TREE_TAG (x), x->category);
        ABORT ();
    }
    return x->category;
}

```

#### 7.16.4.186 enum cb\_class cb\_tree\_class ( cb\_tree x )

Definition at line 451 of file tree.c.

```

{
    return category_to_class_table[CB_TREE_CATEGORY (x)];
}

```

#### 7.16.4.187 int cb\_tree\_type ( cb\_tree x )

Definition at line 537 of file tree.c.

```

{
    struct cb_field *f;

    f = cb_field (x);
    if (f->children) {
        return COB_TYPE_GROUP;
    }

    switch (CB_TREE_CATEGORY (x)) {
    case CB_CATEGORY_ALPHABETIC:
    case CB_CATEGORY_ALPHANUMERIC:
        return COB_TYPE_ALPHANUMERIC;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        return COB_TYPE_ALPHANUMERIC_EDITED;
    case CB_CATEGORY_NUMERIC:
        switch (f->usage) {
        case CB_USAGE_DISPLAY:
            return COB_TYPE_NUMERIC_DISPLAY;
        case CB_USAGE_BINARY:
        case CB_USAGE_COMP_5:
        case CB_USAGE_COMP_X:
        case CB_USAGE_INDEX:
        case CB_USAGE_LENGTH:
            return COB_TYPE_NUMERIC_BINARY;
        case CB_USAGE_FLOAT:
            return COB_TYPE_NUMERIC_FLOAT;
        case CB_USAGE_DOUBLE:
            return COB_TYPE_NUMERIC_DOUBLE;
        case CB_USAGE_PACKED:
            return COB_TYPE_NUMERIC_PACKED;
        default:
            fprintf (stderr, "Unexpected numeric usage -> %d\n", f->
usage);
                ABORT ();
            }
        case CB_CATEGORY_NUMERIC_EDITED:
            return COB_TYPE_NUMERIC_EDITED;
        case CB_CATEGORY_OBJECT_REFERENCE:
        case CB_CATEGORY_DATA_POINTER:
        case CB_CATEGORY_PROGRAM_POINTER:
            return COB_TYPE_NUMERIC_BINARY;
        default:
            fprintf (stderr, "Unexpected category -> %d\n", CB_TREE_CATEGORY
(x));
                ABORT ();
            }
    }
    /* NOT REACHED */
    return 0;
}

```

#### 7.16.4.188 struct cb\_field\* cb.validate\_78.item ( struct cb\_field \* p ) [read]

Definition at line 1033 of file field.c.

```

{
    cb_tree x;

    x = CB_TREE (f);
    if (!f->values) {

```



```
        level_require_error (x, "VALUE");
    }

    if (f->pic || f->flag_occurs) {
        level_except_error (x, "VALUE");
    }
    cb_add_78 (f);
    return last_real_field;
}
```

#### 7.16.4.189 void cb\_validate\_88\_item ( struct cb\_field \* p )

Definition at line 1018 of file field.c.

```
{
    cb_tree x;

    x = CB_TREE (f);
    if (!f->values) {
        level_require_error (x, "VALUE");
    }

    if (f->pic || f->flag_occurs) {
        level_except_error (x, "VALUE");
    }
}
```

#### 7.16.4.190 void cb\_validate\_field ( struct cb\_field \* p )

Definition at line 973 of file field.c.

```
{
    struct cb_field      *c;

    if (validate_field_1 (f) != 0) {
        f->flag_invalid = 1;
        return;
    }
    /* RXW - Remove */
    if (f->flag_item_78) {
        f->flag_is_verified = 1;
        return;
    }

    /* setup parameters */
    if (f->storage == CB_STORAGE_LOCAL ||
        f->storage == CB_STORAGE_LINKAGE ||
        f->flag_item_based) {
        f->flag_local = 1;
    }
    if (f->storage == CB_STORAGE_LINKAGE || f->flag_item_based) {
        f->flag_base = 1;
    }
    setup_parameters (f);
}
```

```

/* compute size */
compute_size (f);
if (!f->redefines) {
    f->memory_size = f->size * f->occurs_max;
} else if (f->redefines->memory_size < f->size * f->occurs_max) {
    f->redefines->memory_size = f->size * f->occurs_max;
}

validate_field_value (f);
if (f->flag_is_global) {
    f->count++;
    for (c = f->children; c; c = c->sister) {
        c->flag_is_global = 1;
        c->count++;
    }
}
f->flag_is_verified = 1;
}

```

#### 7.16.4.191 void cb\_validate\_program\_body ( struct cb\_program \* prog )

Definition at line 1416 of file typeck.c.

```

{
    /* resolve all labels */
    cb_tree l;
    cb_tree x;
    cb_tree v;

    for (l = cb_list_reverse (prog->label_list); l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        v = cb_ref (x);
        if (CB_LABEL_P (v)) {
            CB_LABEL (v)->need_begin = 1;
            if (CB_REFERENCE (x)->length) {
                CB_LABEL (v)->need_return = 1;
            }
        } else if (v != cb_error_node) {
            cb_error_x (x, _("%s' not procedure name"), cb_name (x))
        }
    }

    prog->file_list = cb_list_reverse (prog->file_list);
    prog->exec_list = cb_list_reverse (prog->exec_list);
}

```

#### 7.16.4.192 void cb\_validate\_program\_data ( struct cb\_program \* prog )

Definition at line 1286 of file typeck.c.

```

{
    cb_tree l;
    cb_tree x;
    cb_tree assign;
}

```

```

struct cb_field *p;
struct cb_file *f;
unsigned char *c;

for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
    f = CB_FILE (CB_VALUE (l));
    if (!f->finalized) {
        finalize_file (f, NULL);
    }
}
/* build undeclared assignment name now */
if (cb_assign_clause == CB_ASSIGN_MF) {
    for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
        assign = CB_FILE (CB_VALUE (l))->assign;
        if (!assign) {
            continue;
        }
        if (CB_REFERENCE_P (assign)) {
            for (x = current_program->file_list; x; x =
CB_CHAIN (x)) {
                if (!strcmp (CB_FILE (CB_VALUE (x))->
name,
                    CB_REFERENCE (assign)->word->name))
                {
                    redefinition_error (assign);
                }
            }
            p = check_level_78 (CB_REFERENCE (assign)->word->
name);
            if (p) {
                c = (unsigned char *)CB_LITERAL(CB_VALUE (
p->values))->data;
                assign = CB_TREE (build_literal (
CB_CATEGORY_ALPHANUMERIC, c, strlen ((char *)c));
                CB_FILE (CB_VALUE (l))->assign = assign;
            }
        }
        if (CB_REFERENCE_P (assign) && CB_REFERENCE (assign)->wor
d->count == 0) {
            if (cb_warn_implicit_define) {
                cb_warning (_("'s' will be implicitly de
fined"), CB_NAME (assign));
            }
            x = cb_build_implicit_field (assign,
COB_SMALL_BUFF);
            p = current_program->working_storage;
            CB_FIELD (x)->count++;
            if (p) {
                while (p->sister) {
                    p = p->sister;
                }
                p->sister = CB_FIELD (x);
            } else {
                current_program->working_storage =
CB_FIELD (x);
            }
        }
    }
    if (CB_REFERENCE_P (assign)) {
        x = cb_ref (assign);
        if (CB_FIELD_P (x) && CB_FIELD (x)->level == 88)
        {
            cb_error_x (assign, _("ASSIGN data item '

```

```

%s' invalid"), CB_NAME (assign));
    }
}

if (prog->cursor_pos) {
    x = cb_ref (prog->cursor_pos);
    if (x == cb_error_node) {
        prog->cursor_pos = NULL;
    } else if (CB_FIELD(x)->size != 6 && CB_FIELD(x)->size != 4) {
        cb_error_x (prog->cursor_pos, _("%s' CURSOR is not 4 or
6 characters long"),
                    cb_name (prog->cursor_pos));
        prog->cursor_pos = NULL;
    }
}

if (prog->crt_status) {
    x = cb_ref (prog->crt_status);
    if (x == cb_error_node) {
        prog->crt_status = NULL;
    } else if (CB_FIELD(x)->size != 4) {
        cb_error_x (prog->crt_status, _("%s' CRT STATUS is not 4
characters long"),
                    cb_name (prog->crt_status));
        prog->crt_status = NULL;
    }
} else {
    l = cb_build_reference ("COB-CRT-STATUS");
    p = CB_FIELD (cb_build_field (l));
    p->usage = CB_USAGE_DISPLAY;
    p->pic = CB_PICTURE (cb_build_picture ("9(4)"));
    cb_validate_field (p);
    p->flag_no_init = 1;
    /* Do not initialize/bump ref count here
    p->values = cb_list_init (cb_zero);
    p->count++;
    */
    current_program->working_storage =
        cb_field_add (current_program->working_storage, p);
    prog->crt_status = 1;
    /* RXWRXW - Maybe better
    prog->crt_status = cb_build_index (cb_build_reference ("COB-CRT-S
TATUS"), cb_zero, 0, NULL);
    */
}

/* resolve all references so far */
for (l = cb_list_reverse (prog->reference_list); l; l = CB_CHAIN (l)) {
    cb_ref (CB_VALUE (l));
}

for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
    f = CB_FILE (CB_VALUE (l));
    if (f->recordDepending && f->recordDepending != cb_error_node)
    {
        x = f->recordDepending;
        if (cb_ref (x) != cb_error_node) {
            if (CB_REF_OR_FIELD_P(x)) {
                p = cb_field (x);
                switch (p->storage) {
                    case CB_STORAGE_WORKING:

```

```

        case CB_STORAGE_LOCAL:
        case CB_STORAGE_LINKAGE:
            break;
        default:
            cb_error (_("RECORD DEPENDING item
must be in WORKING/LOCAL/LINKAGE section"));
    } else {
*/
        if (!CB_REFERENCE_P(x) && !CB_FIELD_P(x)) {
            cb_error (_("Invalid RECORD DEPENDING item"));
        }
    }
}
}
}

```

#### 7.16.4.193 void cb\_validate\_program\_environment ( struct cb\_program \* prog )

Definition at line 1079 of file typeck.c.

```

{
    cb_tree      x;
    cb_tree      y;
    cb_tree      l;
    cb_tree      ls;
    struct cb_alphabet_name *ap;
    unsigned char *data;
    size_t       dupls;
    size_t       unvals;
    size_t       count;
    int          lower;
    int          upper;
    int          size;
    int          n;
    int          i;
    int          lastval;
    int          values[256];

    /* Check ALPHABET clauses */
    for (l = current_program->alphabet_name_list; l; l = CB_CHAIN (l)) {
        ap = CB_ALPHABET_NAME (CB_VALUE (l));
        if (ap->type != CB_ALPHABET_CUSTOM) {
            continue;
        }
        ap->low_val_char = 0;
        ap->high_val_char = 255;
        dupls = 0;
        unvals = 0;
        count = 0;
        lastval = 0;
        for (n = 0; n < 256; n++) {
            values[n] = -1;
        }
        for (y = ap->custom_list; y; y = CB_CHAIN (y)) {
            if (count > 255) {
                unvals = 1;
                break;
            }
        }
    }
}

```

```

}
x = CB_VALUE (y);
if (CB_PAIR_P (x)) {
    /* X THRU Y */
    lower = get_value (CB_PAIR_X (x));
    upper = get_value (CB_PAIR_Y (x));
    lastval = upper;
    if (!count) {
        ap->low_val_char = lower;
    }
    if (lower < 0 || lower > 255) {
        unvals = 1;
        continue;
    }
    if (upper < 0 || upper > 255) {
        unvals = 1;
        continue;
    }
    if (lower <= upper) {
        for (i = lower; i <= upper; i++) {
            if (values[i] != -1) {
                dupls = 1;
            }
            values[i] = i;
            count++;
        }
    } else {
        for (i = lower; i >= upper; i--) {
            if (values[i] != -1) {
                dupls = 1;
            }
            values[i] = i;
            count++;
        }
    }
} else if (CB_LIST_P (x)) {
    /* X ALSO Y ... */
    if (!count) {
        ap->low_val_char = get_value (CB_VALUE (x
));

    }
    for (ls = x; ls; ls = CB_CHAIN (ls)) {
        n = get_value (CB_VALUE (ls));
        if (!CB_CHAIN (ls)) {
            lastval = n;
        }
        if (n < 0 || n > 255) {
            unvals = 1;
            continue;
        }
        if (values[n] != -1) {
            dupls = 1;
        }
        values[n] = n;
        count++;
    }
} else {
    /* literal */
    if (CB_TREE_CLASS (x) == CB_CLASS_NUMERIC) {
        n = get_value (x);
        lastval = n;
        if (!count) {

```

```

        ap->low_val_char = n;
    }
    if (n < 0 || n > 255) {
        unvals = 1;
        continue;
    }
    if (values[n] != -1) {
        dupls = 1;
    }
    values[n] = n;
    count++;
} else if (CB_LITERAL_P (x)) {
    size = (int)CB_LITERAL (x)->size;
    data = CB_LITERAL (x)->data;
    if (!count) {
        ap->low_val_char = data[0];
    }
    lastval = data[size - 1];
    for (i = 0; i < size; i++) {
        n = data[i];
        if (values[n] != -1) {
            dupls = 1;
        }
        values[n] = n;
        count++;
    }
} else {
    n = get_value (x);
    lastval = n;
    if (!count) {
        ap->low_val_char = n;
    }
    if (n < 0 || n > 255) {
        unvals = 1;
        continue;
    }
    if (values[n] != -1) {
        dupls = 1;
    }
    values[n] = n;
    count++;
}
}
}
if (dupls || unvals) {
    if (dupls) {
        cb_error_x (1, _("Duplicate character values in a
alphabet '%s'"),
                    cb_name (CB_VALUE(1)));
    }
    if (unvals) {
        cb_error_x (1, _("Invalid character values in alp
habet '%s'"),
                    cb_name (CB_VALUE(1)));
    }
    ap->low_val_char = 0;
    ap->high_val_char = 255;
    continue;
}
/* Calculate HIGH-VALUE */
/* If all 256 values have been specified, HIGH-VALUE is the last
one */

```

```

        /* Otherwise if HIGH-VALUE has been specified, find the highest *
        /
        /* value that has not been used */
        if (count == 256) {
            ap->high_val_char = lastval;
        } else if (values[255] != -1) {
            for (n = 254; n >= 0; n--) {
                if (values[n] == -1) {
                    ap->high_val_char = n;
                    break;
                }
            }
        }
    }
    /* Rest HIGH/LOW-VALUES */
    cb_low = cb_norm_low;
    cb_high = cb_norm_high;
    /* resolve the program collating sequence */
    if (!prog->collating_sequence) {
        return;
    }
    x = cb_ref (prog->collating_sequence);
/* RXWRXW
    if (x == cb_error_node) {
        prog->collating_sequence = NULL;
        return;
    }
*/
    if (!CB_ALPHABET_NAME_P (x)) {
        cb_error_x (prog->collating_sequence, _("%s' not alphabet name")
        ,
            cb_name (prog->collating_sequence));
        prog->collating_sequence = NULL;
        return;
    }
    if (CB_ALPHABET_NAME (x)->type != CB_ALPHABET_CUSTOM) {
        return;
    }
    if (CB_ALPHABET_NAME (x)->low_val_char) {
        cb_low = cb_build_alphanumeric_literal ((ucharptr)"\0", 1);
        CB_LITERAL(cb_low)->data[0] = CB_ALPHABET_NAME (x)->low_val_char;

        CB_LITERAL(cb_low)->all = 1;
    }
    if (CB_ALPHABET_NAME (x)->high_val_char != 255){
        cb_high = cb_build_alphanumeric_literal ((ucharptr)"\0", 1);
        CB_LITERAL(cb_high)->data[0] = CB_ALPHABET_NAME (x)->high_val_cha
r;
        CB_LITERAL(cb_high)->all = 1;
    }
}

```

#### 7.16.4.194 void cb\_warning\_x ( cb\_tree x, const char \* fmt, ... )

Definition at line 98 of file error.c.

```

{
    va_list ap;

```



```

    va_start (ap, fmt);
    print_error ((char *) (x->source_file), x->source_line, "Warning: ", fmt,
ap);
    va_end (ap);

    warningcount++;
}

```

#### 7.16.4.195 char\* check\_filler\_name ( char \* name )

Definition at line 64 of file error.c.

```

{
    if (!memcmp (name, "WORK$", 5)) {
        name = (char *) "FILLER";
    }
    return name;
}

```

#### 7.16.4.196 struct cb\_field\* check\_level\_78 ( const char \* name ) [read]

#### 7.16.4.197 void cobc\_tree\_cast\_error ( cb\_tree x, const char \* filen, const int linenum, const int tagnum )

Definition at line 317 of file cobc.c.

```

{
    fprintf (stderr, "%s:%d: Invalid type cast from '%s'\n",
        filen, linenum, x ? cb_name (x) : "null");
    fprintf (stderr, "Tag 1 %d Tag 2 %d\n", x ? CB_TREE_TAG(x) : 0,
        tagnum);
    (void) longjmp (cob_jmpbuf, 1);
}

```

#### 7.16.4.198 void codegen ( struct cb\_program \* prog, int nested )

Definition at line 4683 of file codegen.c.

```

{
    int                i;
    cb_tree            l;
    struct attr_list   *j;
    struct literal_list *m;
    struct field_list  *k;
    struct call_list   *clp;
    struct base_list   *blp;
    unsigned char      *s;
    struct cb_program  *cp;
    cb_tree            l1;
    cb_tree            l2;
    const char         *prevprog;
}

```

```

time_t          loctime;
char            locbuff[48];

current_prog = prog;
param_id = 0;
stack_id = 0;
num_cob_fields = 0;
progid = 0;
loop_counter = 0;
output_indent_level = 0;
last_line = 0;
needs_exit_prog = 0;
gen_custom = 0;
call_cache = NULL;
label_cache = NULL;
local_cache = NULL;
excp_current_program_id = prog->orig_source_name;
excp_current_section = NULL;
excp_current_paragraph = NULL;
memset ((char *)i_counters, 0, sizeof (i_counters));

output_target = yyout;

if (!nested) {
    gen_ebcdic = 0;
    gen_ebcdic_ascii = 0;
    gen_full_ebcdic = 0;
    gen_native = 0;
    attr_cache = NULL;
    base_cache = NULL;
    literal_cache = NULL;
    field_cache = NULL;

    loctime = time (NULL);
    strftime (locbuff, sizeof(locbuff) - 1, "%b %d %Y %H:%M:%S %Z",
              localtime (&loctime));
    output_header (output_target, locbuff);
    output_header (cb_storage_file, locbuff);
    for (cp = prog; cp; cp = cp->next_program) {
        output_header (cp->local_storage_file, locbuff);
    }

    output_storage ("/* Frame stack declaration */\n");
    output_storage ("struct cob_frame {\n");
    output_storage ("\tint\tperform_through;\n");
#ifdef __GNUC__
    output_storage ("\tint\treturn_address;\n");
#elif COB_USE_SETJMP
    output_storage ("\tjmp_buf\treturn_address;\n");
#else
    output_storage ("\tvoid\t*return_address;\n");
#endif

    output_storage ("};\n\n");
    output_storage ("/* Union for CALL statement */\n");
    output_storage ("union cob_call_union {\n");
    output_storage ("\tvoid *(*funcptr) ();\n");
    output_storage ("\tint (*funcint) ();\n");
    output_storage ("\tvoid *func_void;\n");
    output_storage ("};\n");
    output_storage ("union cob_call_union\tcob_unifunc;\n\n");

    output ("#define __USE_STRING_INLINES 1\n");

```

```

#ifdef _XOPEN_SOURCE_EXTENDED
    output ("#ifndef _XOPEN_SOURCE_EXTENDED\n");
    output ("#define _XOPEN_SOURCE_EXTENDED 1\n");
    output ("#endif\n");
#endif

    output ("#include <stdio.h>\n");
    output ("#include <stdlib.h>\n");
    output ("#include <string.h>\n");
    output ("#include <math.h>\n");
#if COB_USE_SETJMP
    output ("#include <setjmp.h>\n");
#endif
#ifdef WORDS_BIGENDIAN
    output ("#define WORDS_BIGENDIAN 1\n");
#endif
#ifdef HAVE_BUILTIN_EXPECT
    output ("#define HAVE_BUILTIN_EXPECT\n");
#endif
    if (optimize_flag) {
        output ("#define COB_LOCAL_INLINE\n");
    }
    output ("#include <libcob.h>\n\n");

    output ("#define COB_SOURCE_FILE          \"%s\"\n",
cb_source_file);
    output ("#define COB_PACKAGE_VERSION    \"%s\"\n",
PACKAGE_VERSION);
    output ("#define COB_PATCH_LEVEL        %d\n",
PATCH_LEVEL);
    output ("/* Global variables */\n");
    output ("#include \"%s\"\n", cb_storage_file_name);

    for (cp = prog; cp; cp = cp->next_program) {
        if (cp->gen_decset) {
            output ("static void\n");
            output ("cob_decimal_set_int (cob_decimal *d, cons
t int n)\n");
            output ("{\n");
            output ("    mpz_set_si (d->value, n);\n");
            output ("    d->scale = 0;\n");
            output ("}\n\n");
            break;
        }
    }
    for (cp = prog; cp; cp = cp->next_program) {
        if (cp->gen_udecset) {
            output ("static void\n");
            output ("cob_decimal_set_uint (cob_decimal *d, con
st unsigned int n)\n");
            output ("{\n");
            output ("    mpz_set_ui (d->value, n);\n");
            output ("    d->scale = 0;\n");
            output ("}\n\n");
            break;
        }
    }
    for (cp = prog; cp; cp = cp->next_program) {
        if (cp->gen_ptrmanip) {
            output ("static void\n");
            output ("cob_pointer_manip (cob_field *f1, cob_fie
ld *f2, size_t addsub)\n");
            output ("{\n");

```

```

        output("        unsigned char  *tmptr;\n");
        output("        memcpy (&tmptr, f1->data, sizeof(
void *));\n");
        output("        if (addsub) {\n");
        output("            tmptr -= cob_get_int (f2)
;\n");
        output("        } else {\n");
        output("            tmptr += cob_get_int (f2)
;\n");
        output("        }\n");
        output("        memcpy (f1->data, &tmptr, sizeof(
void *));\n");
        output("}\n\n");
        break;
    }
}
output ("/* Function prototypes */\n\n");
for (cp = prog; cp; cp = cp->next_program) {
    /* Build parameter list */
    for (l = cp->entry_list; l; l = CB_CHAIN (l)) {
        for (l1 = CB_VALUE (l); l1; l1 = CB_CHAIN (l1)) {
            for (l2 = cp->parameter_list; l2; l2 =
CB_CHAIN (l2)) {
                if (strcasecmp (cb_field (
CB_VALUE (l1))->name,
                                cb_field (
CB_VALUE (l2))->name) == 0) {
                    break;
                }
            }
            if (l2 == NULL) {
                cp->parameter_list = cb_list_add
(cp->parameter_list, CB_VALUE (l1));
            }
        }
    }
    if (cp->flag_main) {
        output ("int %s ();\n", cp->program_id);
    } else {
        for (l = cp->entry_list; l; l = CB_CHAIN (l)) {
            output_entry_function (cp, l, cp->
parameter_list, 0);
        }
    }
    output ("static int %s_ (const int", cp->program_id);
    if (!cp->flag_chained) {
        for (l = cp->parameter_list; l; l = CB_CHAIN (l))
        {
            output ("    unsigned char *");
        }
    }
    output (");\n");
}
output ("\n");
}

/* Class-names */
if (!prog->nested_level && prog->class_name_list) {
    output ("/* Class names */\n");
    for (l = prog->class_name_list; l; l = CB_CHAIN (l)) {
        output_class_name_definition (CB_CLASS_NAME (CB_VALUE (l)

```

```

));
    }
}

/* Main function */
if (prog->flag_main) {
    output_main_function (prog);
}

/* Functions */
if (!nested) {
    output ("/* Functions */\n\n");
}
for (l = prog->entry_list; l; l = CB_CHAIN (l)) {
    output_entry_function (prog, l, prog->parameter_list, 1);
}
output_internal_function (prog, prog->parameter_list);

if (!prog->next_program) {
    output ("/* End functions */\n\n");
}

if (gen_native || gen_full_ebcdic || gen_ebcdic_ascii || prog->
alphabet_name_list) {
    (void)lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
}

output_target = cb_storage_file;

/* Program local stuff */
if (call_cache) {
    output_local ("\n/* Call pointers */\n");
    for (clp = call_cache; clp; clp = clp->next) {
        output_local ("static union cob_call_union\tcall_%s = { N
ULL };\n", clp->callname);
    }
    output_local ("\n");
}

for (i = 0; i < COB_MAX_SUBSCRIPTS; i++) {
    if (i_counters[i]) {
        output_local ("int\t\t%i%d;\n", i);
    }
}

if (num_cob_fields) {
    output_local ("\n/* Local cob_field items */\n");
    for (i = 0; i < num_cob_fields; i++) {
        output_local ("cob_field\tf%d;\n", i);
    }
    output_local ("\n");
}

/* Skip to next nested program */

if (prog->next_program) {
    codegen (prog->next_program, 1);
    return;
}

/* Finalize the storage file */

```

```

    if (base_cache) {
        output_storage ("\n/* Storage */\n");
        base_cache = list_cache_sort (base_cache, &base_cache_cmp);
        prevprog = NULL;
        for (blp = base_cache; blp; blp = blp->next) {
            if (blp->curr_prog != prevprog) {
                prevprog = blp->curr_prog;
                output_storage ("\n/* PROGRAM-ID : %s */\n", prev
prog);
            }
#ifdef HAVE_ATTRIBUTE_ALIGNED
            output_storage ("static unsigned char %s%d[%d] __attribut
e__((aligned));",
#else
            output_storage ("static unsigned char %s%d[%d];",
#endif
                            CB_PREFIX_BASE, blp->f->id,
                            blp->f->memory_size);
            output_storage ("\t/* %s */\n", blp->f->name);
        }
        output_storage ("\n/* End of storage */\n\n");
    }

    if (attr_cache) {
        output_storage ("\n/* Attributes */\n\n");
        attr_cache = attr_list_reverse (attr_cache);
        for (j = attr_cache; j; j = j->next) {
            output_storage ("static const cob_field_attr %s%d = ",
                            CB_PREFIX_ATTR, j->id);
            output_storage ("%d, %d, %d, %d, ", j->type, j->digits,
                            j->scale, j->flags);
            if (j->pic) {
                output_storage ("\"");
                for (s = j->pic; *s; s += 5) {
                    output_storage ("%c\\%03o\\%03o\\%03o\\%0
3o",
                                    s[0], s[1], s[2], s[3], s[4]);
                }
                output_storage ("\"");
            } else {
                output_storage ("NULL");
            }
            output_storage (");\n");
        }
    }

    if (field_cache) {
        output_storage ("\n/* Fields */\n");
        field_cache = list_cache_sort (field_cache, &field_cache_cmp);
        prevprog = NULL;
        for (k = field_cache; k; k = k->next) {
            if (k->curr_prog != prevprog) {
                prevprog = k->curr_prog;
                output_storage ("\n/* PROGRAM-ID : %s */\n", prev
prog);
            }
            output ("static cob_field %s%d\t= ", CB_PREFIX_FIELD, k->
f->id);

            if (!k->f->flag_local && !k->f->flag_item_external) {
                output_field (k->x);
            } else {
                output ("{");
            }
        }
    }

```

```

        output_size (k->x);
        output ("", NULL, "");
        output_attr (k->x);
        output ("");
    }
    output (";\t/* %s */\n", k->f->name);
}
output_storage ("\n/* End of fields */\n\n");
}
if (literal_cache) {
    output_storage ("/* Constants */\n");
    literal_cache = literal_list_reverse (literal_cache);
    for (m = literal_cache; m; m = m->next) {
        output ("static cob_field %s%d\t= ", CB_PREFIX_CONST, m->
id);
        output_field (m->x);
        output (";\n");
    }
    output ("\n");
}

if (gen_ebcdic) {
    output_storage ("/* EBCDIC translate table */\n");
    output ("static const unsigned char\tcob_a2e[256] = {\n");
    if (alt_ebcdic) {
        output ("\t0x00, 0x01, 0x02, 0x03, 0x37, 0x2D, 0x2E, 0x2F
,\n");
        output ("\t0x16, 0x05, 0x25, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
,\n");
        output ("\t0x10, 0x11, 0x12, 0x13, 0x3C, 0x3D, 0x32, 0x26
,\n");
        output ("\t0x18, 0x19, 0x3F, 0x27, 0x1C, 0x1D, 0x1E, 0x1F
,\n");
        output ("\t0x40, 0x5A, 0x7F, 0x7B, 0x5B, 0x6C, 0x50, 0x7D
,\n");
        output ("\t0x4D, 0x5D, 0x5C, 0x4E, 0x6B, 0x60, 0x4B, 0x61
,\n");
        output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7
,\n");
        output ("\t0xF8, 0xF9, 0x7A, 0x5E, 0x4C, 0x7E, 0x6E, 0x6F
,\n");
        output ("\t0x7C, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7
,\n");
        output ("\t0xC8, 0xC9, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6
,\n");
        output ("\t0xD7, 0xD8, 0xD9, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6
,\n");
        output ("\t0xE7, 0xE8, 0xE9, 0xAD, 0xE0, 0xBD, 0x5F, 0x6D
,\n");
        output ("\t0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87
,\n");
        output ("\t0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96
,\n");
        output ("\t0x97, 0x98, 0x99, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6
,\n");
        output ("\t0xA7, 0xA8, 0xA9, 0xC0, 0x6A, 0xD0, 0xA1, 0x07
,\n");
        output ("\t0x68, 0xDC, 0x51, 0x42, 0x43, 0x44, 0x47, 0x48
,\n");
        output ("\t0x52, 0x53, 0x54, 0x57, 0x56, 0x58, 0x63, 0x67
,\n");
        output ("\t0x71, 0x9C, 0x9E, 0xCB, 0xCC, 0xCD, 0xDB, 0xDD

```

```

, \n" );
, \n" );
output ( "\t0xDF, 0xEC, 0xFC, 0xB0, 0xB1, 0xB2, 0x3E, 0xB4
, \n" );
output ( "\t0x45, 0x55, 0xCE, 0xDE, 0x49, 0x69, 0x9A, 0x9B
, \n" );
output ( "\t0xAB, 0x9F, 0xBA, 0xB8, 0xB7, 0xAA, 0x8A, 0x8B
, \n" );
output ( "\t0xB6, 0xB5, 0x62, 0x4F, 0x64, 0x65, 0x66, 0x20
, \n" );
output ( "\t0x21, 0x22, 0x70, 0x23, 0x72, 0x73, 0x74, 0xBE
, \n" );
output ( "\t0x76, 0x77, 0x78, 0x80, 0x24, 0x15, 0x8C, 0x8D
, \n" );
output ( "\t0x8E, 0x41, 0x06, 0x17, 0x28, 0x29, 0x9D, 0x2A
, \n" );
output ( "\t0x2B, 0x2C, 0x09, 0x0A, 0xAC, 0x4A, 0xAE, 0xAF
, \n" );
output ( "\t0x1B, 0x30, 0x31, 0xFA, 0x1A, 0x33, 0x34, 0x35
, \n" );
output ( "\t0x36, 0x59, 0x08, 0x38, 0xBC, 0x39, 0xA0, 0xBF
, \n" );
output ( "\t0xCA, 0x3A, 0xFE, 0x3B, 0x04, 0xCF, 0xDA, 0x14
, \n" );
output ( "\t0xE1, 0x8F, 0x46, 0x75, 0xFD, 0xEB, 0xEE, 0xED
, \n" );
output ( "\t0x90, 0xEF, 0xB3, 0xFB, 0xB9, 0xEA, 0xBB, 0xFF
\n" );
} else {
/* MF */
output ( "\t0x00, 0x01, 0x02, 0x03, 0x1D, 0x19, 0x1A, 0x1B
, \n" );
output ( "\t0x0F, 0x04, 0x16, 0x06, 0x07, 0x08, 0x09, 0x0A
, \n" );
output ( "\t0x0B, 0x0C, 0x0D, 0x0E, 0x1E, 0x1F, 0x1C, 0x17
, \n" );
output ( "\t0x10, 0x11, 0x20, 0x18, 0x12, 0x13, 0x14, 0x15
, \n" );
output ( "\t0x21, 0x27, 0x3A, 0x36, 0x28, 0x30, 0x26, 0x38
, \n" );
output ( "\t0x24, 0x2A, 0x29, 0x25, 0x2F, 0x2C, 0x22, 0x2D
, \n" );
output ( "\t0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A
, \n" );
output ( "\t0x7B, 0x7C, 0x35, 0x2B, 0x23, 0x39, 0x32, 0x33
, \n" );
output ( "\t0x37, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D
, \n" );
output ( "\t0x5E, 0x5F, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66
, \n" );
output ( "\t0x67, 0x68, 0x69, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F
, \n" );
output ( "\t0x70, 0x71, 0x72, 0x7D, 0x6A, 0x7E, 0x7F, 0x31
, \n" );
output ( "\t0x34, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41
, \n" );
output ( "\t0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49
, \n" );
output ( "\t0x4A, 0x4B, 0x4C, 0x4E, 0x4F, 0x50, 0x51, 0x52
, \n" );
output ( "\t0x53, 0x54, 0x55, 0x56, 0x2E, 0x60, 0x4D, 0x05
, \n" );
output ( "\t0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87

```



```

, \n");
output ("\t0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F
, \n");
output ("\t0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97
, \n");
output ("\t0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F
, \n");
output ("\t0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7
, \n");
output ("\t0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF
, \n");
output ("\t0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7
, \n");
output ("\t0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF
, \n");
output ("\t0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7
, \n");
output ("\t0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF
, \n");
output ("\t0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7
, \n");
output ("\t0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF
, \n");
output ("\t0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7
, \n");
output ("\t0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF
, \n");
output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7
, \n");
output ("\t0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
\n");
}
output ("};\n");
output_storage ("\n");
}
if (gen_full_ebcdic) {
output ("static const unsigned char\tcob_ebcdic[256] = {\n");
output ("\t0x00, 0x01, 0x02, 0x03, 0x37, 0x2D, 0x2E, 0x2F,\n");
output ("\t0x16, 0x05, 0x25, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,\n");
output ("\t0x10, 0x11, 0x12, 0x13, 0x3C, 0x3D, 0x32, 0x26,\n");
output ("\t0x18, 0x19, 0x3F, 0x27, 0x1C, 0x1D, 0x1E, 0x1F,\n");
output ("\t0x40, 0x5A, 0x7F, 0x7B, 0x5B, 0x6C, 0x50, 0x7D,\n");
output ("\t0x4D, 0x5D, 0x5C, 0x4E, 0x6B, 0x60, 0x4B, 0x61,\n");
output ("\t0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7,\n");
output ("\t0xF8, 0xF9, 0x7A, 0x5E, 0x4C, 0x7E, 0x6E, 0x6F,\n");
output ("\t0x7C, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7,\n");
output ("\t0xC8, 0xC9, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6,\n");
output ("\t0xD7, 0xD8, 0xD9, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6,\n");
output ("\t0xE7, 0xE8, 0xE9, 0xAD, 0xE0, 0xBD, 0x5F, 0x6D,\n");
output ("\t0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,\n");
output ("\t0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96,\n");
output ("\t0x97, 0x98, 0x99, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6,\n");
output ("\t0xA7, 0xA8, 0xA9, 0xC0, 0x6A, 0xD0, 0xA1, 0x07,\n");
output ("\t0x68, 0xDC, 0x51, 0x42, 0x43, 0x44, 0x47, 0x48,\n");
output ("\t0x52, 0x53, 0x54, 0x57, 0x56, 0x58, 0x63, 0x67,\n");
output ("\t0x71, 0x9C, 0x9E, 0xCB, 0xCC, 0xCD, 0xDB, 0xDD,\n");
output ("\t0xDF, 0xEC, 0xFC, 0xB0, 0xB1, 0xB2, 0x3E, 0xB4,\n");
output ("\t0x45, 0x55, 0xCE, 0xDE, 0x49, 0x69, 0x9A, 0x9B,\n");
output ("\t0xAB, 0x9F, 0xBA, 0xB8, 0xB7, 0xAA, 0x8A, 0x8B,\n");
output ("\t0xB6, 0xB5, 0x62, 0x4F, 0x64, 0x65, 0x66, 0x20,\n");
output ("\t0x21, 0x22, 0x70, 0x23, 0x72, 0x73, 0x74, 0xBE,\n");
output ("\t0x76, 0x77, 0x78, 0x80, 0x24, 0x15, 0x8C, 0x8D,\n");
}

```

```

        output ("\t0x8E, 0x41, 0x06, 0x17, 0x28, 0x29, 0x9D, 0x2A, \n");
        output ("\t0x2B, 0x2C, 0x09, 0x0A, 0xAC, 0x4A, 0xAE, 0xAF, \n");
        output ("\t0x1B, 0x30, 0x31, 0xFA, 0x1A, 0x33, 0x34, 0x35, \n");
        output ("\t0x36, 0x59, 0x08, 0x38, 0xBC, 0x39, 0xA0, 0xBF, \n");
        output ("\t0xCA, 0x3A, 0xFE, 0x3B, 0x04, 0xCF, 0xDA, 0x14, \n");
        output ("\t0xE1, 0x8F, 0x46, 0x75, 0xFD, 0xEB, 0xEE, 0xED, \n");
        output ("\t0x90, 0xEF, 0xB3, 0xFB, 0xB9, 0xEA, 0xBB, 0xFF \n");
        output ("");
        i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
        output
            ("static cob_field f_ebcdic = { 256, (unsigned char *)cob_ebc
dic, &%s%d }; \n",
            CB_PREFIX_ATTR, i);
        output_storage ("\n");
    }
    if (gen_ebcdic_ascii) {
        output ("static const unsigned char\tcob_ebcdic_ascii[256] = {\n"
);
        output ("\t0x00, 0x01, 0x02, 0x03, 0xEC, 0x09, 0xCA, 0x7F, \n");
        output ("\t0xE2, 0xD2, 0xD3, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, \n");
        output ("\t0x10, 0x11, 0x12, 0x13, 0xEF, 0xC5, 0x08, 0xCB, \n");
        output ("\t0x18, 0x19, 0xDC, 0xD8, 0x1C, 0x1D, 0x1E, 0x1F, \n");
        output ("\t0xB7, 0xB8, 0xB9, 0xBB, 0xC4, 0x0A, 0x17, 0x1B, \n");
        output ("\t0xCC, 0xCD, 0xCF, 0xD0, 0xD1, 0x05, 0x06, 0x07, \n");
        output ("\t0xD9, 0xDA, 0x16, 0xDD, 0xDE, 0xDF, 0xE0, 0x04, \n");
        output ("\t0xE3, 0xE5, 0xE9, 0xEB, 0x14, 0x15, 0x9E, 0x1A, \n");
        output ("\t0x20, 0xC9, 0x83, 0x84, 0x85, 0xA0, 0xF2, 0x86, \n");
        output ("\t0x87, 0xA4, 0xD5, 0x2E, 0x3C, 0x28, 0x2B, 0xB3, \n");
        output ("\t0x26, 0x82, 0x88, 0x89, 0x8A, 0xA1, 0x8C, 0x8B, \n");
        output ("\t0x8D, 0xE1, 0x21, 0x24, 0x2A, 0x29, 0x3B, 0x5E, \n");
        output ("\t0x2D, 0x2F, 0xB2, 0x8E, 0xB4, 0xB5, 0xB6, 0x8F, \n");
        output ("\t0x80, 0xA5, 0x7C, 0x2C, 0x25, 0x5F, 0x3E, 0x3F, \n");
        output ("\t0xBA, 0x90, 0xBC, 0xBD, 0xBE, 0xF3, 0xC0, 0xC1, \n");
        output ("\t0xC2, 0x60, 0x3A, 0x23, 0x40, 0x27, 0x3D, 0x22, \n");
        output ("\t0xC3, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, \n");
        output ("\t0x68, 0x69, 0xAE, 0xAF, 0xC6, 0xC7, 0xC8, 0xF1, \n");
        output ("\t0xF8, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, \n");
        output ("\t0x71, 0x72, 0xA6, 0xA7, 0x91, 0xCE, 0x92, 0xA9, \n");
        output ("\t0xE6, 0x7E, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, \n");
        output ("\t0x79, 0x7A, 0xAD, 0xA8, 0xD4, 0x5B, 0xD6, 0xD7, \n");
        output ("\t0x9B, 0x9C, 0x9D, 0xFA, 0x9F, 0xB1, 0xB0, 0xAC, \n");
        output ("\t0xAB, 0xFC, 0xAA, 0xFE, 0xE4, 0x5D, 0xBF, 0xE7, \n");
        output ("\t0x7B, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, \n");
        output ("\t0x48, 0x49, 0xE8, 0x93, 0x94, 0x95, 0xA2, 0xED, \n");
        output ("\t0x7D, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, \n");
        output ("\t0x51, 0x52, 0xEE, 0x96, 0x81, 0x97, 0xA3, 0x98, \n");
        output ("\t0x5C, 0xF0, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, \n");
        output ("\t0x59, 0x5A, 0xFD, 0xF5, 0x99, 0xF7, 0xF6, 0xF9, \n");
        output ("\t0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, \n");
        output ("\t0x38, 0x39, 0xDB, 0xFB, 0x9A, 0xF4, 0xEA, 0xFF \n");
        output ("");
        i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
        output
            ("static cob_field f_ebcdic_ascii = { 256, (unsigned char *)c
ob_ebcdic_ascii, &%s%d }; \n",
            CB_PREFIX_ATTR, i);
        output_storage ("\n");
    }
    if (gen_native) {
        output ("static const unsigned char\tcob_native[256] = {\n");
        output ("\t0, 1, 2, 3, 4, 5, 6, 7, \n");
        output ("\t8, 9, 10, 11, 12, 13, 14, 15, \n");
    }

```

```

        output ("\t16, 17, 18, 19, 20, 21, 22, 23,\n");
        output ("\t24, 25, 26, 27, 28, 29, 30, 31,\n");
        output ("\t32, 33, 34, 35, 36, 37, 38, 39,\n");
        output ("\t40, 41, 42, 43, 44, 45, 46, 47,\n");
        output ("\t48, 49, 50, 51, 52, 53, 54, 55,\n");
        output ("\t56, 57, 58, 59, 60, 61, 62, 63,\n");
        output ("\t64, 65, 66, 67, 68, 69, 70, 71,\n");
        output ("\t72, 73, 74, 75, 76, 77, 78, 79,\n");
        output ("\t80, 81, 82, 83, 84, 85, 86, 87,\n");
        output ("\t88, 89, 90, 91, 92, 93, 94, 95,\n");
        output ("\t96, 97, 98, 99, 100, 101, 102, 103,\n");
        output ("\t104, 105, 106, 107, 108, 109, 110, 111,\n");
        output ("\t112, 113, 114, 115, 116, 117, 118, 119,\n");
        output ("\t120, 121, 122, 123, 124, 125, 126, 127,\n");
        output ("\t128, 129, 130, 131, 132, 133, 134, 135,\n");
        output ("\t136, 137, 138, 139, 140, 141, 142, 143,\n");
        output ("\t144, 145, 146, 147, 148, 149, 150, 151,\n");
        output ("\t152, 153, 154, 155, 156, 157, 158, 159,\n");
        output ("\t160, 161, 162, 163, 164, 165, 166, 167,\n");
        output ("\t168, 169, 170, 171, 172, 173, 174, 175,\n");
        output ("\t176, 177, 178, 179, 180, 181, 182, 183,\n");
        output ("\t184, 185, 186, 187, 188, 189, 190, 191,\n");
        output ("\t192, 193, 194, 195, 196, 197, 198, 199,\n");
        output ("\t200, 201, 202, 203, 204, 205, 206, 207,\n");
        output ("\t208, 209, 210, 211, 212, 213, 214, 215,\n");
        output ("\t216, 217, 218, 219, 220, 221, 222, 223,\n");
        output ("\t224, 225, 226, 227, 228, 229, 230, 231,\n");
        output ("\t232, 233, 234, 235, 236, 237, 238, 239,\n");
        output ("\t240, 241, 242, 243, 244, 245, 246, 247,\n");
        output ("\t248, 249, 250, 251, 252, 253, 254, 255\n");
        output ("");
        i = lookup_attr (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL, 0);
        output
        ("static cob_field f_native = { 256, (unsigned char *)cob_nat
ive, &%s%d }; \n",
         CB_PREFIX_ATTR, i);
        output_storage ("\n");
    }
}

```

#### 7.16.4.199 void finalize\_file ( struct cb\_file \* f, struct cb\_field \* records )

Definition at line 1611 of file tree.c.

```

{
    struct cb_field *p;
    struct cb_field *v;
    cb_tree      l;
    cb_tree      x;
    char          buff[COB_MINI_BUFF];

    if (f->special) {
        f->organization = COB_ORG_LINE_SEQUENTIAL;
    }
    if (f->fileid_assign && !f->assign) {
        f->assign = cb_build_alphanumeric_literal ((unsigned char *)f->
name,
                                                strlen (f->name));
    }
}

```

```

/* check the record size if it is limited */
for (p = records; p; p = p->sister) {
    if (f->record_min > 0) {
        if (p->size < f->record_min) {
            cb_error (_("Record size too small '%s'", p->
name);
        }
    }
    if (f->record_max > 0) {
        if (p->size > f->record_max) {
            cb_error (_("Record size too large '%s' (%d)",
p->name, p->size);
        }
    }
}

/* compute the record size */
if (f->record_min == 0) {
    if (records) {
        f->record_min = records->size;
    } else {
        f->record_min = 0;
    }
}
for (p = records; p; p = p->sister) {
    v = cb_field_variable_size (p);
    if (v && v->offset + v->size * v->occurs_min < f->record_min) {
        f->record_min = v->offset + v->size * v->occurs_min;
    }
    if (p->size < f->record_min) {
        f->record_min = p->size;
    }
    if (p->size > f->record_max) {
        f->record_max = p->size;
    }
}

if (f->same_clause) {
    for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
        if (CB_FILE (CB_VALUE (l))->same_clause == f->
same_clause) {
            if (CB_FILE (CB_VALUE (l))->finalized) {
                if (f->record_max > CB_FILE (CB_VALUE (l)
->record->memory_size) {
                    CB_FILE (CB_VALUE (l))->record->m
emory_size =
                        f->record_max;
                }
                f->record = CB_FILE (CB_VALUE (l))->recor
d;
            }
            for (p = records; p; p = p->sister) {
                p->file = f;
                p->redefines = f->record;
            }
            for (p = f->record->sister; p; p = p->
sister) {
                if (!p->sister) {
                    p->sister = records;
                    break;
                }
            }
        }
    }
}

```

```

                                f->finalized = 1;
                                return;
                            }
                        }
                    }
}
/* create record */
snprintf (buff, COB_MINI_MAX, "%s_record", f->name);
if (f->record_max == 0) {
    f->record_max = 32;
    f->record_min = 32;
}
if (f->organization == COB_ORG_LINE_SEQUENTIAL) {
    f->record_min = 0;
}
f->record = CB_FIELD (cb_build_implicit_field (cb_build_reference (buff),
                                                f->record_max));
f->record->sister = records;
f->record->count++;
if (f->external) {
    has_external = 1;
    f->record->flag_external = 1;
}

for (p = records; p; p = p->sister) {
    p->file = f;
    p->redefines = f->record;
}
f->finalized = 1;
if (f->linage) {
    snprintf (buff, COB_MINI_MAX, "LC_%s", f->name);
    x = cb_build_field (cb_build_reference (buff));
    CB_FIELD (x)->pic = CB_PICTURE (cb_build_picture ("9(9)"));
    CB_FIELD (x)->usage = CB_USAGE_COMP_5;
    CB_FIELD (x)->values = cb_list_init (cb_zero);
    CB_FIELD (x)->count++;
    cb_validate_field (CB_FIELD (x));
    f->linage_ctr = cb_build_field_reference (CB_FIELD (x), NULL);
    current_program->working_storage =
        cb_field_add (current_program->working_storage, CB_FIELD (x))
;
}
}
}

```

#### 7.16.4.200 void group\_error ( cb\_tree x, const char \* clause )

Definition at line 250 of file error.c.

```

{
    cb_error_x (x, _("Group item '%s' cannot have %s clause"),
        check_filler_name (cb_name (x)), clause);
}

```

#### 7.16.4.201 void level\_except\_error ( cb\_tree x, const char \* clause )

Definition at line 270 of file error.c.

```

{
    cb_error_x (x, _("Level %02d item '%s' cannot have other than %s clause"),
        ,
        cb_field (x)->level, check_filler_name (cb_name (x)), clause)
    ;
}

```

#### 7.16.4.202 void level\_redundant\_error ( cb\_tree x, const char \* clause )

Definition at line 256 of file error.c.

```

{
    cb_error_x (x, _("Level %02d item '%s' cannot have %s clause"),
        cb_field (x)->level, check_filler_name (cb_name (x)), clause)
    ;
}

```

#### 7.16.4.203 void level\_require\_error ( cb\_tree x, const char \* clause )

Definition at line 263 of file error.c.

```

{
    cb_error_x (x, _("Level %02d item '%s' requires %s clause"),
        cb_field (x)->level, check_filler_name (cb_name (x)), clause)
    ;
}

```

#### 7.16.4.204 struct cb\_intrinsic\_table\* lookup\_intrinsic ( const char \* name, const int checkres ) [read]

Definition at line 929 of file reserved.c.

```

{
    struct cb_intrinsic_table      *cbp;
    struct noreserve              *noresp;

    if (checkres) {
        for (noresp = nores; noresp; noresp = noresp->next) {
            if (strcasemp (name, noresp->noresword) == 0) {
                return NULL;
            }
        }
    }
    cbp = bsearch (name, function_list, NUM_INTRINSICS,
        sizeof (struct cb_intrinsic_table), intrinsic_comp);
    if (cbp && cbp->implemented) {
        return cbp;
    }
    return NULL;
}

```

**7.16.4.205 int lookup\_reserved\_word ( const char \* name )**

Definition at line 906 of file reserved.c.

```
{
    struct reserved *p;
    struct noreserve      *norespstr;

    p = bsearch (name, reserved_words, NUM_RESERVED_WORDS,
                sizeof (struct reserved), reserve_comp);
    if (!p) {
        return 0;
    }
    for (norespstr = noresstab; norespstr; norespstr = norespstr->next) {
        if (strcasemp (name, norespstr->noresword) == 0) {
            return 0;
        }
    }
    if (p->token != -1) {
        return p->token;
    }
    cb_error (_("%s' reserved word, but not supported yet"), name);
    return 0;
}
```

**7.16.4.206 cb\_tree lookup\_system\_name ( const char \* name )**

Definition at line 893 of file reserved.c.

```
{
    int    i;

    for (i = 0; system_table[i].name != NULL; ++i) {
        if (strcasemp (name, system_table[i].name) == 0) {
            return system_table[i].node;
        }
    }
    return cb_error_node;
}
```

**7.16.4.207 void redefinition\_error ( cb\_tree x )**

Definition at line 154 of file error.c.

```
{
    struct cb_word *w;

    w = CB_REFERENCE (x)->word;
    cb_error_x (x, _("Redefinition of '%s'"), w->name);
    cb_error_x (CB_VALUE (w->items), _("%s' previously defined here"), w->
name);
}
```

**7.16.4.208 void redefinition\_warning ( cb\_tree x, cb\_tree y )**

Definition at line 164 of file error.c.

```

{
    struct cb_word *w;

    w = CB_REFERENCE (x)->word;
    cb_warning_x (x, _("Redefinition of '%s'"), w->name);
    if (y) {
        cb_warning_x (y, _("%s' previously defined here"), w->name);
    } else {
        cb_warning_x (CB_VALUE (w->items), _("%s' previously defined here"), w->name);
    }
}

```

**7.16.4.209 void undefined\_error ( cb\_tree x )**

Definition at line 178 of file error.c.

```

{
    struct cb_reference *r;
    cb_tree c;

    if (!errnamebuff) {
        errnamebuff = cobc_malloc (COB_NORMAL_BUFFER);
    }
    r = CB_REFERENCE (x);
    snprintf (errnamebuff, COB_NORMAL_MAX, "%s'", CB_NAME (x));
    for (c = r->chain; c; c = CB_REFERENCE (c)->chain) {
        strcat (errnamebuff, " in '");
        strcat (errnamebuff, CB_NAME (c));
        strcat (errnamebuff, "'");
    }
    cb_error_x (x, _("%s undefined"), errnamebuff);
}

```

**7.16.4.210 void validate\_file ( struct cb\_file \*f, cb\_tree name )**

Definition at line 1593 of file tree.c.

```

{
    /* check RECORD/RELATIVE KEY clause */
    switch (f->organization) {
    case COB_ORG_INDEXED:
        if (f->key == NULL) {
            file_error (name, "RECORD KEY");
        }
        break;
    case COB_ORG_RELATIVE:
        if (f->key == NULL && f->access_mode != COB_ACCESS_SEQUENTIAL) {
            file_error (name, "RELATIVE KEY");
        }
    }
}

```



```

        break;
    }
}

```

#### 7.16.4.211 int validate\_move ( cb\_tree src, cb\_tree dst, size\_t is\_value )

Definition at line 3944 of file typeck.c.

```

{
    struct cb_field      *f;
    struct cb_literal    *l;
    unsigned char        *p;
    cb_tree              loc;
    long long            val;
    size_t                i;
    size_t                is_numeric_edited = 0;
    int                   src_scale_mod;
    int                   dst_scale_mod;
    int                   dst_size_mod;
    int                   size;
    int                   most_significant;
    int                   least_significant;

    loc = src->source_line ? src : dst;
    if (CB_REFERENCE_P(dst) && CB_ALPHABET_NAME_P(CB_REFERENCE(dst)->value))
    {
        goto invalid;
    }
    if (CB_REFERENCE_P(dst) && CB_FILE_P(CB_REFERENCE(dst)->value)) {
        goto invalid;
    }
    if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_BOOLEAN) {
        cb_error_x (loc, _("Invalid destination for MOVE"));
        return -1;
    }

    if (CB_TREE_CLASS (dst) == CB_CLASS_POINTER) {
        if (CB_TREE_CLASS (src) == CB_CLASS_POINTER) {
            return 0;
        } else {
            goto invalid;
        }
    }

    f = cb_field (dst);
    switch (CB_TREE_TAG (src)) {
    case CB_TAG_CONST:
        if (src == cb_space) {
            if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_NUMERIC
                || (CB_TREE_CATEGORY (dst) ==
CB_CATEGORY_NUMERIC_EDITED && !is_value)) {
                goto invalid;
            }
        } else if (src == cb_zero) {
            if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_ALPHABETIC) {
                goto invalid;
            }
        }
        break;

```

```

case CB_TAG_LITERAL:
    /* TODO: ALL literal */

    l = CB_LITERAL (src);
    if (CB_TREE_CLASS (src) == CB_CLASS_NUMERIC) {
        /* Numeric literal */
        if (l->all) {
            goto invalid;
        }
        most_significant = -999;
        least_significant = 999;

        /* compute the most significant figure place */
        for (i = 0; i < l->size; i++) {
            if (l->data[i] != '0') {
                break;
            }
        }
        if (i != l->size) {
            most_significant = (int) (l->size - l->scale - i
- 1);
        }

        /* compute the least significant figure place */
        for (i = 0; i < l->size; i++) {
            if (l->data[l->size - i - 1] != '0') {
                break;
            }
        }
        if (i != l->size) {
            least_significant = (int) (-l->scale + i);
        }

        /* value check */
        switch (CB_TREE_CATEGORY (dst)) {
        case CB_CATEGORY_ALPHANUMERIC:
        case CB_CATEGORY_ALPHANUMERIC_EDITED:
            if (is_value) {
                goto expect_alphanumeric;
            }

            if (l->scale == 0) {
                goto expect_alphanumeric;
            } else {
                goto invalid;
            }
        case CB_CATEGORY_NUMERIC:
            if (f->pic->scale < 0) {
                /* check for PIC 9(n)P(m) */
                if (least_significant < -f->pic->scale) {
                    goto value_mismatch;
                }
            } else if (f->pic->scale > f->pic->size) {
                /* check for PIC P(n)9(m) */
                if (most_significant >= f->pic->size - f-
>pic->scale) {
                    goto value_mismatch;
                }
            }
            break;
        case CB_CATEGORY_NUMERIC_EDITED:

```

```

        if (is_value) {
            goto expect_alphanumeric;
        }

        /* TODO */
        break;
default:
    if (is_value) {
        goto expect_alphanumeric;
    }
    goto invalid;
}

/* sign check */
if (l->sign != 0 && !f->pic->have_sign) {
    if (is_value) {
        cb_error_x (loc, _("Data item not signed"
));
        return -1;
    }
    if (cb_warn_constant) {
        cb_warning_x (loc, _("Ignoring negative s
ign"));
    }
}

/* size check */
if (f->flag_real_binary ||
    ((f->usage == CB_USAGE_COMP_5 ||
    f->usage == CB_USAGE_COMP_X ||
    f->usage == CB_USAGE_BINARY) &&
    f->pic->scale == 0)) {
    p = l->data;
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            p = &l->data[i];
            break;
        }
    }
    i = l->size - i;
    switch (f->size) {
    case 1:
        if (i > 18) {
            goto numlit_overflow;
        }
        val = cb_get_long_long (src);
        if (f->pic->have_sign) {
            if (val < -128LL ||
                val > 127LL) {
                goto numlit_overflow;
            }
        }
        else {
            if (val > 255LL) {
                goto numlit_overflow;
            }
        }
        break;
    case 2:
        if (i > 18) {
            goto numlit_overflow;
        }
        val = cb_get_long_long (src);

```

```
        if (f->pic->have_sign) {
            if (val < -32768LL ||
                val > 32767LL) {
                goto numlit_overflow;
            }
        } else {
            if (val > 65535LL) {
                goto numlit_overflow;
            }
        }
        break;
case 3:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -8388608LL ||
            val > 8388607LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 16777215LL) {
            goto numlit_overflow;
        }
    }
    break;
case 4:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -2147483648LL ||
            val > 2147483647LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 4294967295LL) {
            goto numlit_overflow;
        }
    }
    break;
case 5:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -549755813888LL ||
            val > 549755813887LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 1099511627775LL) {
            goto numlit_overflow;
        }
    }
    break;
case 6:
    if (i > 18) {
        goto numlit_overflow;
    }
```

```

    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -140737488355328LL ||
            val > 140737488355327LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 281474976710655LL) {
            goto numlit_overflow;
        }
    }
    break;
case 7:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -36028797018963968LL ||
            val > 36028797018963967LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 72057594037927935LL) {
            goto numlit_overflow;
        }
    }
    break;
default:
    if (f->pic->have_sign) {
        if (i < 19) {
            break;
        }
        if (i > 19) {
            goto numlit_overflow;
        }
        if (memcmp (p, "92233720368547758
07", 19) > 0) {
            goto numlit_overflow;
        }
    } else {
        if (i < 20) {
            break;
        }
        if (i > 20) {
            goto numlit_overflow;
        }
        if (memcmp (p, "18446744073709551
615", 20) > 0) {
            goto numlit_overflow;
        }
    }
    break;
}
return 0;
}
if (least_significant < -f->pic->scale) {
    goto size_overflow;
}
if (f->pic->scale > 0) {

```

```

        size = f->pic->digits - f->pic->scale;
    } else {
        size = f->pic->digits;
    }
    if (most_significant >= size) {
        goto size_overflow;
    }
} else {
    /* Alphanumeric literal */

    /* value check */
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_ALPHABETIC:
        for (i = 0; i < l->size; i++) {
            if (!isalpha (l->data[i]) && !isspace (l-
>data[i])) {
                goto value_mismatch;
            }
        }
        break;
    case CB_CATEGORY_NUMERIC:
        goto expect_numeric;
    case CB_CATEGORY_NUMERIC_EDITED:
        if (!is_value) {
            goto expect_numeric;
        }

        /* TODO: validate the value */
        break;
    default:
        break;
    }

    /* size check */
    size = cb_field_size (dst);
    if (size >= 0 && (int)l->size > size) {
        goto size_overflow;
    }
}
break;
case CB_TAG_FIELD:
case CB_TAG_REFERENCE:
    if (CB_REFERENCE_P(src) &&
        CB_ALPHABET_NAME_P(CB_REFERENCE(src)->value)) {
        break;
    }
    if (CB_REFERENCE_P(src) &&
        CB_FILE_P(CB_REFERENCE(src)->value)) {
        goto invalid;
    }
    size = cb_field_size (src);
    if (size < 0) {
        size = cb_field (src)->size;
    }
    /* non-elementary move */
    if (cb_field (src)->children || cb_field (dst)->children) {
        if (size > cb_field (dst)->size) {
            goto size_overflow_1;
        }
    }
    break;
}

```

```

/* elementary move */
switch (CB_TREE_CATEGORY (src)) {
case CB_CATEGORY_ALPHANUMERIC:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_NUMERIC:
    case CB_CATEGORY_NUMERIC_EDITED:
        if (size > cb_field (dst)->pic->digits) {
            goto size_overflow_2;
        }
        break;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        if (size >
            count_pic_alphanumeric_edited (cb_field (dst)
)) {
            goto size_overflow_1;
        }
        break;
    default:
        if (size > cb_field (dst)->size) {
            goto size_overflow_1;
        }
        break;
    }
    break;
case CB_CATEGORY_ALPHABETIC:
case CB_CATEGORY_ALPHANUMERIC_EDITED:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_NUMERIC:
    case CB_CATEGORY_NUMERIC_EDITED:
        goto invalid;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        if (size >
            count_pic_alphanumeric_edited(cb_field (dst)
) {
            goto size_overflow_1;
        }
        break;
    default:
        if (size > cb_field (dst)->size) {
            goto size_overflow_1;
        }
        break;
    }
    break;
case CB_CATEGORY_NUMERIC:
case CB_CATEGORY_NUMERIC_EDITED:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_ALPHABETIC:
        goto invalid;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        is_numeric_edited = 1;
        /* Drop through */
    case CB_CATEGORY_ALPHANUMERIC:
        if (is_numeric_edited) {
            dst_size_mod = count_pic_alphanumeric_edi
ted (cb_field (dst));
        } else {
            dst_size_mod = cb_field (dst)->size;
        }
        if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC
            && cb_field (src)->pic->scale > 0) {

```

```

        if (cb_move_noninteger_to_alphanumeric ==
CB_ERROR) {
            goto invalid;
        }
        cb_warning_x (loc, _("Move non-integer to
alphanumeric"));
        break;
    }
    if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC
) {
        goto size_overflow_2;
    }
    if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC_EDITED
        && cb_field (src)->pic->digits > dst_size_mod
    ) {
        goto size_overflow_1;
    }
    break;
default:
    src_scale_mod = cb_field (src)->pic->scale < 0 ?
        0 : cb_field (src)->pic->scale;
    dst_scale_mod = cb_field (dst)->pic->scale < 0 ?
        0 : cb_field (dst)->pic->scale;
    if (cb_field (src)->pic->digits - src_scale_mod >
        cb_field (dst)->pic->digits - dst_scale_mod
||
        src_scale_mod > dst_scale_mod) {
        goto size_overflow_2;
    }
    break;
}
break;
default:
    cb_error_x (loc, _("Invalid source for MOVE"));
    return -1;
}
break;
case CB_TAG_INTEGER:
case CB_TAG_BINARY_OP:
case CB_TAG_INTRINSIC:
    /* TODO: check this */
    break;
default:
    fprintf (stderr, "Invalid tree tag %d\n", CB_TREE_TAG (src));
    ABORT ();
}
return 0;

invalid:
if (is_value) {
    cb_error_x (loc, _("Invalid VALUE clause"));
} else {
    cb_error_x (loc, _("Invalid MOVE statement"));
}
return -1;

numlit_overflow:
if (is_value) {
    cb_error_x (loc, _("Invalid VALUE clause - literal exceeds data s

```



```
ize"));
    return -1;
}
if (cb_warn_constant) {
    cb_warning_x (loc, _("Numeric literal exceeds data size"));
}
return 0;

expect_numeric:
    return move_error (src, dst, is_value, cb_warn_strict_typing, 0,
        _("Numeric value is expected"));

expect_alphanumeric:
    return move_error (src, dst, is_value, cb_warn_strict_typing, 0,
        _("Alphanumeric value is expected"));

value_mismatch:
    return move_error (src, dst, is_value, cb_warn_constant, 0,
        _("Value does not fit the picture string"));

size_overflow:
    return move_error (src, dst, is_value, cb_warn_constant, 0,
        _("Value size exceeds data size"));

size_overflow_1:
    return move_error (src, dst, is_value, cb_warn_truncate, 1,
        _("Sending field larger than receiving field"));

size_overflow_2:
    return move_error (src, dst, is_value, cb_warn_truncate, 1,
        _("Some digits may be truncated"));
}
```

## 7.16.5 Variable Documentation

### 7.16.5.1 `cb_tree cb_any`

Definition at line 74 of file tree.c.

### 7.16.5.2 `cb_tree cb_error_node`

Definition at line 93 of file tree.c.

### 7.16.5.3 `cb_tree cb_false`

Definition at line 76 of file tree.c.

### 7.16.5.4 `cb_tree cb_high`

Definition at line 82 of file tree.c.

**7.16.5.5   cb\_tree cb\_i[8]**

Definition at line 92 of file tree.c.

**7.16.5.6   cb\_tree cb\_int0**

Definition at line 86 of file tree.c.

**7.16.5.7   cb\_tree cb\_int1**

Definition at line 87 of file tree.c.

**7.16.5.8   cb\_tree cb\_int2**

Definition at line 88 of file tree.c.

**7.16.5.9   cb\_tree cb\_int3**

Definition at line 89 of file tree.c.

**7.16.5.10   cb\_tree cb\_int4**

Definition at line 90 of file tree.c.

**7.16.5.11   cb\_tree cb\_int5**

Definition at line 91 of file tree.c.

**7.16.5.12   cb\_tree cb\_intr\_e**

Definition at line 97 of file tree.c.

**7.16.5.13   cb\_tree cb\_intr\_pi**

Definition at line 96 of file tree.c.

**7.16.5.14   cb\_tree cb\_intr\_whencomp**

Definition at line 95 of file tree.c.

**7.16.5.15   cb\_tree cb\_low**

Definition at line 81 of file tree.c.

**7.16.5.16   size\_t cb\_needs\_01**

Definition at line 33 of file field.c.

**7.16.5.17   cb\_tree cb\_norm\_high**

Definition at line 84 of file tree.c.

**7.16.5.18   cb\_tree cb\_norm\_low**

Definition at line 83 of file tree.c.

**7.16.5.19   cb\_tree cb\_null**

Definition at line 77 of file tree.c.

**7.16.5.20   cb\_tree cb\_one**

Definition at line 79 of file tree.c.

**7.16.5.21   cb\_tree cb\_quote**

Definition at line 85 of file tree.c.

**7.16.5.22   cb\_tree cb\_space**

Definition at line 80 of file tree.c.

**7.16.5.23   cb\_tree cb\_standard\_error\_handler**

Definition at line 99 of file tree.c.

**7.16.5.24   cb\_tree cb\_true**

Definition at line 75 of file tree.c.

### 7.16.5.25 `cb_tree` `cb_zero`

Definition at line 78 of file `tree.c`.

### 7.16.5.26 `size_t` `gen_screen_ptr`

Definition at line 101 of file `tree.c`.

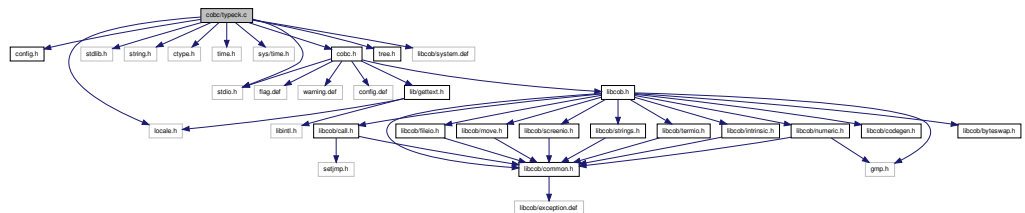
### 7.16.5.27 `int` `non_const_word`

Definition at line 988 of file `parser.c`.

## 7.17 `cobc/typeeck.c` File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <sys/time.h>
#include <locale.h>
#include "cobc.h"
#include "tree.h"
#include "libcob/system.def"
```

Include dependency graph for `typeeck.c`:



## Classes

- struct `system_table`

- struct [expr\\_node](#)

## Defines

- #define [START\\_STACK\\_SIZE](#) 32
- #define [TOKEN](#)(offset) (expr\_stack[expr\_index + offset].token)
- #define [VALUE](#)(offset) (expr\_stack[expr\_index + offset].value)
- #define [dpush](#)(x) decimal\_stack = cb\_cons (x, decimal\_stack)
- #define [cb\\_emit](#)(x) [current\\_statement](#)->body = cb\_list\_add ([current\\_statement](#)->body, x)
- #define [cb\\_emit\\_list](#)(l) [current\\_statement](#)->body = cb\_list\_append ([current\\_statement](#)->body, l)
- #define [COB\\_SYSTEM\\_GEN](#)(x, y, z) { x, y },

## Functions

- [cb\\_tree](#) [cb\\_check\\_numeric\\_value](#) ([cb\\_tree](#) x)
- void [cb\\_build\\_registers](#) (void)
- char \* [cb\\_encode\\_program\\_id](#) (const char \*name)
- const char \* [cb\\_build\\_program\\_id](#) ([cb\\_tree](#) name, [cb\\_tree](#) alt\_name)
- void [cb\\_define\\_switch\\_name](#) ([cb\\_tree](#) name, [cb\\_tree](#) sname, [cb\\_tree](#) flag, [cb\\_tree](#) ref)
- [cb\\_tree](#) [cb\\_build\\_section\\_name](#) ([cb\\_tree](#) name, int sect\_or\_para)
- [cb\\_tree](#) [cb\\_build\\_assignment\\_name](#) (struct [cb\\_file](#) \*cfile, [cb\\_tree](#) name)
- [cb\\_tree](#) [cb\\_build\\_index](#) ([cb\\_tree](#) x, [cb\\_tree](#) values, int indexed\_by, struct [cb\\_field](#) \*qual)
- [cb\\_tree](#) [cb\\_build\\_identifier](#) ([cb\\_tree](#) x)
- [cb\\_tree](#) [cb\\_build\\_const\\_length](#) ([cb\\_tree](#) x)
- [cb\\_tree](#) [cb\\_build\\_length](#) ([cb\\_tree](#) x)
- [cb\\_tree](#) [cb\\_build\\_address](#) ([cb\\_tree](#) x)
- [cb\\_tree](#) [cb\\_build\\_ppointer](#) ([cb\\_tree](#) x)
- void [cb\\_validate\\_program\\_environment](#) (struct [cb\\_program](#) \*prog)
- void [cb\\_validate\\_program\\_data](#) (struct [cb\\_program](#) \*prog)
- void [cb\\_validate\\_program\\_body](#) (struct [cb\\_program](#) \*prog)
- [cb\\_tree](#) [cb\\_build\\_expr](#) ([cb\\_tree](#) list)
- void [cb\\_emit\\_arithmetic](#) ([cb\\_tree](#) vars, int op, [cb\\_tree](#) val)
- [cb\\_tree](#) [cb\\_build\\_cond](#) ([cb\\_tree](#) x)
- [cb\\_tree](#) [cb\\_build\\_add](#) ([cb\\_tree](#) v, [cb\\_tree](#) n, [cb\\_tree](#) round\_opt)
- [cb\\_tree](#) [cb\\_build\\_sub](#) ([cb\\_tree](#) v, [cb\\_tree](#) n, [cb\\_tree](#) round\_opt)
- void [cb\\_emit\\_corresponding](#) ([cb\\_tree](#)(\*func)([cb\\_tree](#) f1, [cb\\_tree](#) f2, [cb\\_tree](#) f3), [cb\\_tree](#) x1, [cb\\_tree](#) x2, [cb\\_tree](#) opt)
- void [cb\\_emit\\_move\\_corresponding](#) ([cb\\_tree](#) x1, [cb\\_tree](#) x2)
- void [cb\\_emit\\_accept](#) ([cb\\_tree](#) var, [cb\\_tree](#) pos, [cb\\_tree](#) fgc, [cb\\_tree](#) bgc, [cb\\_tree](#) scroll, int dispattrs)
- void [cb\\_emit\\_accept\\_line\\_or\\_col](#) ([cb\\_tree](#) var, const int l\_or\_c)
- void [cb\\_emit\\_accept\\_date](#) ([cb\\_tree](#) var)

- void `cb_emit_accept_date_yyyymmdd` (`cb_tree` var)
- void `cb_emit_accept_day` (`cb_tree` var)
- void `cb_emit_accept_day_yyyddd` (`cb_tree` var)
- void `cb_emit_accept_day_of_week` (`cb_tree` var)
- void `cb_emit_accept_time` (`cb_tree` var)
- void `cb_emit_accept_command_line` (`cb_tree` var)
- void `cb_emit_get_environment` (`cb_tree` envvar, `cb_tree` envval)
- void `cb_emit_accept_environment` (`cb_tree` var)
- void `cb_emit_accept_arg_number` (`cb_tree` var)
- void `cb_emit_accept_arg_value` (`cb_tree` var)
- void `cb_emit_accept_mnemonic` (`cb_tree` var, `cb_tree` mnemonic)
- void `cb_emit_accept_name` (`cb_tree` var, `cb_tree` name)
- void `cb_emit_allocate` (`cb_tree` target1, `cb_tree` target2, `cb_tree` size, `cb_tree` initialize)
- void `cb_emit_call` (`cb_tree` prog, `cb_tree` using, `cb_tree` returning, `cb_tree` on\_exception, `cb_tree` not\_on\_exception)
- void `cb_emit_cancel` (`cb_tree` prog)
- void `cb_emit_close` (`cb_tree` file, `cb_tree` opt)
- void `cb_emit_commit` (void)
- void `cb_emit_continue` (void)
- void `cb_emit_delete` (`cb_tree` file)
- void `cb_emit_env_name` (`cb_tree` value)
- void `cb_emit_env_value` (`cb_tree` value)
- void `cb_emit_arg_number` (`cb_tree` value)
- void `cb_emit_command_line` (`cb_tree` value)
- void `cb_emit_display` (`cb_tree` values, `cb_tree` upon, `cb_tree` no\_adv, `cb_tree` pos, `cb_tree` fg, `cb_tree` bg, `cb_tree` scroll, int dispattrs)
- `cb_tree` `cb_build_display_upon` (`cb_tree` x)
- `cb_tree` `cb_build_display_upon_direct` (`cb_tree` x)
- void `cb_emit_divide` (`cb_tree` dividend, `cb_tree` divisor, `cb_tree` quotient, `cb_tree` remainder)
- void `cb_emit_evaluate` (`cb_tree` subject\_list, `cb_tree` case\_list)
- void `cb_emit_free` (`cb_tree` vars)
- void `cb_emit_goto` (`cb_tree` target, `cb_tree` depending)
- void `cb_emit_exit` (size\_t goback)
- void `cb_emit_if` (`cb_tree` cond, `cb_tree` stmt1, `cb_tree` stmt2)
- void `cb_emit_initialize` (`cb_tree` vars, `cb_tree` fillinit, `cb_tree` value, `cb_tree` replacing, `cb_tree` def)
- void `cb_emit_inspect` (`cb_tree` var, `cb_tree` body, `cb_tree` replacing, int replconv)
- void `cb_init_tarrying` (void)
- `cb_tree` `cb_build_tarrying_data` (`cb_tree` x)
- `cb_tree` `cb_build_tarrying_characters` (`cb_tree` l)
- `cb_tree` `cb_build_tarrying_all` (void)
- `cb_tree` `cb_build_tarrying_leading` (void)
- `cb_tree` `cb_build_tarrying_trailing` (void)
- `cb_tree` `cb_build_tarrying_value` (`cb_tree` x, `cb_tree` l)
- `cb_tree` `cb_build_replacing_characters` (`cb_tree` x, `cb_tree` l)

- `cb_tree cb_build_replacing_all (cb_tree x, cb_tree y, cb_tree l)`
- `cb_tree cb_build_replacing_leading (cb_tree x, cb_tree y, cb_tree l)`
- `cb_tree cb_build_replacing_first (cb_tree x, cb_tree y, cb_tree l)`
- `cb_tree cb_build_replacing_trailing (cb_tree x, cb_tree y, cb_tree l)`
- `cb_tree cb_build_converting (cb_tree x, cb_tree y, cb_tree l)`
- `cb_tree cb_build_inspect_region_start (void)`
- `cb_tree cb_build_inspect_region (cb_tree l, cb_tree pos, cb_tree x)`
- `int validate_move (cb_tree src, cb_tree dst, size_t is_value)`
- `cb_tree cb_build_move (cb_tree src, cb_tree dst)`
- `void cb_emit_move (cb_tree src, cb_tree dst)`
- `void cb_emit_open (cb_tree file, cb_tree mode, cb_tree sharing)`
- `void cb_emit_perform (cb_tree perform, cb_tree body)`
- `cb_tree cb_build_perform_once (cb_tree body)`
- `cb_tree cb_build_perform_times (cb_tree times)`
- `cb_tree cb_build_perform_until (cb_tree condition, cb_tree varying)`
- `cb_tree cb_build_perform_forever (cb_tree body)`
- `cb_tree cb_build_perform_exit (struct cb_label *label)`
- `void cb_emit_read (cb_tree ref, cb_tree next, cb_tree into, cb_tree key, cb_tree lock_opts)`
- `void cb_emit_rewrite (cb_tree record, cb_tree from, cb_tree lockopt)`
- `void cb_emit_release (cb_tree record, cb_tree from)`
- `void cb_emit_return (cb_tree ref, cb_tree into)`
- `void cb_emit_rollback (void)`
- `void cb_emit_search (cb_tree table, cb_tree varying, cb_tree at_end, cb_tree whens)`
- `void cb_emit_search_all (cb_tree table, cb_tree at_end, cb_tree when, cb_tree stmts)`
- `void cb_emit_setenv (cb_tree x, cb_tree y)`
- `void cb_emit_set_to (cb_tree vars, cb_tree x)`
- `void cb_emit_set_up_down (cb_tree l, cb_tree flag, cb_tree x)`
- `void cb_emit_set_on_off (cb_tree l, cb_tree flag)`
- `void cb_emit_set_true (cb_tree l)`
- `void cb_emit_set_false (cb_tree l)`
- `void cb_emit_sort_init (cb_tree name, cb_tree keys, cb_tree col)`
- `void cb_emit_sort_using (cb_tree file, cb_tree l)`
- `void cb_emit_sort_input (cb_tree proc)`
- `void cb_emit_sort_giving (cb_tree file, cb_tree l)`
- `void cb_emit_sort_output (cb_tree proc)`
- `void cb_emit_sort_finish (cb_tree file)`
- `void cb_emit_start (cb_tree file, cb_tree op, cb_tree key)`
- `void cb_emit_stop_run (cb_tree x)`
- `void cb_emit_string (cb_tree items, cb_tree into, cb_tree pointer)`
- `void cb_emit_unlock (cb_tree ref)`
- `void cb_emit_unstring (cb_tree name, cb_tree delimited, cb_tree into, cb_tree pointer, cb_tree tallying)`
- `cb_tree cb_build_unstring_delimited (cb_tree all, cb_tree value)`
- `cb_tree cb_build_unstring_into (cb_tree name, cb_tree delimiter, cb_tree count)`

- void `cb_emit_write` (`cb_tree` record, `cb_tree` from, `cb_tree` opt, `cb_tree` lockopt)
- `cb_tree` `cb_build_write_advancing_lines` (`cb_tree` pos, `cb_tree` lines)
- `cb_tree` `cb_build_write_advancing_mnemonic` (`cb_tree` pos, `cb_tree` mnemonic)
- `cb_tree` `cb_build_write_advancing_page` (`cb_tree` pos)

## Variables

- size\_t `sending_id` = 0
- size\_t `suppress_warn` = 0

### 7.17.1 Define Documentation

7.17.1.1 `#define cb_emit( x ) current_statement->body = cb_list_add  
(current_statement->body, x)`

Definition at line 67 of file typeck.c.

7.17.1.2 `#define cb_emit_list( l ) current_statement->body = cb_list_append  
(current_statement->body, l)`

Definition at line 69 of file typeck.c.

7.17.1.3 `#define COB_SYSTEM_GEN( x, y, z ) { x, y },`

7.17.1.4 `#define dpush( x ) decimal_stack = cb_cons (x, decimal_stack)`

Definition at line 65 of file typeck.c.

7.17.1.5 `#define START_STACK_SIZE 32`

Definition at line 61 of file typeck.c.

7.17.1.6 `#define TOKEN( offset ) (expr_stack[expr_index + offset].token)`

Definition at line 62 of file typeck.c.

7.17.1.7 `#define VALUE( offset ) (expr_stack[expr_index + offset].value)`

Definition at line 63 of file typeck.c.



## 7.17.2 Function Documentation

### 7.17.2.1 `cb_tree cb_build_add ( cb_tree v, cb_tree n, cb_tree round_opt )`

Definition at line 2591 of file typeck.c.

```

{
    cb_tree      opt;
    struct cb_field *f;

#ifdef COB_NON_ALIGNED
    if (CB_INDEX_P (v)) {
        return cb_build_move (cb_build_binary_op (v, '+', n), v);
    }
    if (CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        current_program->gen_ptrmanip = 1;
        return cb_build_funcall_3 ("cob_pointer_manip", v, n, cb_int0);
    }
#else
    if (CB_INDEX_P (v) || CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        return cb_build_move (cb_build_binary_op (v, '+', n), v);
    }
#endif

    if (CB_REF_OR_FIELD_P (v)) {
        f = cb_field (v);
        f->count++;
    }
    if (CB_REF_OR_FIELD_P (n)) {
        f = cb_field (n);
        f->count++;
    }
    if (round_opt == cb_high) {
        if (cb_fits_int (n)) {
            return cb_build_optim_add (v, n);
        } else {
            return cb_build_funcall_3 ("cob_add", v, n, cb_int0);
        }
    }
    opt = build_store_option (v, round_opt);
    if (opt == cb_int0 && cb_fits_int (n)) {
        return cb_build_optim_add (v, n);
    }
    return cb_build_funcall_3 ("cob_add", v, n, opt);
}

```

### 7.17.2.2 `cb_tree cb_build_address ( cb_tree x )`

Definition at line 1027 of file typeck.c.

```

{
    if (x == cb_error_node ||
        (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node)) {
        return cb_error_node;
    }

    return cb_build_cast_address (x);
}

```

7.17.2.3 `cb_tree cb_build_assignment_name ( struct cb_file * cfile, cb_tree name )`

Definition at line 677 of file typeck.c.

```

{
    const char    *s;
    const char    *p;

    if (name == cb_error_node) {
        return cb_error_node;
    }

    switch (CB_TREE_TAG (name)) {
    case CB_TAG_LITERAL:
        if (strcmp ((char *) (CB_LITERAL(name)->data), "$#@DUMMY@#$") == 0
) {
            cfile->special = 2;
        }
        return name;

    case CB_TAG_REFERENCE:
        s = CB_REFERENCE (name)->word->name;
        if (strcasecmp (s, "KEYBOARD") == 0) {
            s = "#DUMMY#";
            cfile->special = 1;
            return cb_build_alphanumeric_literal ((ucharptr)s, strlen
(s));
        }
        switch (cb_assign_clause) {
        case CB_ASSIGN_COBOL2002:
            /* TODO */
            return cb_error_node;

        case CB_ASSIGN_MF:
            if (cfile->external_assign) {
                p = strchr (s, '-');
                if (p) {
                    s = p + 1;
                }
                return cb_build_alphanumeric_literal ((ucharptr)s
, strlen (s));
            }
            current_program->reference_list =
                cb_list_add (current_program->reference_list, name);
            return name;

        case CB_ASSIGN_IBM:
            /* check organization */
            if (strncmp (s, "S-", 2) == 0 ||
                strncmp (s, "AS-", 3) == 0) {
                goto org;
            }
            /* skip the device label if exists */
            if ((p = strchr (s, '-')) != NULL) {
                s = p + 1;
            }
            /* check organization again */
            if (strncmp (s, "S-", 2) == 0 ||
                strncmp (s, "AS-", 3) == 0) {
org:
                /* skip it for now */
                s = strchr (s, '-') + 1;

```

```

    }
    /* convert the name into literal */
    return cb_build_alphanumeric_literal ((ucharptr)s, strlen
(s));
}

default:
    return cb_error_node;
}
}

```

#### 7.17.2.4 `cb_tree cb_build_cond ( cb_tree x )`

Definition at line 2354 of file typeck.c.

```

{
    int                size1;
    int                size2;
    struct cb_field    *f;
    struct cb_binary_op *p;
    cb_tree            d1;
    cb_tree            d2;

    switch (CB_TREE_TAG (x)) {
    case CB_TAG_CONST:
    case CB_TAG_FUNCALL:
        return x;
    case CB_TAG_REFERENCE:
        if (!CB_FIELD_P (cb_ref (x))) {
            return cb_build_cond (cb_ref (x));
        }

        f = cb_field (x);

        /* level 88 condition */
        if (f->level == 88) {
            /* We need to build a 88 condition at every occurrence
            instead of once at the beginning because a 88 item
            may be subscripted (i.e., it is not a constant tree).
            */
            return cb_build_cond (build_cond_88 (x));
        }

        cb_error_x (x, _("Invalid expression"));
        return cb_error_node;
    case CB_TAG_BINARY_OP:
        p = CB_BINARY_OP (x);
        switch (p->op) {
        case '!':
            return cb_build_negation (cb_build_cond (p->x));
        case '&':
        case '|':
            return cb_build_binary_op (cb_build_cond (p->x), p->op,
cb_build_cond (p->y));
        default:
            if (CB_INDEX_P (p->x) || CB_INDEX_P (p->y)
                || CB_TREE_CLASS (p->x) == CB_CLASS_POINTER
                || CB_TREE_CLASS (p->y) == CB_CLASS_POINTER) {
                x = cb_build_binary_op (p->x, '-', p->y);
            }
        }
    }
}

```

```

    } else if (CB_BINARY_OP_P (p->x) || CB_BINARY_OP_P (p->y)
) {
    /* decimal comparison */
    d1 = decimal_alloc ();
    d2 = decimal_alloc ();

    decimal_expand (d1, p->x);
    decimal_expand (d2, p->y);
    dpush (cb_build_funcall_2 ("cob_decimal_cmp", d1,
d2));

    decimal_free ();
    decimal_free ();
    x = cb_list_reverse (decimal_stack);
    decimal_stack = NULL;
} else {
    if (cb_chk_num_cond (p->x, p->y)) {
        size1 = cb_field_size (p->x);
        x = cb_build_funcall_3 ("memcmp",
            cb_build_cast_address (p->x),
            cb_build_cast_address (p->y),
            cb_int (size1));
        break;
    }
    if (CB_TREE_CLASS (p->x) == CB_CLASS_NUMERIC
        && CB_TREE_CLASS (p->y) == CB_CLASS_NUMERIC
        && cb_fits_int (p->y)) {
        x = cb_build_optim_cond (p);
        break;
    }

    /* field comparison */
    if ((CB_REF_OR_FIELD_P (p->x))
        && (CB_TREE_CATEGORY (p->x) ==
CB_CATEGORY_ALPHANUMERIC ||
CB_CATEGORY_ALPHABETIC)
        && (CB_TREE_CATEGORY (p->x) ==
CB_CATEGORY_ALPHANUMERIC ||
CB_CATEGORY_ALPHABETIC)
        && (cb_field_size (p->x) == 1)
        && (!current_program->alphabet_name_list)
        && (p->y == cb_space || p->y == cb_low ||
p->y == cb_high || p->y == cb_zero)) {
        x = cb_build_funcall_2 ("$G", p->x, p->y)
;
        break;
    }
    if (cb_chk_alpha_cond (p->x) && cb_chk_alpha_cond
(p->y)) {
        size1 = cb_field_size (p->x);
        size2 = cb_field_size (p->y);
    } else {
        size1 = 0;
        size2 = 0;
    }
    if (size1 == 1 && size2 == 1) {
        x = cb_build_funcall_2 ("$G", p->x, p->y)
;
    } else if (size1 != 0 && size1 == size2) {
        x = cb_build_funcall_3 ("memcmp",
            cb_build_cast_address (p->x),
            cb_build_cast_address (p->y),
            cb_int (size1));
    } else {
        if (CB_TREE_CLASS (p->x) ==

```

```

CB_CLASS_NUMERIC && p->y == cb_zero) {
    x = cb_build_optim_cond (p);
    } else {
    x = cb_build_funcall_2 ("cob_cmp"
, p->x, p->y);
    }
    }
    }
    }
    return cb_build_binary_op (x, p->op, p->y);
default:
cb_error_x (x, _("Invalid expression"));
return cb_error_node;
}
/* NOT REACHED */
return x;
}

```

#### 7.17.2.5 `cb_tree cb_build_const_length ( cb_tree x )`

Definition at line 956 of file typeck.c.

```

{
    struct cb_field      *f;
    char                 buff[64];

    if (x == cb_error_node) {
        return cb_error_node;
    }
    if (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node) {
        return cb_error_node;
    }

    memset (buff, 0, sizeof (buff));
    f = CB_FIELD (cb_ref (x));
    if (f->flag_any_length) {
        cb_error (_("ANY LENGTH item not allowed here"));
        return cb_error_node;
    }
    if (f->level == 88) {
        cb_error (_("88 level item not allowed here"));
        return cb_error_node;
    }
    if (!f->flag_is_verified) {
        cb_validate_field (f);
    }
    sprintf (buff, "%d", f->memory_size);
    return cb_build_numeric_literal (0, (ucharptr)buff, 0);
}

```

#### 7.17.2.6 `cb_tree cb_build_converting ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3847 of file typeck.c.

```

{

```

```

        return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_converting", x, y
    ));
}

```

### 7.17.2.7 `cb_tree cb_build.display_upon ( cb_tree x )`

Definition at line 3406 of file typeck.c.

```

{
    if (x == cb_error_node) {
        return cb_error_node;
    }

    switch (CB_SYSTEM_NAME (cb_ref (x))->token) {
    case CB_DEVICE_CONSOLE:
    case CB_DEVICE_SYSOUT:
        return cb_int0;
    case CB_DEVICE_SYSERR:
        return cb_int1;
    default:
        cb_error_x (x, _("Invalid output stream"));
        return cb_error_node;
    }
}

```

### 7.17.2.8 `cb_tree cb_build.display_upon_direct ( cb_tree x )`

Definition at line 3425 of file typeck.c.

```

{
    const char    *name;
    cb_tree      sys;

    if (x == cb_error_node) {
        return cb_error_node;
    }
    name = CB_NAME (x);
    if (CB_REFERENCE (x)->word->count == 0) {
        sys = lookup_system_name (CB_NAME (x));
        if (sys != cb_error_node) {
            switch (CB_SYSTEM_NAME (sys)->token) {
            case CB_DEVICE_CONSOLE:
            case CB_DEVICE_SYSOUT:
                cb_warning_x (x, _("%s' undefined in SPECIAL-NAM
ES"), name);
                return cb_int0;
            case CB_DEVICE_SYSERR:
                cb_warning_x (x, _("%s' undefined in SPECIAL-NAM
ES"), name);
                return cb_int1;
            default:
                break;
            }
        }
    }
}

```

```

    cb_error_x (x, _("%s' undefined in SPECIAL-NAMES"), name);
    return cb_error_node;
}

```

### 7.17.2.9 `cb_tree cb_build_expr ( cb_tree list )`

Definition at line 1820 of file typeck.c.

```

{
    cb_tree l;
/* RXW
    cb_tree x;
*/
    int    op;

    cb_expr_init ();

    for (l = list; l; l = CB_CHAIN (l)) {
        op = CB_PURPOSE_INT (l);
        switch (op) {
            case '9': /* NUMERIC */
                cb_expr_shift_class ("cob_is_numeric");
                break;
            case 'A': /* ALPHABETIC */
                cb_expr_shift_class ("cob_is_alpha");
                break;
            case 'L': /* ALPHABETIC_LOWER */
                cb_expr_shift_class ("cob_is_lower");
                break;
            case 'U': /* ALPHABETIC_UPPER */
                cb_expr_shift_class ("cob_is_upper");
                break;
            case 'P': /* POSITIVE */
                cb_expr_shift_sign ('>');
                break;
            case 'N': /* NEGATIVE */
                cb_expr_shift_sign ('<');
                break;
            case 'O': /* OMITTED */
                current_statement->null_check = NULL;
                cb_expr_shift_class ("cob_is_omitted");
                break;
/* RXW
            case 'x':
                if (CB_VALUE (l) && CB_REFERENCE_P (CB_VALUE (l))) {
                    x = CB_CHAIN (l);
                    if (x && cb_field (CB_VALUE (l))->level == 88) {
                        switch (CB_PURPOSE_INT (x)) {
                            case '&':
                            case '|':
                            case '(':
                            case ')':
                                break;
                            default:
                                cb_error (_("Invalid condition"));
                                break;
                        }
                    }
                }

```

```

        }
        cb_expr_shift (op, CB_VALUE (1));
        break;
*/
        default:
            cb_expr_shift (op, CB_VALUE (1));
            break;
    }
}

return cb_expr_finish ();
}

```

### 7.17.2.10 `cb_tree cb_build_identifier ( cb_tree x )`

Definition at line 763 of file typeck.c.

```

{
    struct cb_reference    *r;
    struct cb_field       *f;
    struct cb_field       *p;
    const char            *name;
    cb_tree                v;
    cb_tree                e1;
    cb_tree                e2;
    cb_tree                l;
    cb_tree                sub;
    int                    offset;
    int                    length;
    int                    n;

    if (x == cb_error_node) {
        return cb_error_node;
    }

    r = CB_REFERENCE (x);
    name = r->word->name;

    /* resolve reference */
    v = cb_ref (x);
    if (v == cb_error_node) {
        return cb_error_node;
    }

    /* check if it is a data name */
    if (!CB_FIELD_P (v)) {
        if (r->subs) {
            cb_error_x (x, _("%s' cannot be subscripted"), name);
            return cb_error_node;
        }
        if (r->offset) {
            cb_error_x (x, _("%s' cannot be reference modified"), name);
            return cb_error_node;
        }
        return x;
    }
    f = CB_FIELD (v);
}

```



```

/* BASED check */
if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_PTR)) {
    for (p = f; p->parent; p = p->parent) {
        ;
    }
    if (current_statement) {
        if (p->flag_item_based ||
            (f->storage == CB_STORAGE_LINKAGE &&
             !p->flag_is_pdiv_parm)) {
            current_statement->null_check =
cb_build_funcall_2 (
                    "cob_check_based",
                    cb_build_address (
cb_build_field_reference (p, NULL)),
                    cb_build_string0 ((ucharptr)name));
        }
    }
}

/* check the number of subscripts */
if (!r->all && cb_list_length (r->subs) != f->indexes) {
    switch (f->indexes) {
        case 0:
            cb_error_x (x, _("%s' cannot be subscripted"), name);
            return cb_error_node;
        case 1:
            cb_error_x (x, _("%s' requires 1 subscript"), name);
            return cb_error_node;
        default:
            cb_error_x (x, _("%s' requires %d subscripts"), name, f-
>indexes);
            return cb_error_node;
    }
}

/* subscript check */
if (!r->all && r->subs) {
    l = r->subs;
    for (p = f; p; p = p->parent) {
        if (p->flag_occurs) {
            sub = cb_check_integer_value (CB_VALUE (l));

            l = CB_CHAIN (l);

            if (sub == cb_error_node) {
                continue;
            }

            /* compile-time check */
            if (CB_LITERAL_P (sub)) {
                n = cb_get_int (sub);
                if (n < 1 || n > p->occurs_max) {
                    cb_error_x (x, _("Subscript of '%
s' out of bounds: %d"),
                                name, n);
                }
            }

            /* run-time check */
            if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_SUBSCRIPT))
{

```

```

        if (p->occursDepending) {
            e1 = cb_build_funcall_4 ("cob_che
ck_odo",
                                cb_build_cast_integer (p
->occursDepending),
                                cb_int (p->occurs_min),
                                cb_int (p->occurs_max),
                                cb_build_string0
                                ((ucharptr) (cb_field (p-
>occursDepending)->name)));
            e2 = cb_build_funcall_4 ("cob_che
ck_subscript",
                                cb_build_cast_integer (s
ub),
                                cb_int1,
                                cb_build_cast_integer (p
->occursDepending),
                                cb_build_string0 ((
ucharptr) name));
            e1;
            e2;
        } else {
            if (!CB_LITERAL_P (sub)) {
                e1 = cb_build_funcall_4 (
                "cob_check_subscript",
                cb_build_cast_integer (sub),
                cb_int1,
                cb_int (p->
occurs_max),
                cb_build_string0
                ((ucharptr) name));
                r->check = cb_list_add (r
->check, e1);
            }
        }
    }
}

/* reference modification check */
if (r->offset) {
    /* compile-time check */
    if (CB_LITERAL_P (r->offset)) {
        offset = cb_get_int (r->offset);
        if (offset < 1 || offset > f->size) {
            cb_error_x (x, _("Offset of '%s' out of bounds: %
d"), name, offset);
        } else if (r->length && CB_LITERAL_P (r->length)) {
            length = cb_get_int (r->length);
            if (length < 1 || length > f->size - offset + 1)
                cb_error_x (x, _("Length of '%s' out of b
ounds: %d"),
                            name, length);
        }
    }
}

```

```

        /* run-time check */
        if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_REF_MOD)) {
            if (!CB_LITERAL_P (r->offset)
                || (r->length && !CB_LITERAL_P (r->length))) {
                e1 = cb_build_funcall_4 ("cob_check_ref_mod",
                    cb_build_cast_integer (r
->offset),
                    r->length ?
cb_build_cast_integer (r->length) :
cb_int1, cb_int (f->
size),
                    cb_build_string0 ((
ucharptr)f->name));
                r->check = cb_list_add (r->check, e1);
            }
        }
    }

    if (f->storage == CB_STORAGE_CONSTANT) {
        return CB_VALUE (f->values);
    }

    return x;
}

```

#### 7.17.2.11 **cb\_tree** **cb\_build\_index** ( **cb\_tree** *x*, **cb\_tree** *values*, **int** *indexed\_by*, **struct** **cb\_field** \* *qual* )

Definition at line 744 of file typeck.c.

```

{
    struct cb_field *f;

    f = CB_FIELD (cb_build_field (x));
    f->usage = CB_USAGE_INDEX;
    cb_validate_field (f);
    if (values) {
        f->values = cb_list_init (values);
    }
    if (qual) {
        f->index_qual = qual;
    }
    f->flag_indexed_by = indexed_by;
    current_program->working_storage = cb_field_add (current_program->
working_storage, f);
    return x;
}

```

#### 7.17.2.12 **cb\_tree** **cb\_build\_inspect\_region** ( **cb\_tree** *l*, **cb\_tree** *pos*, **cb\_tree** *x* )

Definition at line 3859 of file typeck.c.

```

{
    if (pos == CB_BEFORE) {
        return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_before",

```

```

x));
} else {
    return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_after", x
));
}
}

```

#### 7.17.2.13 `cb_tree cb_build_inspect_region_start ( void )`

Definition at line 3853 of file typeck.c.

```

{
    return cb_list_init (cb_build_funcall_0 ("cob_inspect_start"));
}

```

#### 7.17.2.14 `cb_tree cb_build_length ( cb_tree x )`

Definition at line 986 of file typeck.c.

```

{
    struct cb_field      *f;
    struct cb_literal    *l;
    cb_tree              temp;
    char                 buff[64];

    if (x == cb_error_node) {
        return cb_error_node;
    }
    if (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node) {
        return cb_error_node;
    }

    memset (buff, 0, sizeof (buff));
    if (CB_LITERAL_P (x)) {
        l = CB_LITERAL (x);
        sprintf (buff, "%d", (int)l->size);
        return cb_build_numeric_literal (0, (ucharptr)buff, 0);
    }
    if (CB_REF_OR_FIELD_P (x)) {
        f = CB_FIELD (cb_ref (x));
        if (f->flag_any_length) {
            return cb_build_any_intrinsic (cb_list_init (x));
        }
        if (cb_field_variable_size (f) == NULL) {
            sprintf (buff, "%d", cb_field_size (x));
            return cb_build_numeric_literal (0, (ucharptr)buff, 0);
        }
    }
    if (CB_INTRINSIC_P (x)) {
        return cb_build_any_intrinsic (cb_list_init (x));
    }
    temp = cb_build_index (cb_build_filler (), NULL, 0, NULL);
    CB_FIELD (cb_ref (temp))->usage = CB_USAGE_LENGTH;
    CB_FIELD (cb_ref (temp))->count++;
    cb_emit (cb_build_assign (temp, cb_build_length_1 (x)));
    return temp;
}

```

7.17.2.15 `cb_tree cb_build_move ( cb_tree src, cb_tree dst )`

Definition at line 4964 of file typeck.c.

```
{
    struct cb_field *f;
    struct cb_field *p;

    if (src == cb_error_node || dst == cb_error_node) {
        return cb_error_node;
    }

    if (validate_move (src, dst, 0) < 0) {
        return cb_error_node;
    }

    if (CB_REFERENCE_P (src)) {
        CB_REFERENCE (src)->type = CB_SENDING_OPERAND;
    }
    if (CB_REFERENCE_P (dst)) {
        CB_REFERENCE (dst)->type = CB_RECEIVING_OPERAND;
    }

    if (CB_TREE_CLASS (dst) == CB_CLASS_POINTER) {
        return cb_build_assign (dst, src);
    }

    if (CB_REFERENCE_P (src) && CB_ALPHABET_NAME_P (CB_REFERENCE (src)->value))
    {
        return cb_build_move_call (src, dst);
    }
    if (CB_INDEX_P (dst)) {
        if (src == cb_null) {
            return cb_build_assign (dst, cb_zero);
        }
        return cb_build_assign (dst, src);
    }

    if (CB_INDEX_P (src)) {
        return cb_build_funcall_2 ("cob_set_int", dst,
cb_build_cast_integer (src));
    }

    if (CB_INTRINSIC_P (src) || CB_INTRINSIC_P (dst)) {
        return cb_build_move_call (src, dst);
    }

    f = cb_field (dst);

    if (CB_EXCEPTION_ENABLE (COB_EC_BOUND_SUBSCRIPT)) {
        for (p = f; p; p = p->parent) {
            if (p->flag_occurs) {
                return cb_build_move_call (src, dst);
            }
        }
        if (CB_REF_OR_FIELD_P (src)) {
            for (p = cb_field (src); p; p = p->parent) {
                if (p->flag_occurs) {
                    return cb_build_move_call (src, dst);
                }
            }
        }
    }
}
```

```

    }

    /* output optimal code */
    if (src == cb_zero) {
        return cb_build_move_zero (dst);
    } else if (src == cb_space) {
        return cb_build_move_space (dst);
    } else if (src == cb_high) {
        return cb_build_move_high (dst);
    } else if (src == cb_low) {
        return cb_build_move_low (dst);
    } else if (src == cb_quote) {
        return cb_build_move_quote (dst);
    } else if (CB_LITERAL_P (src)) {
        return cb_build_move_literal (src, dst);
    }
    return cb_build_move_field (src, dst);
}

```

#### 7.17.2.16 **cb\_tree** **cb\_build\_perform.exit** ( **struct cb\_label** \* *label* )

Definition at line 5156 of file typeck.c.

```

{
    cb_tree x;

    x = cb_build_perform (CB_PERFORM_EXIT);
    CB_PERFORM (x)->data = CB_TREE (label);
    return x;
}

```

#### 7.17.2.17 **cb\_tree** **cb\_build\_perform.forever** ( **cb\_tree** *body* )

Definition at line 5143 of file typeck.c.

```

{
    cb_tree x;

    if (body == cb_error_node) {
        return cb_error_node;
    }
    x = cb_build_perform (CB_PERFORM_FOREVER);
    CB_PERFORM (x)->body = body;
    return x;
}

```

#### 7.17.2.18 **cb\_tree** **cb\_build\_perform.once** ( **cb\_tree** *body* )

Definition at line 5105 of file typeck.c.

```

{
    cb_tree x;

```

```
    if (body == cb_error_node) {
        return cb_error_node;
    }
    x = cb_build_perform (CB_PERFORM_ONCE);
    CB_PERFORM (x)->body = body;
    return x;
}
```

#### 7.17.2.19 `cb_tree cb_build_perform_times ( cb_tree times )`

Definition at line 5118 of file typeck.c.

```
{
    cb_tree x;

    if (cb_check_integer_value (times) == cb_error_node) {
        return cb_error_node;
    }

    x = cb_build_perform (CB_PERFORM_TIMES);
    CB_PERFORM (x)->data = times;
    return x;
}
```

#### 7.17.2.20 `cb_tree cb_build_perform_until ( cb_tree condition, cb_tree varying )`

Definition at line 5132 of file typeck.c.

```
{
    cb_tree x;

    x = cb_build_perform (CB_PERFORM_UNTIL);
    CB_PERFORM (x)->test = condition;
    CB_PERFORM (x)->varying = varying;
    return x;
}
```

#### 7.17.2.21 `cb_tree cb_build_ppointer ( cb_tree x )`

Definition at line 1038 of file typeck.c.

```
{
    struct cb_field *f;

    if (x == cb_error_node ||
        (CB_REFERENCE_P (x) && cb_ref (x) == cb_error_node)) {
        return cb_error_node;
    }

    if (CB_REFERENCE_P (x)) {
        f = cb_field (cb_ref(x));
    }
}
```

```

        f->count++;
    }
    return cb_build_cast_ppointer (x);
}

```

### 7.17.2.22 `const char* cb_build_program_id ( cb_tree name, cb_tree alt_name )`

Definition at line 591 of file typeck.c.

```

{
    const char    *s;

    /* This needs some more thought, should we generate an entry
       point per program source name ?
       if (alt_name) {
           s = (char *)CB_LITERAL (alt_name)->data;
       } else if (CB_LITERAL_P (name)) {
           s = (char *)CB_LITERAL (name)->data;
       } else {
           s = (char *)CB_NAME (name);
       }

       if (!cb_flag_main && strcmp (s, source_name)) {
           cb_warning (_("Source name '%s' differs from PROGRAM-ID '%s'"),
                       source_name, s);
           current_program->source_name = strdup (source_name);
       }
    End comment out */

    if (alt_name) {
        current_program->orig_source_name = strdup ((char *)CB_LITERAL (a
lt_name)->data);
        s = (char *)CB_LITERAL (alt_name)->data;
    } else if (CB_LITERAL_P (name)) {
        current_program->orig_source_name = strdup ((char *)CB_LITERAL (n
ame)->data);
        s = cb_encode_program_id ((char *)CB_LITERAL (name)->data);
    } else {
        current_program->orig_source_name = strdup (CB_NAME (name));
        s = cb_encode_program_id (CB_NAME (name));
    }
    if (cobc_check_valid_name (current_program->orig_source_name)) {
        cb_error (_("PROGRAM-ID '%s' invalid"), current_program->
orig_source_name);
    }
    return s;
}

```

### 7.17.2.23 `void cb_build_registers ( void )`

Definition at line 494 of file typeck.c.

```

{
#ifdef !defined(__linux__) && !defined(__CYGWIN__) && defined(HAVE_TIMEZONE)
    long    contz;

```



```

#endif
    time_t  t;
    char    buff[48];

    /* RETURN-CODE */
    if (!current_program->nested_level) {
        current_program->cb_return_code =
            cb_build_index (cb_build_reference ("RETURN-CODE"),
                           cb_zero, 0, NULL);
        cb_field (current_program->cb_return_code)->flag_is_global = 1;
    }

    /* SORT-RETURN */
    current_program->cb_sort_return =
        cb_build_index (cb_build_reference ("SORT-RETURN"), cb_zero, 0,
NULL);
    cb_field (current_program->cb_sort_return)->flag_no_init = 1;

    /* NUMBER-OF-CALL-PARAMETERS */
    current_program->cb_call_params =
        cb_build_index (cb_build_reference ("NUMBER-OF-CALL-PARAMETERS"),
cb_zero, 0, NULL);
    cb_field (current_program->cb_call_params)->flag_no_init = 1;

    /* TALLY */
    /* 01 TALLY GLOBAL PICTURE 9(9) USAGE COMP-5 VALUE ZERO. */
    /* TALLY/EXAMINE not standard/supported */

    t = time (NULL);

    /* WHEN-COMPILED */
    memset (buff, 0, sizeof (buff));
    strftime (buff, 17, "%m/%d/%y%H.%M.%S", localtime (&t));
    cb_build_constant (cb_build_reference ("WHEN-COMPILED"),
        cb_build_alphanumeric_literal ((ucharptr)buff, 16));

    /* FUNCTION WHEN-COMPILED */
    memset (buff, 0, sizeof (buff));
    #if defined(__linux__) || defined(__CYGWIN__)
        strftime (buff, 22, "%Y%m%d%H%M%S00z", localtime (&t));
    #elif defined(HAVE_TIMEZONE)
        strftime (buff, 17, "%Y%m%d%H%M%S00", localtime (&t));
        if (timezone <= 0) {
            contz = -timezone;
            buff[16] = '+';
        } else {
            contz = timezone;
            buff[16] = '-';
        }
    }
    sprintf (&buff[17], "%2.2ld%2.2ld", contz / 3600, contz % 60);
    #else
        strftime (buff, 22, "%Y%m%d%H%M%S0000000", localtime (&t));
    #endif
    cb_intr_whencomp = cb_build_alphanumeric_literal ((ucharptr)buff, 21);

    /* FUNCTION PI */
    memset (buff, 0, sizeof (buff));
    strcpy (buff, "31415926535897932384626433832795029");
    cb_intr_pi = cb_build_numeric_literal (0, (ucharptr)buff, 34);

    /* FUNCTION E */
    memset (buff, 0, sizeof (buff));

```

```
strcpy (buff, "27182818284590452353602874713526625");
cb_intr_e = cb_build_numeric_literal (0, (ucharptr)buff, 34);
}
```

#### 7.17.2.24 `cb_tree cb_build_replacing_all ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3823 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_all", y, x));
}
```

#### 7.17.2.25 `cb_tree cb_build_replacing_characters ( cb_tree x, cb_tree l )`

Definition at line 3817 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_1 ("cob_inspect_characters", x));
}
```

#### 7.17.2.26 `cb_tree cb_build_replacing_first ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3835 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_first", y, x));
}
```

#### 7.17.2.27 `cb_tree cb_build_replacing_leading ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3829 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_leading", y, x));
}
```

#### 7.17.2.28 `cb_tree cb_build_replacing_trailing ( cb_tree x, cb_tree y, cb_tree l )`

Definition at line 3841 of file typeck.c.

```
{
    return cb_list_add (l, cb_build_funcall_2 ("cob_inspect_trailing", y, x))
;
}
```

7.17.2.29 `cb_tree cb_build_section_name ( cb_tree name, int sect_or_para )`

Definition at line 653 of file typeck.c.

```

{
    cb_tree x;

    if (name == cb_error_node) {
        return cb_error_node;
    }

    if (CB_REFERENCE (name)->word->count > 0) {
        x = CB_VALUE (CB_REFERENCE (name)->word->items);
        /* Used as a non-label name or used as a section name.
           Duplicate paragraphs are allowed if not referenced;
           Checked in typeck.c */
        if (!CB_LABEL_P (x) || sect_or_para == 0
            || (sect_or_para && CB_LABEL_P (x) && CB_LABEL (x)->is_sectio
n)) {
            redefinition_error (name);
            return cb_error_node;
        }
    }

    return name;
}

```

7.17.2.30 `cb_tree cb_build_sub ( cb_tree v, cb_tree n, cb_tree round_opt )`

Definition at line 2633 of file typeck.c.

```

{
    cb_tree      opt;
    struct cb_field *f;

#ifdef COB_NON_ALIGNED
    if (CB_INDEX_P (v)) {
        return cb_build_move (cb_build_binary_op (v, '-', n), v);
    }
    if (CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        current_program->gen_ptrmanip = 1;
        return cb_build_funcall_3 ("cob_pointer_manip", v, n, cb_int1);
    }
#else
    if (CB_INDEX_P (v) || CB_TREE_CLASS (v) == CB_CLASS_POINTER) {
        return cb_build_move (cb_build_binary_op (v, '-', n), v);
    }
#endif

    if (CB_REF_OR_FIELD_P (v)) {
        f = cb_field (v);
        f->count++;
    }
    if (CB_REF_OR_FIELD_P (n)) {
        f = cb_field (n);
        f->count++;
    }
    opt = build_store_option (v, round_opt);
}

```

```

    if (opt == cb_int0 && cb_fits_int (n)) {
        return cb_build_optim_sub (v, n);
    }
    return cb_build_funcall_3 ("cob_sub", v, n, opt);
}

```

#### 7.17.2.31 **cb\_tree** **cb\_build\_tarrying\_all** ( void )

Definition at line 3778 of file typeck.c.

```

{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before ALL"));
    }
    inspect_func = "cob_inspect_all";
    return NULL;
}

```

#### 7.17.2.32 **cb\_tree** **cb\_build\_tarrying\_characters** ( **cb\_tree** / )

Definition at line 3768 of file typeck.c.

```

{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before CHARACTERS"));
    }
    inspect_func = NULL;
    return cb_list_add (1, cb_build_funcall_1 ("cob_inspect_characters", insp
ect_data));
}

```

#### 7.17.2.33 **cb\_tree** **cb\_build\_tarrying\_data** ( **cb\_tree** x )

Definition at line 3761 of file typeck.c.

```

{
    inspect_data = x;
    return NULL;
}

```

#### 7.17.2.34 **cb\_tree** **cb\_build\_tarrying\_leading** ( void )

Definition at line 3788 of file typeck.c.

```

{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before LEADING"));
    }
    inspect_func = "cob_inspect_leading";
    return NULL;
}

```

**7.17.2.35 `cb_tree` `cb_build_tarrying_trailing` ( `void` )**

Definition at line 3798 of file typeck.c.

```
{
    if (inspect_data == NULL) {
        cb_error (_("Data name expected before TRAILING"));
    }
    inspect_func = "cob_inspect_trailing";
    return NULL;
}
```

**7.17.2.36 `cb_tree` `cb_build_tarrying_value` ( `cb_tree x`, `cb_tree l` )**

Definition at line 3808 of file typeck.c.

```
{
    if (inspect_func == NULL) {
        cb_error_x (x, _("ALL, LEADING or TRAILING expected before '%s'"),
        , cb_name (x));
    }
    return cb_list_add (l, cb_build_funcall_2 (inspect_func, inspect_data, x)
    );
}
```

**7.17.2.37 `cb_tree` `cb_build_unstring_delimited` ( `cb_tree all`, `cb_tree value` )**

Definition at line 5866 of file typeck.c.

```
{
    if (cb_validate_one (value)) {
        return cb_error_node;
    }
    return cb_build_funcall_2 ("cob_unstring_delimited", value, all);
}
```

**7.17.2.38 `cb_tree` `cb_build_unstring_into` ( `cb_tree name`, `cb_tree delimiter`, `cb_tree count` )**

Definition at line 5875 of file typeck.c.

```
{
    if (cb_validate_one (name)) {
        return cb_error_node;
    }
    if (delimiter == NULL) {
        delimiter = cb_int0;
    }
    if (count == NULL) {
        count = cb_int0;
    }
    return cb_build_funcall_3 ("cob_unstring_into", name, delimiter, count);
}
```

**7.17.2.39 `cb_tree cb_build_write_advancing_lines ( cb_tree pos, cb_tree lines )`**

Definition at line 5957 of file typeck.c.

```

{
    cb_tree e;
    int     opt;

    opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
    e = cb_build_binary_op (cb_int (opt | COB_WRITE_LINES), '+', lines);
    return cb_build_cast_integer (e);
}

```

**7.17.2.40 `cb_tree cb_build_write_advancing_mnemonic ( cb_tree pos, cb_tree mnemonic )`**

Definition at line 5968 of file typeck.c.

```

{
    int     opt;
    int     token;

    token = CB_SYSTEM_NAME (cb_ref (mnemonic))->token;
    switch (token) {
    case CB_FEATURE_FORMFEED:
        opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
        return cb_int (opt | COB_WRITE_PAGE);
    case CB_FEATURE_C01:
    case CB_FEATURE_C02:
    case CB_FEATURE_C03:
    case CB_FEATURE_C04:
    case CB_FEATURE_C05:
    case CB_FEATURE_C06:
    case CB_FEATURE_C07:
    case CB_FEATURE_C08:
    case CB_FEATURE_C09:
    case CB_FEATURE_C10:
    case CB_FEATURE_C11:
    case CB_FEATURE_C12:
        opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;
        return cb_int (opt | COB_WRITE_CHANNEL | COB_WRITE_PAGE | token);
    default:
        cb_error_x (mnemonic, _("Invalid mnemonic name"));
        return cb_error_node;
    }
}

```

**7.17.2.41 `cb_tree cb_build_write_advancing_page ( cb_tree pos )`**

Definition at line 5999 of file typeck.c.

```

{
    int opt = (pos == CB_BEFORE) ? COB_WRITE_BEFORE : COB_WRITE_AFTER;

    return cb_int (opt | COB_WRITE_PAGE);
}

```

7.17.2.42 `cb_tree cb_check_numeric_value ( cb_tree x )`

Definition at line 426 of file typeck.c.

```
{
    if (x == cb_error_node) {
        return cb_error_node;
    }

    if (CB_TREE_CATEGORY (x) == CB_CATEGORY_NUMERIC) {
        return x;
    }

    cb_error_x (x, _("%s' is not a numeric value"), cb_name (x));
    return cb_error_node;
}
```

7.17.2.43 `void cb_define_switch_name ( cb_tree name, cb_tree sname, cb_tree flag, cb_tree ref )`

Definition at line 629 of file typeck.c.

```
{
    cb_tree switch_id;
    cb_tree value;

    if (name == cb_error_node) {
        return;
    }
    if (sname == cb_error_node) {
        return;
    }
    if (CB_SYSTEM_NAME (sname)->category != CB_SWITCH_NAME) {
        cb_error_x (ref, _("Switch-name is expected '%s'"), CB_NAME (ref)
);
    } else {
        switch_id = cb_int (CB_SYSTEM_NAME (sname)->token);
        value = cb_build_funcall_1 ("cob_get_switch", switch_id);
        if (flag == cb_int0) {
            value = cb_build_negation (value);
        }
        cb_build_constant (name, value);
    }
}
```

7.17.2.44 `void cb_emit_accept ( cb_tree var, cb_tree pos, cb_tree fgc, cb_tree bgc, cb_tree scroll, int dispatrs )`

Definition at line 2802 of file typeck.c.

```
{
    cb_tree line;
    cb_tree column;
```

```

if (cb_validate_one (var)) {
    return;
}
if (cb_validate_one (pos)) {
    return;
}
if (cb_validate_one (fgc)) {
    return;
}
if (cb_validate_one (bgc)) {
    return;
}
if (cb_validate_one (scroll)) {
    return;
}
if (current_program->flag_screen) {
    /* Bump ref count to force CRT STATUS field generation */
    cb_field (current_program->crt_status)->count++;
    if ((CB_REF_OR_FIELD_P (var)) &&
        CB_FIELD (cb_ref (var))->storage == CB_STORAGE_SCREEN) {
        output_screen_from (CB_FIELD (cb_ref (var)), 0);
        gen_screen_ptr = 1;
        if (pos) {
            if (CB_PAIR_P (pos)) {
                line = CB_PAIR_X (pos);
                column = CB_PAIR_Y (pos);
                cb_emit (cb_build_funcall_3 ("cob_screen_
accept",
                                           var, line, column));
            } else {
                cb_emit (cb_build_funcall_3 ("cob_screen_
accept",
                                           var, pos, NULL));
            }
        } else {
            cb_emit (cb_build_funcall_3 ("cob_screen_accept",
                                        var, NULL, NULL));
        }
        gen_screen_ptr = 0;
        output_screen_to (CB_FIELD (cb_ref (var)), 0);
    } else {
        if (pos || fgc || bgc) {
            if (!pos) {
                cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                           var, NULL, NULL, fgc, bgc,
                                           scroll, cb_int (dispattr));
            } else if (CB_PAIR_P (pos)) {
                line = CB_PAIR_X (pos);
                column = CB_PAIR_Y (pos);
                cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                           var, line, column, fgc, bgc,
                                           scroll, cb_int (dispattr));
            } else {
                cb_emit (cb_build_funcall_7 ("cob_field_a
ccept",
                                           var, pos, NULL, fgc, bgc,
                                           scroll, cb_int (dispattr));
            }
        } else {

```



```

        cb_emit (cb_build_funcall_7 ("cob_field_accept",
                                     var, NULL, NULL, fgc, bgc,
                                     scroll, cb_int (dispattrs)));
    }
} else if (pos || fgc || bgc || scroll) {
    /* Bump ref count to force CRT STATUS field generation */
    cb_field (current_program->crt_status)->count++;
    if (!pos) {
        cb_emit (cb_build_funcall_7 ("cob_field_accept",
                                     var, NULL, NULL, fgc, bgc, scroll,
                                     cb_int (dispattrs)));
    } else if (CB_PAIR_P (pos)) {
        line = CB_PAIR_X (pos);
        column = CB_PAIR_Y (pos);
        cb_emit (cb_build_funcall_7 ("cob_field_accept",
                                     var, line, column, fgc, bgc, scroll,
                                     cb_int (dispattrs)));
    } else {
        cb_emit (cb_build_funcall_7 ("cob_field_accept",
                                     var, pos, NULL, fgc, bgc, scroll,
                                     cb_int (dispattrs)));
    }
} else {
    cb_emit (cb_build_funcall_1 ("cob_accept", var));
}
}

```

#### 7.17.2.45 void cb\_emit\_accept\_arg\_number ( cb\_tree var )

Definition at line 2986 of file typeck.c.

```

{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_arg_number", var));
}

```

#### 7.17.2.46 void cb\_emit\_accept\_arg\_value ( cb\_tree var )

Definition at line 2995 of file typeck.c.

```

{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_arg_value", var));
}

```

#### 7.17.2.47 void cb\_emit\_accept\_command\_line ( cb\_tree var )

Definition at line 2956 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_command_line", var));
}
```

#### 7.17.2.48 void cb\_emit\_accept\_date ( cb\_tree var )

Definition at line 2902 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_date", var));
}
```

#### 7.17.2.49 void cb\_emit\_accept\_date\_yyyymmdd ( cb\_tree var )

Definition at line 2911 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_date_yyyymmdd", var));
}
```

#### 7.17.2.50 void cb\_emit\_accept\_day ( cb\_tree var )

Definition at line 2920 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_day", var));
}
```

#### 7.17.2.51 void cb\_emit\_accept\_day\_of\_week ( cb\_tree var )

Definition at line 2938 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_day_of_week", var));
}
```

**7.17.2.52 void cb\_emit\_accept\_day\_yyyyddd ( cb\_tree var )**

Definition at line 2929 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_day_yyyyddd", var));
}
```

**7.17.2.53 void cb\_emit\_accept\_environment ( cb\_tree var )**

Definition at line 2977 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_environment", var));
}
```

**7.17.2.54 void cb\_emit\_accept\_line\_or\_col ( cb\_tree var, const int l\_or\_c )**

Definition at line 2893 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_2 ("cob_screen_line_col", var, cb_int (l_or_c))
);
}
```

**7.17.2.55 void cb\_emit\_accept\_mnemonic ( cb\_tree var, cb\_tree mnemonic )**

Definition at line 3004 of file typeck.c.

```
{
    if (cb_validate_one (var)) {
        return;
    }
    switch (CB_SYSTEM_NAME (cb_ref (mnemonic))->token) {
    case CB_DEVICE_CONSOLE:
    case CB_DEVICE_SYSIN:
        cb_emit (cb_build_funcall_1 ("cob_accept", var));
        break;
    default:
        cb_error_x (mnemonic, _("Invalid input stream '%s'"),
            cb_name (mnemonic));
        break;
    }
}
```

**7.17.2.56 void cb\_emit\_accept.name ( cb\_tree var, cb\_tree name )**

Definition at line 3022 of file typeck.c.

```

{
    cb_tree sys;

    if (cb_validate_one (var)) {
        return;
    }
    if (CB_REFERENCE (name)->word->count == 0) {
        sys = lookup_system_name (CB_NAME (name));

        if (sys != cb_error_node) {
            switch (CB_SYSTEM_NAME (sys)->token) {
                case CB_DEVICE_CONSOLE:
                case CB_DEVICE_SYSIN:
                    cb_warning_x (name, _("%s' undefined in SPECIAL-
NAMES"), CB_NAME (name));
                    cb_emit (cb_build_funcall_1 ("cob_accept", var));
                    return;
                default:
                    break;
            }
        }

        cb_error_x (name, _("%s' undefined in SPECIAL-NAMES"), CB_NAME (name));
    }
}

```

**7.17.2.57 void cb\_emit\_accept.time ( cb\_tree var )**

Definition at line 2947 of file typeck.c.

```

{
    if (cb_validate_one (var)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_accept_time", var));
}

```

**7.17.2.58 void cb\_emit\_allocate ( cb\_tree target1, cb\_tree target2, cb\_tree size, cb\_tree initialize )**

Definition at line 3053 of file typeck.c.

```

{
    cb_tree x;
    char buff[32];

    if (cb_validate_one (target1)) {
        return;
    }
}

```

```

    if (cb_validate_one (target2)) {
        return;
    }
    if (cb_validate_one (size)) {
        return;
    }
    if (target1) {
        if (!(CB_REFERENCE_P(target1) &&
            cb_field (target1)->flag_item_based)) {
            cb_error_x (CB_TREE(current_statement),
                _("Target of ALLOCATE is not a BASED item"));
        }
    }
    if (target2) {
        if (!(CB_REFERENCE_P(target2) &&
            CB_TREE_CLASS (target2) == CB_CLASS_POINTER)) {
            cb_error_x (CB_TREE(current_statement),
                _("Target of RETURNING is not a data pointer"));
        }
    }
    if (size) {
        if (CB_TREE_CLASS (size) != CB_CLASS_NUMERIC) {
            cb_error_x (CB_TREE(current_statement),
                _("The CHARACTERS field of ALLOCATE must be numeric"));
        }
    }
    if (target1) {
        sprintf (buff, "%d", cb_field (target1)->memory_size);
        x = cb_build_numeric_literal (0, (ucharptr)buff, 0);
        cb_emit (cb_build_funcall_3 ("cob_allocate",
            cb_build_cast_addr_of_addr (target1), target2, x));
    } else {
        cb_emit (cb_build_funcall_3 ("cob_allocate",
            NULL, target2, size));
    }
    if (initialize && target1) {
        current_statement->handler2 =
            cb_build_initialize (target1, cb_true, NULL, cb_true, 0);
    }
}

```

#### 7.17.2.59 void cb\_emit\_arg\_number ( cb\_tree value )

Definition at line 3273 of file typeck.c.

```

{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_arg_number", value));
}

```

#### 7.17.2.60 void cb\_emit\_arithmetic ( cb\_tree vars, int op, cb\_tree val )

Definition at line 2094 of file typeck.c.

```

{
    cb_tree      l;
    struct cb_field *f;

    val = cb_check_numeric_value (val);
    if (op) {
        cb_list_map (cb_check_numeric_name, vars);
    } else {
        cb_list_map (cb_check_numeric_edited_name, vars);
    }

    if (cb_validate_one (val)) {
        return;
    }
    if (cb_validate_list (vars)) {
        return;
    }

    if (!CB_BINARY_OP_P (val)) {
        if (op == '+' || op == '-') {
            if (CB_EXCEPTION_ENABLE (COB_EC_DATA_INCOMPATIBLE) &&
                (CB_REF_OR_FIELD_P (val))) {
                f = cb_field (val);
                if (f->usage == CB_USAGE_DISPLAY ||
                    f->usage == CB_USAGE_PACKED) {
                    cb_emit (cb_build_funcall_2 ("cob_check_n
umeric",
                                                val,
                                                cb_build_string0 ((
ucharptr) (f->name))));
                }
            }
            for (l = vars; l; l = CB_CHAIN (l)) {
                if (CB_EXCEPTION_ENABLE (COB_EC_DATA_INCOMPATIBLE
) &&
                    (CB_REF_OR_FIELD_P (CB_VALUE(l)))) {
                    f = cb_field (CB_VALUE(l));
                    if (f->usage == CB_USAGE_DISPLAY ||
                        f->usage == CB_USAGE_PACKED) {
                        cb_emit (cb_build_funcall_2 ("cob
_check_numeric",
                                                    CB_VALUE(l),
                                                    cb_build_string0 ((
ucharptr) (f->name))));
                    }
                }
                if (op == '+') {
                    CB_VALUE (l) = cb_build_add (CB_VALUE (l)
, val, CB_PURPOSE (l));
                } else {
                    CB_VALUE (l) = cb_build_sub (CB_VALUE (l)
, val, CB_PURPOSE (l));
                }
            }
            cb_emit_list (vars);
            return;
        }
    }

    cb_emit (build_decimal_assign (vars, op, val));
}

```

7.17.2.61 void `cb.emit.call ( cb_tree prog, cb_tree using, cb_tree returning, cb_tree on_exception, cb_tree not_on_exception )`

Definition at line 3108 of file `typeck.c`.

```

{
    cb_tree          l;
    cb_tree          x;
    const struct system_table *psyst;
    int              is_sys_call = 0;

    if (CB_INTRINSIC_P (prog)) {
        if (CB_INTRINSIC(prog)->intr_tab->category !=
CB_CATEGORY_ALPHANUMERIC) {
            cb_error (_("Only alphanumeric FUNCTION types are allowed
here"));
            return;
        }
    }
    if (returning) {
        if (CB_TREE_CLASS(returning) != CB_CLASS_NUMERIC &&
            CB_TREE_CLASS(returning) != CB_CLASS_POINTER) {
            cb_error (_("Invalid RETURNING field"));
            return;
        }
    }
    for (l = using; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            continue;
        }
        if (CB_CONST_P (x) && x != cb_null) {
            cb_error_x (x, _("Figurative constant invalid here"));
        }
        if ((CB_REFERENCE_P (x) && CB_FIELD_P(CB_REFERENCE(x)->value)
|| CB_FIELD_P (x)) {
            if (cb_field (x)->level == 88) {
                cb_error_x (x, _("%s' Not a data name"),
CB_NAME (x));
                return;
            }
            if (cb_warn_call_params &&
                CB_PURPOSE_INT (l) == CB_CALL_BY_REFERENCE) {
                if (cb_field (x)->level != 01 &&
                    cb_field (x)->level != 77) {
                    cb_warning_x (x, _("%s' is not 01 or 77
level item"), CB_NAME (x));
                }
            }
        }
    }

    if (CB_LITERAL_P(prog)) {
        for (psyst = (const struct system_table *)&system_tab[0]; psyst->
syst_name; psyst++) {
            if (!strcmp((const char *)CB_LITERAL(prog)->data,
                (const char *)psyst->syst_name)) {
                if (psyst->syst_params > cb_list_length (using))
{
                    cb_error (_("Wrong number of CALL paramet
ers for '%s'"),

```

```

                                (char *)psyst->sys_name);
                                return;
                                }
                                is_sys_call = 1;
                                break;
                                }
                                }
                                }
                                }

                                cb_emit (cb_build_call (prog, using, on_exception, not_on_exception,
                                returning, is_sys_call));
                                }

```

#### 7.17.2.62 void cb\_emit\_cancel ( cb\_tree prog )

Definition at line 3177 of file typeck.c.

```

{
    if (cb_validate_one (prog)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_field_cancel", prog));
}

```

#### 7.17.2.63 void cb\_emit\_close ( cb\_tree file, cb\_tree opt )

Definition at line 3190 of file typeck.c.

```

{
    if (file == cb_error_node) {
        return;
    }
    file = cb_ref (file);
    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    cb_emit (cb_build_funcall_3 ("cob_close", file, opt,
        CB_FILE(file)->file_status));
}

```

#### 7.17.2.64 void cb\_emit\_command\_line ( cb\_tree value )

Definition at line 3282 of file typeck.c.

```

{
    if (cb_validate_one (value)) {
        return;
    }
}

```



```
    }
    cb_emit (cb_build_funcall_1 ("cob_display_command_line", value));
}
```

#### 7.17.2.65 void cb.emit.commit ( void )

Definition at line 3213 of file typeck.c.

```
{
    cb_emit (cb_build_funcall_0 ("cob_commit"));
}
```

#### 7.17.2.66 void cb.emit.continue ( void )

Definition at line 3223 of file typeck.c.

```
{
    cb_emit (cb_build_continue ());
}
```

#### 7.17.2.67 void cb.emit.corresponding ( cb\_tree\*(cb\_tree f1, cb\_tree f2, cb\_tree f3) func, cb\_tree x1, cb\_tree x2, cb\_tree opt )

Definition at line 2695 of file typeck.c.

```
{
    x1 = cb_check_group_name (x1);
    x2 = cb_check_group_name (x2);

    if (cb_validate_one (x1)) {
        return;
    }
    if (cb_validate_one (x2)) {
        return;
    }

    emit_corresponding (func, x1, x2, opt);
}
```

#### 7.17.2.68 void cb.emit.delete ( cb\_tree file )

Definition at line 3233 of file typeck.c.

```
{
    if (file == cb_error_node) {
        return;
    }
    file = cb_ref (file);
}
```

```

    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    cb_emit (cb_build_funcall_2 ("cob_delete", file, CB_FILE(file)->file_stat
us));
}

```

**7.17.2.69** void `cb_emit_display ( cb_tree values, cb_tree upon, cb_tree no_adv, cb_tree pos, cb_tree fg, cb_tree bg, cb_tree scroll, int dispatts )`

Definition at line 3291 of file typeck.c.

```

{
    cb_tree l;
    cb_tree x;
    cb_tree line;
    cb_tree column;
    cb_tree p;

    if (cb_validate_list (values)) {
        return;
    }
    if (cb_validate_one (pos)) {
        return;
    }
    if (cb_validate_one (fg)) {
        return;
    }
    if (cb_validate_one (bg)) {
        return;
    }
    if (cb_validate_one (scroll)) {
        return;
    }
    for (l = values; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            return;
        }

        switch (CB_TREE_TAG (x)) {
            case CB_TAG_LITERAL:
            case CB_TAG_INTRINSIC:
            case CB_TAG_CONST:
            case CB_TAG_STRING:
            case CB_TAG_INTEGER:
                break;
            case CB_TAG_REFERENCE:
                if (!CB_FIELD_P (CB_REFERENCE (x)->value)) {
                    cb_error_x (x, _("'%s' is an invalid type for DIS
PLAY operand"), cb_name (x));
                    return;
                }
                break;
        }
    }
}

```

```

        default:
            cb_error_x (x, _("Invalid type for DISPLAY operand"));
            return;
        }
    }
    if (upon == cb_error_node) {
        return;
    }

    x = CB_VALUE (values);
    if ((CB_REF_OR_FIELD_P (x)) &&
        CB_FIELD (cb_ref (x))->storage == CB_STORAGE_SCREEN) {
        output_screen_from (CB_FIELD (cb_ref (x)), 0);
        gen_screen_ptr = 1;
        if (pos) {
            if (CB_PAIR_P (pos)) {
                line = CB_PAIR_X (pos);
                column = CB_PAIR_Y (pos);
                if (line == NULL) {
                    line = cb_one;
                }
                if (column == NULL) {
                    column = cb_one;
                }
                cb_emit (cb_build_funcall_3 ("cob_screen_display"
, x,
, x,
, x,
                line, column));
            } else {
                cb_emit (cb_build_funcall_3 ("cob_screen_display"
, x,
, x,
                pos, NULL));
            }
        } else {
            cb_emit (cb_build_funcall_3 ("cob_screen_display", x,
            NULL, NULL));
        }
        gen_screen_ptr = 0;
    } else if (pos || fgc || bgc || scroll || dispattrs) {
        if (!pos) {
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), NULL, NULL, fgc, bgc,
            scroll, cb_int (dispattrs)));
        } else if (CB_PAIR_P (pos)) {
            line = CB_PAIR_X (pos);
            column = CB_PAIR_Y (pos);
            if (line == NULL) {
                line = cb_one;
            }
            if (column == NULL) {
                column = cb_one;
            }
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), line, column, fgc, bgc,
            scroll, cb_int (dispattrs)));
        } else {
            cb_emit (cb_build_funcall_7 ("cob_field_display",
            CB_VALUE (values), pos, NULL, fgc, bgc,
            scroll, cb_int (dispattrs)));
        }
    } else {
        /* DISPLAY x ... [UPON device-name] */
        p = cb_build_funcall_3 ("cob_display", upon, no_adv, values);
    }

```

```

        CB_FUNCALL(p)->varcnt = cb_list_length (values);
        cb_emit (p);
        for (l = values; l; l = CB_CHAIN (l)) {
            x = CB_VALUE (l);
            if (CB_FIELD_P (x)) {
                CB_FIELD (cb_ref (x))->count++;
            }
        }
    }
}

```

#### 7.17.2.70 void `cb_emit_divide` ( *cb\_tree dividend*, *cb\_tree divisor*, *cb\_tree quotient*, *cb\_tree remainder* )

Definition at line 3460 of file `typeck.c`.

```

{
    if (cb_validate_one (dividend)) {
        return;
    }
    if (cb_validate_one (divisor)) {
        return;
    }
    CB_VALUE (quotient) = cb_check_numeric_edited_name (CB_VALUE (quotient));
    CB_VALUE (remainder) = cb_check_numeric_edited_name (CB_VALUE (remainder)
);
    if (cb_validate_one (CB_VALUE (quotient))) {
        return;
    }
    if (cb_validate_one (CB_VALUE (remainder))) {
        return;
    }
    cb_emit (cb_build_funcall_4 ("cob_div_quotient", dividend, divisor,
                                CB_VALUE (quotient),
                                build_store_option (CB_VALUE (quotient),
CB_PURPOSE (quotient))));
    cb_emit (cb_build_funcall_2 ("cob_div_remainder", CB_VALUE (remainder),
                                build_store_option (CB_VALUE (remainder),
cb_int0)));
}

```

#### 7.17.2.71 void `cb_emit_env_name` ( *cb\_tree value* )

Definition at line 3255 of file `typeck.c`.

```

{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_environment", value));
}

```

**7.17.272 void cb\_emit\_env\_value ( cb\_tree value )**

Definition at line 3264 of file typeck.c.

```
{
    if (cb_validate_one (value)) {
        return;
    }
    cb_emit (cb_build_funcall_1 ("cob_display_env_value", value));
}
```

**7.17.273 void cb\_emit\_evaluate ( cb\_tree subject\_list, cb\_tree case\_list )**

Definition at line 3601 of file typeck.c.

```
{
    cb_emit (build_evaluate (subject_list, case_list));
}
```

**7.17.274 void cb\_emit\_exit ( size\_t goback )**

Definition at line 3675 of file typeck.c.

```
{
    if (goback) {
        cb_emit (cb_build_goto (cb_int1, NULL));
    } else {
        cb_emit (cb_build_goto (NULL, NULL));
    }
}
```

**7.17.275 void cb\_emit\_free ( cb\_tree vars )**

Definition at line 3611 of file typeck.c.

```
{
    cb_tree l;
    struct cb_field *f;
    int i;

    if (cb_validate_list (vars)) {
        return;
    }
    for (l = vars, i = 1; l; l = CB_CHAIN (l), i++) {
        if (CB_TREE_CLASS (CB_VALUE (l)) == CB_CLASS_POINTER) {
            if (CB_CAST_P (CB_VALUE (l))) {
                f = cb_field (CB_CAST (CB_VALUE (l))->val);
                if (!f->flag_item_based) {
                    cb_error_x (CB_TREE (current_statement),
                                _("Target %d of FREE, a data addr
```

```

    ess identifier, must address a BASED data item"), i);
        }
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        cb_build_cast_address (CB_VALUE (1)),
NULL));
        } else {
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        NULL, cb_build_cast_address (CB_VALUE (1)
)));
        }
        } else if (CB_REF_OR_FIELD_P (CB_VALUE (1))) {
        f = cb_field (CB_VALUE (1));
        if (!f->flag_item_based) {
        cb_error_x (CB_TREE (current_statement),
        _("Target %d of FREE, a data addr
ess identifier, must address a BASED data item"), i);
        }
        cb_emit (cb_build_funcall_2 ("cob_free_alloc",
        cb_build_cast_addr_of_addr (CB_VALUE (1))
, NULL));
        } else {
        cb_error_x (CB_TREE (current_statement),
        _("Target %d of FREE must be a data pointer"), i)
;
        }
    }
}

```

#### 7.17.2.76 void `cb_emit_get_environment` ( `cb_tree envvar`, `cb_tree envval` )

Definition at line 2965 of file `typeck.c`.

```

{
    if (cb_validate_one (envvar)) {
        return;
    }
    if (cb_validate_one (envval)) {
        return;
    }
    cb_emit (cb_build_funcall_2 ("cob_get_environment", envvar, envval));
}

```

#### 7.17.2.77 void `cb_emit_goto` ( `cb_tree target`, `cb_tree depending` )

Definition at line 3654 of file `typeck.c`.

```

{
    if (target == cb_error_node) {
        return;
    }
    if (depending) {
        /* GO TO procedure-name ... DEPENDING ON identifier */
        cb_emit (cb_build_goto (target, depending));
    } else {
        /* GO TO procedure-name */
    }
}

```

```

        if (target == NULL) {
            cb_verify (cb_goto_statement_without_name, "GO TO without
procedure-name");
        } else if (CB_CHAIN (target)) {
            cb_error (_("GO TO with multiple procedure-names"));
        } else {
            cb_emit (cb_build_goto (CB_VALUE (target), NULL));
        }
    }
}

```

#### 7.17.2.78 void cb.emit.if ( *cb\_tree cond*, *cb\_tree stmt1*, *cb\_tree stmt2* )

Definition at line 3689 of file typeck.c.

```

{
    cb_emit (cb_build_if (cond, stmt1, stmt2));
}

```

#### 7.17.2.79 void cb.emit.initialize ( *cb\_tree vars*, *cb\_tree fillinit*, *cb\_tree value*, *cb\_tree replacing*, *cb\_tree def* )

Definition at line 3699 of file typeck.c.

```

{
    cb_tree l;
    int fill_init = 1;

    if (cb_validate_list (vars)) {
        return;
    }
    if (value == NULL && replacing == NULL) {
        def = cb_true;
    }
    if (fillinit == cb_true) {
        fill_init = 0;
    }
    for (l = vars; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_initialize (CB_VALUE (l), value, replacing, def
, fill_init));
    }
}

```

#### 7.17.2.80 void cb.emit.inspect ( *cb\_tree var*, *cb\_tree body*, *cb\_tree replacing*, *int replconv* )

Definition at line 3723 of file typeck.c.

```

{
    switch (CB_TREE_TAG(var)) {
        case CB_TAG_REFERENCE:

```

```

        break;
    case CB_TAG_INTRINSIC:
        switch (CB_TREE_CATEGORY(var)) {
            case CB_CATEGORY_ALPHABETIC:
            case CB_CATEGORY_ALPHANUMERIC:
            case CB_CATEGORY_NATIONAL:
                break;
            default:
                cb_error (_("Invalid target for INSPECT"));
                return;
        }
        break;
    case CB_TAG_LITERAL:
        break;
    default:
        cb_error (_("Invalid target for REPLACING/CONVERTING"));
        return;
}
if (replconv && sending_id) {
    cb_error (_("Invalid target for REPLACING/CONVERTING"));
}
cb_emit (cb_build_funcall_2 ("cob_inspect_init", var, replacing));
cb_emit_list (body);
cb_emit (cb_build_funcall_0 ("cob_inspect_finish"));
}

```

#### 7.17.2.81 void cb\_emit\_move ( cb\_tree src, cb\_tree dsts )

Definition at line 5041 of file typeck.c.

```

{
    cb_tree l;

    if (cb_validate_one (src)) {
        return;
    }
    if (cb_validate_list (dsts)) {
        return;
    }

    for (l = dsts; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_move (src, CB_VALUE (l)));
    }
}

```

#### 7.17.2.82 void cb\_emit\_move\_corresponding ( cb\_tree x1, cb\_tree x2 )

Definition at line 2738 of file typeck.c.

```

{
    cb_tree      l;
    cb_tree      v;

    x1 = cb_check_group_name (x1);
    if (cb_validate_one (x1)) {

```



```

        return;
    }
    for (l = x2; l; l = CB_CHAIN(l)) {
        v = CB_VALUE(l);
        v = cb_check_group_name (v);
        if (cb_validate_one (v)) {
            return;
        }
        emit_move_corresponding (x1, v);
    }
}

```

### 7.17.2.83 void cb.emit\_open ( cb\_tree file, cb\_tree mode, cb\_tree sharing )

Definition at line 5062 of file typeck.c.

```

{
    if (file == cb_error_node) {
        return;
    }
    file = cb_ref (file);
    if (file == cb_error_node) {
        return;
    }
    current_statement->file = file;

    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    if (sharing == NULL) {
        sharing = CB_FILE (file)->sharing ? CB_FILE (file)->sharing :
cb_int0;
    }

    /* READ ONLY */
    if (sharing == cb_int0 && CB_INTEGER (mode)->val != COB_OPEN_INPUT) {
        sharing = cb_int1;
    }

    cb_emit (cb_build_funcall_4 ("cob_open", file, mode,
        sharing, CB_FILE(file)->file_status));
}

```

### 7.17.2.84 void cb.emit\_perform ( cb\_tree perform, cb\_tree body )

Definition at line 5095 of file typeck.c.

```

{
    if (perform == cb_error_node) {
        return;
    }
    CB_PERFORM (perform)->body = body;
    cb_emit (perform);
}

```

### 7.17.2.85 void cb\_emit\_read ( cb\_tree ref, cb\_tree next, cb\_tree into, cb\_tree key, cb\_tree lock\_opts )

Definition at line 5170 of file typeck.c.

```

{
    int      read_opts = 0;
    cb_tree file;
    cb_tree rec;

    if (lock_opts == cb_int1) {
        read_opts = COB_READ_LOCK;
    } else if (lock_opts == cb_int2) {
        read_opts = COB_READ_NO_LOCK;
    } else if (lock_opts == cb_int3) {
        read_opts = COB_READ_IGNORE_LOCK;
    } else if (lock_opts == cb_int4) {
        read_opts = COB_READ_WAIT_LOCK;
    }
    if (ref == cb_error_node) {
        return;
    }
    file = cb_ref (ref);
    if (file == cb_error_node) {
        return;
    }
    rec = cb_build_field_reference (CB_FILE (file)->record, ref);
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
            _("Operation not allowed on SORT files"));
    }
    if (next == cb_int1 || next == cb_int2 ||
        CB_FILE (file)->access_mode == COB_ACCESS_SEQUENTIAL) {
        /* READ NEXT/PREVIOUS */
        if (next == cb_int2) {
            if (CB_FILE (file)->organization != COB_ORG_INDEXED) {
                cb_error_x (CB_TREE (current_statement),
                    _("READ PREVIOUS only allowed for INDEXED SEQUENTIAL files"));
            }
            read_opts |= COB_READ_PREVIOUS;
        } else {
            read_opts |= COB_READ_NEXT;
        }
    }
    if (key) {
        cb_warning (_("KEY ignored with sequential READ"));
    }
    cb_emit (cb_build_funcall_4 ("cob_read", file, cb_int0,
        CB_FILE(file)->file_status,
        cb_int (read_opts)));
} else {
    /* READ */
    cb_emit (cb_build_funcall_4 ("cob_read",
        file, key ? key : CB_FILE (file)->key,
        CB_FILE(file)->file_status, cb_int (read_opts)));
}
    if (into) {
        current_statement->handler3 = cb_build_move (rec, into);
    }
    current_statement->file = file;
}

```

## 7.17.2.86 void cb\_emit\_release ( cb\_tree record, cb\_tree from )

Definition at line 5278 of file typeck.c.

```

{
    struct cb_field *f;
    cb_tree file;

    if (record == cb_error_node) {
        return;
    }
    if (from == cb_error_node) {
        return;
    }
    if (cb_ref (record) == cb_error_node) {
        return;
    }
    if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE requires a record name as subject"));
        return;
    }
    if (cb_field (record)->storage != CB_STORAGE_FILE) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE subject does not refer to a record name"));
        return;
    }
    f = CB_FIELD (cb_ref (record));
    file = CB_TREE (f->file);
    if (CB_FILE (file)->organization != COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
                    _("RELEASE not allowed on this record item"));
        return;
    }
    current_statement->file = file;
    if (from) {
        cb_emit (cb_build_move (from, record));
    }
    cb_emit (cb_build_funcall_1 ("cob_file_release", file));
}

```

## 7.17.2.87 void cb\_emit\_return ( cb\_tree ref, cb\_tree into )

Definition at line 5321 of file typeck.c.

```

{
    cb_tree file;
    cb_tree rec;

    if (ref == cb_error_node) {
        return;
    }
    if (into == cb_error_node) {
        return;
    }
    file = cb_ref (ref);
    if (file == cb_error_node) {
        return;
    }
}

```

```

}
rec = cb_build_field_reference (CB_FILE (file)->record, ref);
cb_emit (cb_build_funcall_1 ("cob_file_return", file));
if (into) {
    current_statement->handler3 = cb_build_move (rec, into);
}
current_statement->file = file;
}

```

#### 7.17.2.88 void cb\_emit\_rewrite ( cb\_tree record, cb\_tree from, cb\_tree lockopt )

Definition at line 5232 of file typeck.c.

```

{
    cb_tree file;
    int     opts = 0;

    if (record == cb_error_node || cb_ref (record) == cb_error_node) {
        return;
    }
    if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
        cb_error_x (CB_TREE (current_statement),
                    _("REWRITE requires a record name as subject"));
        return;
    }
    if (cb_field (record)->storage != CB_STORAGE_FILE) {
        cb_error_x (CB_TREE (current_statement),
                    _("REWRITE subject does not refer to a record name"));
        return;
    }
    file = CB_TREE (CB_FIELD (cb_ref (record))->file);
    current_statement->file = file;
    if (CB_FILE (file)->organization == COB_ORG_SORT) {
        cb_error_x (CB_TREE (current_statement),
                    _("Operation not allowed on SORT files"));
    } else if (current_statement->handler_id == COB_EC_I_O_INVALID_KEY &&
               (CB_FILE (file)->organization != COB_ORG_RELATIVE &&
                CB_FILE (file)->organization != COB_ORG_INDEXED)) {
        cb_error_x (CB_TREE (current_statement),
                    _("INVALID KEY clause invalid with this file type"));
    } else if ((CB_FILE (file)->lock_mode & COB_LOCK_AUTOMATIC) && lockopt) {
        cb_error_x (CB_TREE (current_statement),
                    _("LOCK clause invalid with file LOCK AUTOMATIC"));
    } else if (lockopt == cb_int1) {
        opts = COB_WRITE_LOCK;
    }
    if (from) {
        cb_emit (cb_build_move (from, record));
    }
    cb_emit (cb_build_funcall_4 ("cob_rewrite", file, record,
                                cb_int (opts), CB_FILE (file)->file_status));
}

```

#### 7.17.2.89 void cb\_emit\_rollback ( void )

Definition at line 5349 of file typeck.c.

```
{
    cb_emit (cb_build_funcall_0 ("cob_rollback"));
}
```

#### 7.17.2.90 void cb.emit\_search ( cb\_tree table, cb\_tree varying, cb\_tree at\_end, cb\_tree whens )

Definition at line 5427 of file typeck.c.

```
{
    if (cb_validate_one (table)) {
        return;
    }
    if (cb_validate_one (varying)) {
        return;
    }
    if (table == cb_error_node) {
        return;
    }
    cb_emit (cb_build_search (0, table, varying, at_end, whens));
}
```

#### 7.17.2.91 void cb.emit\_search\_all ( cb\_tree table, cb\_tree at\_end, cb\_tree when, cb\_tree stmts )

Definition at line 5442 of file typeck.c.

```
{
    if (cb_validate_one (table)) {
        return;
    }
    if (table == cb_error_node) {
        return;
    }
    cb_emit (cb_build_search (1, table, NULL, at_end,
                             cb_build_if (cb_build_search_all (table, when),
                             stmts, NULL)));
}
```

#### 7.17.2.92 void cb.emit\_set\_false ( cb\_tree l )

Definition at line 5604 of file typeck.c.

```
{
    cb_tree      x;
    struct cb_field *f;
    cb_tree      ref;
    cb_tree      val;

    for (; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
    }
}
```

```

        if (x == cb_error_node) {
            return;
        }
        if (!(CB_REFERENCE_P (x) && CB_FIELD_P(CB_REFERENCE(x)->value))
            && !CB_FIELD_P (x)) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        f = cb_field (x);
        if (f->level != 88) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        if (!f->>false_88) {
            cb_error_x (x, _("Field does not have FALSE clause"));
            return;
        }
        ref = cb_build_field_reference (f->parent, x);
        val = CB_VALUE (f->>false_88);
        if (CB_PAIR_P (val)) {
            val = CB_PAIR_X (val);
        }
        cb_emit (cb_build_move (val, ref));
    }
}

```

#### 7.17.2.93 void cb\_emit\_set\_on.off ( cb\_tree l, cb\_tree flag )

Definition at line 5558 of file typeck.c.

```

{
    struct cb_system_name *s;

    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {
        s = CB_SYSTEM_NAME (cb_ref (CB_VALUE (l)));
        cb_emit (cb_build_funcall_2 ("cob_set_switch", cb_int (s->token),
flag));
    }
}

```

#### 7.17.2.94 void cb\_emit\_set\_to ( cb\_tree vars, cb\_tree x )

Definition at line 5465 of file typeck.c.

```

{
    cb_tree      l;
    cb_tree      v;
    struct cb_cast *p;

#ifdef 0
    enum cb_class class = CB_CLASS_UNKNOWN;
#endif
}

```

```

    if (cb_validate_one (x)) {
        return;
    }
    if (cb_validate_list (vars)) {
        return;
    }
#endif
    /* determine the class of targets */
    for (l = vars; l; l = CB_CHAIN (l)) {
        if (CB_TREE_CLASS (CB_VALUE (l)) != CB_CLASS_UNKNOWN) {
            if (class == CB_CLASS_UNKNOWN) {
                class = CB_TREE_CLASS (CB_VALUE (l));
            } else if (class != CB_TREE_CLASS (CB_VALUE (l))) {
                break;
            }
        }
    }
    if (l || (class != CB_CLASS_INDEX && class != CB_CLASS_POINTER)) {
        cb_error_x (CB_TREE (current_statement),
            _("The targets of SET must be either indexes or pointers"));
        return;
    }
#endif

    if (CB_CAST_P (x)) {
        p = CB_CAST (x);
        if (p->type == CB_CAST_PROGRAM_POINTER) {
            for (l = vars; l; l = CB_CHAIN (l)) {
                v = CB_VALUE (l);
                if (!CB_REFERENCE_P (v)) {
                    cb_error_x (CB_TREE (current_statement),
                        _("SET targets must be PROGRAM-POINTER"));
                }
                CB_VALUE (l) = cb_error_node;
            } else if (CB_FIELD(cb_ref(v))->usage !=
CB_USAGE_PROGRAM_POINTER) {
                cb_error_x (CB_TREE (current_statement),
                    _("SET targets must be PROGRAM-POINTER"));
            }
            CB_VALUE (l) = cb_error_node;
        }
    }
}
/* validate the targets */
for (l = vars; l; l = CB_CHAIN (l)) {
    v = CB_VALUE (l);
    if (CB_CAST_P (v)) {
        p = CB_CAST (v);
        if (p->type == CB_CAST_ADDRESS
            && !CB_FIELD (cb_ref (p->val))->flag_item_based
            && CB_FIELD (cb_ref (p->val))->storage !=
CB_STORAGE_LINKAGE) {
            cb_error_x (p->val, _("The address of '%s' cannot
be changed"),
                cb_name (p->val));
            CB_VALUE (l) = cb_error_node;
        }
    }
}

```

```

    if (cb_validate_list (vars)) {
        return;
    }

    for (l = vars; l; l = CB_CHAIN (l)) {
        cb_emit (cb_build_move (x, CB_VALUE (l)));
    }
}

```

#### 7.17.2.95 void cb\_emit\_set\_true ( cb\_tree l )

Definition at line 5572 of file typeck.c.

```

{
    cb_tree      x;
    struct cb_field *f;
    cb_tree      ref;
    cb_tree      val;

    for (; l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        if (x == cb_error_node) {
            return;
        }
        if (!(CB_REFERENCE_P (x) && CB_FIELD_P (CB_REFERENCE (x)->value))
            && !CB_FIELD_P (x)) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        f = cb_field (x);
        if (f->level != 88) {
            cb_error_x (x, _("Invalid SET statement"));
            return;
        }
        ref = cb_build_field_reference (f->parent, x);
        val = CB_VALUE (f->values);
        if (CB_PAIR_P (val)) {
            val = CB_PAIR_X (val);
        }
        cb_emit (cb_build_move (val, ref));
    }
}

```

#### 7.17.2.96 void cb\_emit\_set\_up.down ( cb\_tree l, cb\_tree flag, cb\_tree x )

Definition at line 5540 of file typeck.c.

```

{
    if (cb_validate_one (x)) {
        return;
    }
    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {

```



```

        if (flag == cb_int0) {
            cb_emit (cb_build_add (CB_VALUE (l), x, cb_int0));
        } else {
            cb_emit (cb_build_sub (CB_VALUE (l), x, cb_int0));
        }
    }
}

```

#### 7.17.2.97 void cb.emit\_setenv ( cb\_tree x, cb\_tree y )

Definition at line 5459 of file typeck.c.

```

{
    cb_emit (cb_build_funcall_2 ("cob_set_environment", x, y));
}

```

#### 7.17.2.98 void cb.emit\_sort\_finish ( cb\_tree file )

Definition at line 5741 of file typeck.c.

```

{
    if (CB_FILE_P (cb_ref (file))) {
        cb_emit (cb_build_funcall_1 ("cob_file_sort_close", cb_ref (file)
    ));
    }
}

```

#### 7.17.2.99 void cb.emit\_sort\_giving ( cb\_tree file, cb\_tree l )

Definition at line 5715 of file typeck.c.

```

{
    cb_tree      p;
    int          listlen;

    if (cb_validate_list (l)) {
        return;
    }
    for (p = l; p; p = CB_CHAIN (p)) {
        if (CB_FILE (cb_ref(CB_VALUE(p)))->organization == COB_ORG_SORT)
        {
            cb_error (_("Invalid SORT GIVING parameter"));
        }
    }
    listlen = cb_list_length (l);
    p = cb_build_funcall_2 ("cob_file_sort_giving", cb_ref (file), l);
    CB_FUNCALL(p)->varcnt = listlen;
    cb_emit (p);
}

```

## 7.17.2.100 void cb\_emit\_sort\_init ( cb\_tree name, cb\_tree keys, cb\_tree col )

Definition at line 5644 of file typeck.c.

```

{
    cb_tree      l;
    struct cb_field *f;

    if (cb_validate_list (keys)) {
        return;
    }
    for (l = keys; l; l = CB_CHAIN (l)) {
        if (CB_VALUE (l) == NULL) {
            CB_VALUE (l) = name;
        }
        cb_ref (CB_VALUE (l));
    }

    if (CB_FILE_P (cb_ref (name))) {
        if (CB_FILE (cb_ref (name))->organization != COB_ORG_SORT) {
            cb_error_x (name, _("Invalid SORT filename"));
        }
        cb_field (current_program->cb_sort_return)->count++;
        cb_emit (cb_build_funcall_5 ("cob_file_sort_init", cb_ref (name),
                                     cb_int (cb_list_length (keys)), col,
                                     cb_build_cast_address (
current_program->cb_sort_return),
                                     CB_FILE(cb_ref (name))->file_status)
);
        for (l = keys; l; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_4 ("cob_file_sort_init_key",
cb_ref (name),
                                     CB_PURPOSE (l),
                                     CB_VALUE (l),
                                     cb_int (cb_field (CB_VALUE(l))->offset)))
;
        }
    } else {
        f = CB_FIELD (cb_ref (name));
        if (keys == NULL) {
            cb_error_x (name, _("Table sort without keys not implemen
ted yet"));
        }
        cb_emit (cb_build_funcall_2 ("cob_table_sort_init", cb_int (
cb_list_length (keys)), col));
        for (l = keys; l; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_3 ("cob_table_sort_init_key",
                                     CB_PURPOSE (l),
                                     CB_VALUE (l),
                                     cb_int (cb_field (CB_VALUE(l))->offset)))
;
        }
        cb_emit (cb_build_funcall_2 ("cob_table_sort", name,
                                     (f->occurs Depending
                                     ? cb_build_cast_integer (f->
occurs Depending)
                                     : cb_int (f->occurs_max))));
    }
}

```

**7.17.2.101 void cb\_emit\_sort\_input ( cb\_tree proc )**

Definition at line 5709 of file typeck.c.

```
{
    cb_emit (cb_build_perform_once (proc));
}
```

**7.17.2.102 void cb\_emit\_sort\_output ( cb\_tree proc )**

Definition at line 5735 of file typeck.c.

```
{
    cb_emit (cb_build_perform_once (proc));
}
```

**7.17.2.103 void cb\_emit\_sort\_using ( cb\_tree file, cb\_tree l )**

Definition at line 5694 of file typeck.c.

```
{
    if (cb_validate_list (l)) {
        return;
    }
    for (; l; l = CB_CHAIN (l)) {
        if (CB_FILE (cb_ref(CB_VALUE(l)))->organization == COB_ORG_SORT)
        {
            cb_error (_("Invalid SORT USING parameter"));
        }
        cb_emit (cb_build_funcall_2 ("cob_file_sort_using",
            cb_ref (file), cb_ref (CB_VALUE (l))));
    }
}
```

**7.17.2.104 void cb\_emit\_start ( cb\_tree file, cb\_tree op, cb\_tree key )**

Definition at line 5753 of file typeck.c.

```
{
    if (cb_validate_one (key)) {
        return;
    }
    if (file != cb_error_node) {
        current_statement->file = cb_ref (file);
        cb_emit (cb_build_funcall_4 ("cob_start", cb_ref (file), op,
            key ? key : CB_FILE (cb_ref (file))-
            >key,
            CB_FILE(cb_ref(file))->file_statu
            s));
    }
}
```

**7.17.2.105 void cb.emit.stop\_run ( cb\_tree x )**

Definition at line 5771 of file typeck.c.

```
{
    cb_emit (cb_build_funcall_1 ("cob_stop_run", cb_build_cast_integer (x)));
}
```

**7.17.2.106 void cb.emit.string ( cb\_tree items, cb\_tree into, cb\_tree pointer )**

Definition at line 5781 of file typeck.c.

```
{
    cb_tree start;
    cb_tree l;
    cb_tree end;
    cb_tree dlm;

    if (cb_validate_one (into)) {
        return;
    }
    if (cb_validate_one (pointer)) {
        return;
    }
    start = items;
    cb_emit (cb_build_funcall_2 ("cob_string_init", into, pointer));
    while (start) {

        /* find DELIMITED item */
        for (end = start; end; end = CB_CHAIN (end)) {
            if (CB_PAIR_P (CB_VALUE (end))) {
                break;
            }
        }

        /* cob_string_delimited */
        dlm = end ? CB_PAIR_X (CB_VALUE (end)) : cb_int0;
        cb_emit (cb_build_funcall_1 ("cob_string_delimited", dlm));

        /* cob_string_append */
        for (l = start; l != end; l = CB_CHAIN (l)) {
            cb_emit (cb_build_funcall_1 ("cob_string_append",
CB_VALUE (l)));
        }

        start = end ? CB_CHAIN (end) : NULL;
    }
    cb_emit (cb_build_funcall_0 ("cob_string_finish"));
}
```

**7.17.2.107 void cb.emit.unlock ( cb\_tree ref )**

Definition at line 5824 of file typeck.c.

```

{
    cb_tree file;

    if (ref != cb_error_node) {
        file = cb_ref (ref);
        cb_emit (cb_build_funcall_2 ("cob_unlock_file",
            file, CB_FILE(file)->file_status));
        current_statement->file = file;
    }
}

```

#### 7.17.2.108 void cb\_emit\_unstring ( cb\_tree name, cb\_tree delimited, cb\_tree into, cb\_tree pointer, cb\_tree tallying )

Definition at line 5841 of file typeck.c.

```

{
    if (cb_validate_one (name)) {
        return;
    }
    if (cb_validate_one (tallying)) {
        return;
    }
    if (cb_validate_list (delimited)) {
        return;
    }
    if (cb_validate_list (into)) {
        return;
    }
    cb_emit (cb_build_funcall_3 ("cob_unstring_init", name, pointer,
        cb_int (cb_list_length (delimited))));
    cb_emit_list (delimited);
    cb_emit_list (into);
    if (tallying) {
        cb_emit (cb_build_funcall_1 ("cob_unstring_tallying", tallying));
    }
    cb_emit (cb_build_funcall_0 ("cob_unstring_finish"));
}

```

#### 7.17.2.109 void cb\_emit\_write ( cb\_tree record, cb\_tree from, cb\_tree opt, cb\_tree lockopt )

Definition at line 5894 of file typeck.c.

```

{
    cb_tree    file;
    int        val;

    if (record != cb_error_node && cb_ref (record) != cb_error_node) {
        if (!CB_REF_OR_FIELD_P (cb_ref (record))) {
            cb_error_x (CB_TREE (current_statement),
                _("WRITE requires a record name as subject"));
            return;
        }
    }
}

```

```

        if (cb_field (record)->storage != CB_STORAGE_FILE) {
            cb_error_x (CB_TREE (current_statement),
                _("WRITE subject does not refer to a record name"
));
            return;
        }
        file = CB_TREE (CB_FIELD (cb_ref (record))->file);
        current_statement->file = file;
        if (CB_FILE (file)->organization == COB_ORG_SORT) {
            cb_error_x (CB_TREE (current_statement),
                _("Operation not allowed on SORT files"));
        } else if (current_statement->handler_id == COB_EC_I_O_INVALID_KEY
Y &&
            (CB_FILE(file)->organization != COB_ORG_RELATIVE &&
            CB_FILE(file)->organization != COB_ORG_INDEXED)) {
            cb_error_x (CB_TREE(current_statement),
                _("INVALID KEY clause invalid with this file type
"));
        } else if (lockopt) {
            if ((CB_FILE (file)->lock_mode & COB_LOCK_AUTOMATIC)) {
                cb_error_x (CB_TREE (current_statement),
                    _("LOCK clause invalid with file LOCK AUTOMATIC"
));
            } else if (opt != cb_int0) {
                cb_error_x (CB_TREE (current_statement),
                    _("LOCK clause invalid here"));
            } else if (lockopt == cb_int1) {
                opt = cb_int (COB_WRITE_LOCK);
            }
        }
        if (from) {
            cb_emit (cb_build_move (from, record));
        }
        if (CB_FILE (file)->organization == COB_ORG_LINE_SEQUENTIAL &&
            opt == cb_int0) {
            opt = cb_int (COB_WRITE_BEFORE | COB_WRITE_LINES | 1);
        }
        /* RXW - This is horrible */
        if (current_statement->handler_id == COB_EC_I_O_EOP &&
            current_statement->handler1) {
            if (CB_CAST_P(opt)) {
                val = CB_INTEGER(CB_BINARY_OP(CB_CAST(opt)->val)-
>x)->val;
                val |= COB_WRITE_EOP;
                CB_BINARY_OP (CB_CAST(opt)->val)->x = cb_int (val)
;
            } else {
                val = CB_INTEGER(opt)->val;
                val |= COB_WRITE_EOP;
                opt = cb_int (val);
            }
        }
        cb_emit (cb_build_funcall_4 ("cob_write", file, record, opt,
            CB_FILE(file)->file_status));
    }
}

```

#### 7.17.2.110 char\* cb\_encode\_program\_id ( const char \* name )

Definition at line 563 of file typeck.c.

```

{
    unsigned char      *p;
    const unsigned char *s;
    unsigned char      buff[COB_SMALL_BUFF];

    p = buff;
    s = (const unsigned char *)name;
    /* encode the initial digit */
    if (isdigit (*s)) {
        p += sprintf ((char *)p, "_%02X", *s++);
    }
    /* encode invalid letters */
    for (; *s; s++) {
        if (isalnum (*s) || *s == '_') {
            *p++ = *s;
        } else if (*s == '-') {
            *p++ = '_';
            *p++ = '_';
        } else {
            p += sprintf ((char *)p, "_%02X", *s);
        }
    }
    *p = 0;
    return strdup ((char *)buff);
}

```

#### 7.17.2.111 void cb\_init.tarrying ( void )

Definition at line 3754 of file typeck.c.

```

{
    inspect_func = NULL;
    inspect_data = NULL;
}

```

#### 7.17.2.112 void cb\_validate\_program.body ( struct cb\_program \* prog )

Definition at line 1416 of file typeck.c.

```

{
    /* resolve all labels */
    cb_tree l;
    cb_tree x;
    cb_tree v;

    for (l = cb_list_reverse (prog->label_list); l; l = CB_CHAIN (l)) {
        x = CB_VALUE (l);
        v = cb_ref (x);
        if (CB_LABEL_P (v)) {
            CB_LABEL (v)->need_begin = 1;
            if (CB_REFERENCE (x)->length) {
                CB_LABEL (v)->need_return = 1;
            }
        } else if (v != cb_error_node) {
            cb_error_x (x, _("%s' not procedure name"), cb_name (x))
        }
    }
}

```

```

;
    }
}

prog->file_list = cb_list_reverse (prog->file_list);
prog->exec_list = cb_list_reverse (prog->exec_list);
}

```

### 7.17.2.113 void cb\_validate\_program\_data ( struct cb\_program \* prog )

Definition at line 1286 of file typeck.c.

```

{
    cb_tree      l;
    cb_tree      x;
    cb_tree      assign;
    struct cb_field *p;
    struct cb_file *f;
    unsigned char *c;

    for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
        f = CB_FILE (CB_VALUE (l));
        if (!f->finalized) {
            finalize_file (f, NULL);
        }
    }
    /* build undeclared assignment name now */
    if (cb_assign_clause == CB_ASSIGN_MF) {
        for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
            assign = CB_FILE (CB_VALUE (l))->assign;
            if (!assign) {
                continue;
            }
            if (CB_REFERENCE_P (assign)) {
                for (x = current_program->file_list; x; x =
CB_CHAIN (x)) {
                    if (!strcmp (CB_FILE (CB_VALUE (x))->
name,
                                CB_REFERENCE (assign)->word->name))
                    {
                        redefinition_error (assign);
                    }
                }
                p = check_level_78 (CB_REFERENCE (assign)->word->
name);
                if (p) {
                    c = (unsigned char *)CB_LITERAL(CB_VALUE (
p->values))->data;
                    assign = CB_TREE (build_literal (
CB_CATEGORY_ALPHANUMERIC, c, strlen ((char *)c));
                    CB_FILE (CB_VALUE (l))->assign = assign;
                }
            }
            if (CB_REFERENCE_P (assign) && CB_REFERENCE (assign)->wor
d->count == 0) {
                if (cb_warn_implicit_define) {
                    cb_warning (_("'s' will be implicitly de
fined"), CB_NAME (assign));
                }
            }
        }
    }
}

```



```

        x = cb_build_implicit_field (assign,
COB_SMALL_BUFF);
        p = current_program->working_storage;
        CB_FIELD (x)->count++;
        if (p) {
            while (p->sister) {
                p = p->sister;
            }
            p->sister = CB_FIELD (x);
        } else {
            current_program->working_storage =
CB_FIELD (x);
        }
        if (CB_REFERENCE_P (assign)) {
            x = cb_ref (assign);
            if (CB_FIELD_P (x) && CB_FIELD (x)->level == 88)
{
                cb_error_x (assign, _("ASSIGN data item '
%s' invalid"), CB_NAME (assign));
            }
        }
    }

    if (prog->cursor_pos) {
        x = cb_ref (prog->cursor_pos);
        if (x == cb_error_node) {
            prog->cursor_pos = NULL;
        } else if (CB_FIELD(x)->size != 6 && CB_FIELD(x)->size != 4) {
            cb_error_x (prog->cursor_pos, _("'%s' CURSOR is not 4 or
6 characters long"),
                cb_name (prog->cursor_pos));
            prog->cursor_pos = NULL;
        }
    }

    if (prog->crt_status) {
        x = cb_ref (prog->crt_status);
        if (x == cb_error_node) {
            prog->crt_status = NULL;
        } else if (CB_FIELD(x)->size != 4) {
            cb_error_x (prog->crt_status, _("'%s' CRT STATUS is not 4
characters long"),
                cb_name (prog->crt_status));
            prog->crt_status = NULL;
        }
    }
} else {
    l = cb_build_reference ("COB-CRT-STATUS");
    p = CB_FIELD (cb_build_field (l));
    p->usage = CB_USAGE_DISPLAY;
    p->pic = CB_PICTURE (cb_build_picture ("9(4)"));
    cb_validate_field (p);
    p->flag_no_init = 1;
    /* Do not initialize/bump ref count here
    p->values = cb_list_init (cb_zero);
    p->count++;
    */
    current_program->working_storage =
        cb_field_add (current_program->working_storage, p);
    prog->crt_status = l;
    /* RXWRXW - Maybe better
    prog->crt_status = cb_build_index (cb_build_reference ("COB-CRT-S

```

```
TATUS"), cb_zero, 0, NULL);
    */
}

/* resolve all references so far */
for (l = cb_list_reverse (prog->reference_list); l; l = CB_CHAIN (l)) {
    cb_ref (CB_VALUE (l));
}
for (l = current_program->file_list; l; l = CB_CHAIN (l)) {
    f = CB_FILE (CB_VALUE (l));
    if (f->recordDepending && f->recordDepending != cb_error_node)
    {
        x = f->recordDepending;
        if (cb_ref (x) != cb_error_node) {
/* RXW - This breaks old legacy programs
            if (CB_REF_OR_FIELD_P(x)) {
                p = cb_field (x);
                switch (p->storage) {
                    case CB_STORAGE_WORKING:
                    case CB_STORAGE_LOCAL:
                    case CB_STORAGE_LINKAGE:
                        break;
                    default:
                        cb_error (_("RECORD DEPENDING item
must be in WORKING/LOCAL/LINKAGE section"));
                }
            } else {
*/
                if (!CB_REFERENCE_P(x) && !CB_FIELD_P(x)) {
                    cb_error (_("Invalid RECORD DEPENDING item
"));
                }
            }
        }
    }
}
}
```

#### 7.17.2.114 void cb\_validate\_program\_environment ( struct cb\_program \* prog )

Definition at line 1079 of file typeck.c.

```
{
    cb_tree x;
    cb_tree y;
    cb_tree l;
    cb_tree ls;
    struct cb_alphabet_name *ap;
    unsigned char *data;
    size_t dupls;
    size_t unvals;
    size_t count;
    int lower;
    int upper;
    int size;
    int n;
    int i;
    int lastval;
    int values[256];
}
```

```

/* Check ALPHABET clauses */
for (l = current_program->alphabet_name_list; l; l = CB_CHAIN (l)) {
    ap = CB_ALPHABET_NAME (CB_VALUE (l));
    if (ap->type != CB_ALPHABET_CUSTOM) {
        continue;
    }
    ap->low_val_char = 0;
    ap->high_val_char = 255;
    dupls = 0;
    unvals = 0;
    count = 0;
    lastval = 0;
    for (n = 0; n < 256; n++) {
        values[n] = -1;
    }
    for (y = ap->custom_list; y; y = CB_CHAIN (y)) {
        if (count > 255) {
            unvals = 1;
            break;
        }
        x = CB_VALUE (y);
        if (CB_PAIR_P (x)) {
            /* X THRU Y */
            lower = get_value (CB_PAIR_X (x));
            upper = get_value (CB_PAIR_Y (x));
            lastval = upper;
            if (!count) {
                ap->low_val_char = lower;
            }
            if (lower < 0 || lower > 255) {
                unvals = 1;
                continue;
            }
            if (upper < 0 || upper > 255) {
                unvals = 1;
                continue;
            }
            if (lower <= upper) {
                for (i = lower; i <= upper; i++) {
                    if (values[i] != -1) {
                        dupls = 1;
                    }
                    values[i] = i;
                    count++;
                }
            } else {
                for (i = lower; i >= upper; i--) {
                    if (values[i] != -1) {
                        dupls = 1;
                    }
                    values[i] = i;
                    count++;
                }
            }
        } else if (CB_LIST_P (x)) {
            /* X ALSO Y ... */
            if (!count) {
                ap->low_val_char = get_value (CB_VALUE (x
));
            }
        }
        for (ls = x; ls; ls = CB_CHAIN (ls)) {
            n = get_value (CB_VALUE (ls));

```

```

        if (!CB_CHAIN (ls)) {
            lastval = n;
        }
        if (n < 0 || n > 255) {
            unvals = 1;
            continue;
        }
        if (values[n] != -1) {
            dupls = 1;
        }
        values[n] = n;
        count++;
    }
} else {
    /* literal */
    if (CB_TREE_CLASS (x) == CB_CLASS_NUMERIC) {
        n = get_value (x);
        lastval = n;
        if (!count) {
            ap->low_val_char = n;
        }
        if (n < 0 || n > 255) {
            unvals = 1;
            continue;
        }
        if (values[n] != -1) {
            dupls = 1;
        }
        values[n] = n;
        count++;
    } else if (CB_LITERAL_P (x)) {
        size = (int)CB_LITERAL (x)->size;
        data = CB_LITERAL (x)->data;
        if (!count) {
            ap->low_val_char = data[0];
        }
        lastval = data[size - 1];
        for (i = 0; i < size; i++) {
            n = data[i];
            if (values[n] != -1) {
                dupls = 1;
            }
            values[n] = n;
            count++;
        }
    } else {
        n = get_value (x);
        lastval = n;
        if (!count) {
            ap->low_val_char = n;
        }
        if (n < 0 || n > 255) {
            unvals = 1;
            continue;
        }
        if (values[n] != -1) {
            dupls = 1;
        }
        values[n] = n;
        count++;
    }
}
}

```

```

    }
    if (dupls || unvals) {
        if (dupls) {
            cb_error_x (1, _("Duplicate character values in a
alphabet '%s'"),
                        cb_name (CB_VALUE(1)));
        }
        if (unvals) {
            cb_error_x (1, _("Invalid character values in alp
phabet '%s'"),
                        cb_name (CB_VALUE(1)));
        }
        ap->low_val_char = 0;
        ap->high_val_char = 255;
        continue;
    }
    /* Calculate HIGH-VALUE */
    /* If all 256 values have been specified, HIGH-VALUE is the last
one */
    /* Otherwise if HIGH-VALUE has been specified, find the highest *
/
    /* value that has not been used */
    if (count == 256) {
        ap->high_val_char = lastval;
    } else if (values[255] != -1) {
        for (n = 254; n >= 0; n--) {
            if (values[n] == -1) {
                ap->high_val_char = n;
                break;
            }
        }
    }
}
/* Rest HIGH/LOW-VALUES */
cb_low = cb_norm_low;
cb_high = cb_norm_high;
/* resolve the program collating sequence */
if (!prog->collating_sequence) {
    return;
}
x = cb_ref (prog->collating_sequence);
/* RXWRXW
if (x == cb_error_node) {
    prog->collating_sequence = NULL;
    return;
}
*/
if (!CB_ALPHABET_NAME_P (x)) {
    cb_error_x (prog->collating_sequence, _("'%s' not alphabet name")
,
                cb_name (prog->collating_sequence));
    prog->collating_sequence = NULL;
    return;
}
if (CB_ALPHABET_NAME (x)->type != CB_ALPHABET_CUSTOM) {
    return;
}
if (CB_ALPHABET_NAME (x)->low_val_char) {
    cb_low = cb_build_alphanumeric_literal ((ucharptr)"\0", 1);
    CB_LITERAL(cb_low)->data[0] = CB_ALPHABET_NAME (x)->low_val_char;

    CB_LITERAL(cb_low)->all = 1;

```

```

    }
    if (CB_ALPHABET_NAME (x)->high_val_char != 255){
        cb_high = cb_build_alphanumeric_literal ((ucharptr)"\0", 1);
        CB_LITERAL(cb_high)->data[0] = CB_ALPHABET_NAME (x)->high_val_cha
r;
        CB_LITERAL(cb_high)->all = 1;
    }
}

```

#### 7.17.2.115 int validate\_move ( cb\_tree src, cb\_tree dst, size\_t is\_value )

Definition at line 3944 of file typeck.c.

```

{
    struct cb_field      *f;
    struct cb_literal    *l;
    unsigned char        *p;
    cb_tree              loc;
    long long            val;
    size_t               i;
    size_t               is_numeric_edited = 0;
    int                  src_scale_mod;
    int                  dst_scale_mod;
    int                  dst_size_mod;
    int                  size;
    int                  most_significant;
    int                  least_significant;

    loc = src->source_line ? src : dst;
    if (CB_REFERENCE_P(dst) && CB_ALPHABET_NAME_P(CB_REFERENCE(dst)->value))
    {
        goto invalid;
    }
    if (CB_REFERENCE_P(dst) && CB_FILE_P(CB_REFERENCE(dst)->value)) {
        goto invalid;
    }
    if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_BOOLEAN) {
        cb_error_x (loc, _("Invalid destination for MOVE"));
        return -1;
    }

    if (CB_TREE_CLASS (dst) == CB_CLASS_POINTER) {
        if (CB_TREE_CLASS (src) == CB_CLASS_POINTER) {
            return 0;
        } else {
            goto invalid;
        }
    }

    f = cb_field (dst);
    switch (CB_TREE_TAG (src)) {
    case CB_TAG_CONST:
        if (src == cb_space) {
            if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_NUMERIC
                || (CB_TREE_CATEGORY (dst) ==
CB_CATEGORY_NUMERIC_EDITED && !is_value)) {
                goto invalid;
            }
        } else if (src == cb_zero) {

```

```

        if (CB_TREE_CATEGORY (dst) == CB_CATEGORY_ALPHABETIC) {
            goto invalid;
        }
    }
    break;
case CB_TAG_LITERAL:
    /* TODO: ALL literal */

    l = CB_LITERAL (src);
    if (CB_TREE_CLASS (src) == CB_CLASS_NUMERIC) {
        /* Numeric literal */
        if (l->all) {
            goto invalid;
        }
        most_significant = -999;
        least_significant = 999;

        /* compute the most significant figure place */
        for (i = 0; i < l->size; i++) {
            if (l->data[i] != '0') {
                break;
            }
        }
        if (i != l->size) {
            most_significant = (int) (l->size - l->scale - i
- 1);
        }

        /* compute the least significant figure place */
        for (i = 0; i < l->size; i++) {
            if (l->data[l->size - i - 1] != '0') {
                break;
            }
        }
        if (i != l->size) {
            least_significant = (int) (-l->scale + i);
        }

        /* value check */
        switch (CB_TREE_CATEGORY (dst)) {
        case CB_CATEGORY_ALPHANUMERIC:
        case CB_CATEGORY_ALPHANUMERIC_EDITED:
            if (is_value) {
                goto expect_alphanumeric;
            }

            if (l->scale == 0) {
                goto expect_alphanumeric;
            } else {
                goto invalid;
            }
        case CB_CATEGORY_NUMERIC:
            if (f->pic->scale < 0) {
                /* check for PIC 9(n)P(m) */
                if (least_significant < -f->pic->scale) {
                    goto value_mismatch;
                }
            } else if (f->pic->scale > f->pic->size) {
                /* check for PIC P(n)9(m) */
                if (most_significant >= f->pic->size - f-
>pic->scale) {

```

```

                                goto value_mismatch;
                            }
                        }
                    }
                    break;
case CB_CATEGORY_NUMERIC_EDITED:
    if (is_value) {
        goto expect_alphanumeric;
    }

    /* TODO */
    break;
default:
    if (is_value) {
        goto expect_alphanumeric;
    }
    goto invalid;
}

/* sign check */
if (l->sign != 0 && !f->pic->have_sign) {
    if (is_value) {
        cb_error_x (loc, _("Data item not signed"
));
        return -1;
    }
    if (cb_warn_constant) {
        cb_warning_x (loc, _("Ignoring negative s
ign"));
    }
}

/* size check */
if (f->flag_real_binary ||
    ((f->usage == CB_USAGE_COMP_5 ||
    f->usage == CB_USAGE_COMP_X ||
    f->usage == CB_USAGE_BINARY) &&
    f->pic->scale == 0)) {
    p = l->data;
    for (i = 0; i < l->size; i++) {
        if (l->data[i] != '0') {
            p = &l->data[i];
            break;
        }
    }
    i = l->size - i;
    switch (f->size) {
case 1:
        if (i > 18) {
            goto numlit_overflow;
        }
        val = cb_get_long_long (src);
        if (f->pic->have_sign) {
            if (val < -128LL ||
                val > 127LL) {
                goto numlit_overflow;
            }
        }
        } else {
            if (val > 255LL) {
                goto numlit_overflow;
            }
        }
    }
    break;
}

```



```
case 2:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -32768LL ||
            val > 32767LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 65535LL) {
            goto numlit_overflow;
        }
    }
    break;
case 3:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -8388608LL ||
            val > 8388607LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 16777215LL) {
            goto numlit_overflow;
        }
    }
    break;
case 4:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -2147483648LL ||
            val > 2147483647LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 4294967295LL) {
            goto numlit_overflow;
        }
    }
    break;
case 5:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -549755813888LL ||
            val > 549755813887LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 1099511627775LL) {
            goto numlit_overflow;
        }
    }
}
```

```

    }
    break;
case 6:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -140737488355328LL ||
            val > 140737488355327LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 281474976710655LL) {
            goto numlit_overflow;
        }
    }
    break;
case 7:
    if (i > 18) {
        goto numlit_overflow;
    }
    val = cb_get_long_long (src);
    if (f->pic->have_sign) {
        if (val < -36028797018963968LL ||
            val > 36028797018963967LL) {
            goto numlit_overflow;
        }
    } else {
        if (val > 72057594037927935LL) {
            goto numlit_overflow;
        }
    }
    break;
default:
    if (f->pic->have_sign) {
        if (i < 19) {
            break;
        }
        if (i > 19) {
            goto numlit_overflow;
        }
        if (memcmp (p, "92233720368547758
07", 19) > 0) {
            goto numlit_overflow;
        }
    } else {
        if (i < 20) {
            break;
        }
        if (i > 20) {
            goto numlit_overflow;
        }
        if (memcmp (p, "18446744073709551
615", 20) > 0) {
            goto numlit_overflow;
        }
    }
    break;
}
return 0;

```

```

    }
    if (least_significant < -f->pic->scale) {
        goto size_overflow;
    }
    if (f->pic->scale > 0) {
        size = f->pic->digits - f->pic->scale;
    } else {
        size = f->pic->digits;
    }
    if (most_significant >= size) {
        goto size_overflow;
    }
} else {
    /* Alphanumeric literal */

    /* value check */
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_ALPHABETIC:
        for (i = 0; i < l->size; i++) {
            if (!isalpha (l->data[i]) && !isspace (l-
>data[i])) {
                goto value_mismatch;
            }
        }
        break;
    case CB_CATEGORY_NUMERIC:
        goto expect_numeric;
    case CB_CATEGORY_NUMERIC_EDITED:
        if (!is_value) {
            goto expect_numeric;
        }

        /* TODO: validate the value */
        break;
    default:
        break;
    }

    /* size check */
    size = cb_field_size (dst);
    if (size >= 0 && (int)l->size > size) {
        goto size_overflow;
    }
}
break;
case CB_TAG_FIELD:
case CB_TAG_REFERENCE:
    if (CB_REFERENCE_P(src) &&
        CB_ALPHABET_NAME_P(CB_REFERENCE(src)->value)) {
        break;
    }
    if (CB_REFERENCE_P(src) &&
        CB_FILE_P(CB_REFERENCE(src)->value)) {
        goto invalid;
    }
    size = cb_field_size (src);
    if (size < 0) {
        size = cb_field (src)->size;
    }
    /* non-elementary move */
    if (cb_field (src)->children || cb_field (dst)->children) {
        if (size > cb_field (dst)->size) {

```

```

        goto size_overflow_1;
    }
    break;
}

/* elementary move */
switch (CB_TREE_CATEGORY (src)) {
case CB_CATEGORY_ALPHANUMERIC:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_NUMERIC:
    case CB_CATEGORY_NUMERIC_EDITED:
        if (size > cb_field (dst)->pic->digits) {
            goto size_overflow_2;
        }
        break;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        if (size >
            count_pic_alphanumeric_edited (cb_field (dst)
)) {
            goto size_overflow_1;
        }
        break;
    default:
        if (size > cb_field (dst)->size) {
            goto size_overflow_1;
        }
        break;
    }
    break;
case CB_CATEGORY_ALPHABETIC:
case CB_CATEGORY_ALPHANUMERIC_EDITED:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_NUMERIC:
    case CB_CATEGORY_NUMERIC_EDITED:
        goto invalid;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        if (size >
            count_pic_alphanumeric_edited(cb_field (dst))
) {
            goto size_overflow_1;
        }
        break;
    default:
        if (size > cb_field (dst)->size) {
            goto size_overflow_1;
        }
        break;
    }
    break;
case CB_CATEGORY_NUMERIC:
case CB_CATEGORY_NUMERIC_EDITED:
    switch (CB_TREE_CATEGORY (dst)) {
    case CB_CATEGORY_ALPHABETIC:
        goto invalid;
    case CB_CATEGORY_ALPHANUMERIC_EDITED:
        is_numeric_edited = 1;
        /* Drop through */
    case CB_CATEGORY_ALPHANUMERIC:
        if (is_numeric_edited) {
            dst_size_mod = count_pic_alphanumeric_edi
ted (cb_field (dst));
        } else {

```

```

        dst_size_mod = cb_field (dst)->size;
    }
    if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC
        && cb_field (src)->pic->scale > 0) {
        if (cb_move_noninteger_to_alphanumeric ==
CB_ERROR) {
            goto invalid;
        }
        cb_warning_x (loc, _("Move non-integer to
alphanumeric"));
        break;
    }
    if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC
        && cb_field (src)->pic->digits > dst_size_mod
) {
        goto size_overflow_2;
    }
    if (CB_TREE_CATEGORY (src) ==
CB_CATEGORY_NUMERIC_EDITED
        && cb_field (src)->size > dst_size_mod) {
        goto size_overflow_1;
    }
    break;
default:
    src_scale_mod = cb_field (src)->pic->scale < 0 ?
        0 : cb_field (src)->pic->scale;
    dst_scale_mod = cb_field (dst)->pic->scale < 0 ?
        0 : cb_field (dst)->pic->scale;
    if (cb_field (src)->pic->digits - src_scale_mod >
        cb_field (dst)->pic->digits - dst_scale_mod
||
        src_scale_mod > dst_scale_mod) {
        goto size_overflow_2;
    }
    break;
}
break;
default:
    cb_error_x (loc, _("Invalid source for MOVE"));
    return -1;
}
break;
case CB_TAG_INTEGER:
case CB_TAG_BINARY_OP:
case CB_TAG_INTRINSIC:
    /* TODO: check this */
    break;
default:
    fprintf (stderr, "Invalid tree tag %d\n", CB_TREE_TAG (src));
    ABORT ();
}
return 0;

invalid:
    if (is_value) {
        cb_error_x (loc, _("Invalid VALUE clause"));
    } else {
        cb_error_x (loc, _("Invalid MOVE statement"));
    }
}

```

```
        return -1;

numlit_overflow:
    if (is_value) {
        cb_error_x (loc, _("Invalid VALUE clause - literal exceeds data s
ize"));
        return -1;
    }
    if (cb_warn_constant) {
        cb_warning_x (loc, _("Numeric literal exceeds data size"));
    }
    return 0;

expect_numeric:
    return move_error (src, dst, is_value, cb_warn_strict_typing, 0,
        _("Numeric value is expected"));

expect_alphanumeric:
    return move_error (src, dst, is_value, cb_warn_strict_typing, 0,
        _("Alphanumeric value is expected"));

value_mismatch:
    return move_error (src, dst, is_value, cb_warn_constant, 0,
        _("Value does not fit the picture string"));

size_overflow:
    return move_error (src, dst, is_value, cb_warn_constant, 0,
        _("Value size exceeds data size"));

size_overflow_1:
    return move_error (src, dst, is_value, cb_warn_truncate, 1,
        _("Sending field larger than receiving field"));

size_overflow_2:
    return move_error (src, dst, is_value, cb_warn_truncate, 1,
        _("Some digits may be truncated"));
}
```

### 7.17.3 Variable Documentation

#### 7.17.3.1 size.t sending\_id = 0

Definition at line 74 of file typeck.c.

#### 7.17.3.2 size.t suppress\_warn = 0

Definition at line 75 of file typeck.c.

## 7.18 config.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define `COB_EXEEXT` ""
- #define `COB_EXPORT_DYN` "-WI,--export-dynamic"
- #define `COB_EXTRA_FLAGS` ""
- #define `COB_HAS_INLINE` 1
- #define `COB_LI_IS_LL` 1
- #define `COB_PIC_FLAGS` "-fPIC -DPIC"
- #define `COB_SHARED_OPT` "-shared"
- #define `COB_STRFTIME` 1
- #define `COB_STRIP_CMD` "strip --strip-unneeded"
- #define `ENABLE_NLS` 1
- #define `HAVE_ALLOCA` 1
- #define `HAVE_ALLOCA_H` 1
- #define `HAVE_ATTRIBUTE_ALIGNED` 1
- #define `HAVE_COLOR_SET` 1
- #define `HAVE_DCGETTEXT` 1
- #define `HAVE_DLFCN_H` 1
- #define `HAVE_FCNTL` 1
- #define `HAVE_FCNTL_H` 1
- #define `HAVE_GETOPT_H` 1
- #define `HAVE_GETTEXT` 1
- #define `HAVE_GETTIMEOFDAY` 1
- #define `HAVE_GMP_H` 1
- #define `HAVE_ICONV` 1
- #define `HAVE_INTTYPES_H` 1
- #define `HAVE_LANGINFO_CODESET` 1
- #define `HAVE_LIBGMP` 1
- #define `HAVE_LIBNCURSES` 1
- #define `HAVE_LOCALE_H` 1
- #define `HAVE_MALLOC_H` 1
- #define `HAVE_MEMMOVE` 1
- #define `HAVE_MEMORY_H` 1
- #define `HAVE_MEMSET` 1
- #define `HAVE_NCURSES_H` 1
- #define `HAVE_PSIGN_OPT` 1
- #define `HAVE_SETLOCALE` 1

- #define [HAVE\\_SIGNAL\\_H](#) 1
- #define [HAVE\\_STDDEF\\_H](#) 1
- #define [HAVE\\_STDINT\\_H](#) 1
- #define [HAVE\\_STDLIB\\_H](#) 1
- #define [HAVE\\_STRCASECMP](#) 1
- #define [HAVE\\_STRCHR](#) 1
- #define [HAVE\\_STRDUP](#) 1
- #define [HAVE\\_STRERROR](#) 1
- #define [HAVE\\_STRINGS\\_H](#) 1
- #define [HAVE\\_STRING\\_H](#) 1
- #define [HAVE\\_STRRCHR](#) 1
- #define [HAVE\\_STRSTR](#) 1
- #define [HAVE\\_STRTOL](#) 1
- #define [HAVE\\_SYS\\_STAT\\_H](#) 1
- #define [HAVE\\_SYS\\_TIME\\_H](#) 1
- #define [HAVE\\_SYS\\_TYPES\\_H](#) 1
- #define [HAVE\\_TIMEZONE](#) 1
- #define [HAVE\\_UNISTD\\_H](#) 1
- #define [HAVE\\_VPRINTF](#) 1
- #define [HAVE\\_WCHAR\\_H](#) 1
- #define [ICONV\\_CONST](#)
- #define [PACKAGE](#) "open-cobol"
- #define [PACKAGE\\_BUGREPORT](#) "open-cobol-list@lists.sourceforge.net"
- #define [PACKAGE\\_NAME](#) "OpenCOBOL"
- #define [PACKAGE\\_STRING](#) "OpenCOBOL 1.1"
- #define [PACKAGE\\_TARNAME](#) "open-cobol"
- #define [PACKAGE\\_VERSION](#) "1.1"
- #define [PATCH\\_LEVEL](#) 0
- #define [STDC\\_HEADERS](#) 1
- #define [USE\\_DB41](#) 1
- #define [USE\\_LIBDL](#) 1
- #define [VERSION](#) "1.1"
- #define [WITH\\_DB](#) 3
- #define [WITH\\_VARSEQ](#) 0
- #define [\\_\\_USE\\_STRING\\_INLINES](#) 1

### 7.18.1 Define Documentation

#### 7.18.1.1 #define [\\_\\_USE\\_STRING\\_INLINES](#) 1

Definition at line 348 of file config.h.

#### 7.18.1.2 #define [COB\\_EXEEXT](#) ""

Definition at line 5 of file config.h.



7.18.1.3 `#define COB_EXPORT_DYN "-Wl,-export-dynamic"`

Definition at line 11 of file config.h.

7.18.1.4 `#define COB_EXTRA_FLAGS ""`

Definition at line 14 of file config.h.

7.18.1.5 `#define COB_HAS_INLINE 1`

Definition at line 17 of file config.h.

7.18.1.6 `#define COB_LI_IS_LL 1`

Definition at line 20 of file config.h.

7.18.1.7 `#define COB_PIC_FLAGS "-fPIC -DPIC"`

Definition at line 29 of file config.h.

7.18.1.8 `#define COB_SHARED_OPT "-shared"`

Definition at line 32 of file config.h.

7.18.1.9 `#define COB_STRFTIME 1`

Definition at line 35 of file config.h.

7.18.1.10 `#define COB_STRIP_CMD "strip --strip-unneeded"`

Definition at line 38 of file config.h.

7.18.1.11 `#define ENABLE_NLS 1`

Definition at line 50 of file config.h.

7.18.1.12 `#define HAVE_ALLOCA 1`

Definition at line 53 of file config.h.

7.18.1.13 `#define HAVE_ALLOCA_H 1`

Definition at line 57 of file config.h.

7.18.1.14 `#define HAVE_ATTRIBUTE_ALIGNED 1`

Definition at line 60 of file config.h.

7.18.1.15 `#define HAVE_COLOR_SET 1`

Definition at line 63 of file config.h.

7.18.1.16 `#define HAVE_DCGETTEXT 1`

Definition at line 100 of file config.h.

7.18.1.17 `#define HAVE_DLFCN_H 1`

Definition at line 106 of file config.h.

7.18.1.18 `#define HAVE_FCNTL 1`

Definition at line 112 of file config.h.

7.18.1.19 `#define HAVE_FCNTL_H 1`

Definition at line 115 of file config.h.

7.18.1.20 `#define HAVE_GETOPT_H 1`

Definition at line 118 of file config.h.

7.18.1.21 `#define HAVE_GETTEXT 1`

Definition at line 121 of file config.h.

7.18.1.22 `#define HAVE_GETTIMEOFDAY 1`

Definition at line 124 of file config.h.

7.18.1.23 `#define HAVE_GMP_H 1`

Definition at line 127 of file config.h.

7.18.1.24 `#define HAVE_ICONV 1`

Definition at line 130 of file config.h.

7.18.1.25 `#define HAVE_INTTYPES_H 1`

Definition at line 133 of file config.h.

7.18.1.26 `#define HAVE_LANGINFO_CODESET 1`

Definition at line 142 of file config.h.

7.18.1.27 `#define HAVE_LIBGMP 1`

Definition at line 151 of file config.h.

7.18.1.28 `#define HAVE_LIBNCURSES 1`

Definition at line 160 of file config.h.

7.18.1.29 `#define HAVE_LOCALE_H 1`

Definition at line 175 of file config.h.

7.18.1.30 `#define HAVE_MALLOC_H 1`

Definition at line 181 of file config.h.

7.18.1.31 `#define HAVE_MEMMOVE 1`

Definition at line 184 of file config.h.

7.18.1.32 `#define HAVE_MEMORY_H 1`

Definition at line 187 of file config.h.

7.18.1.33 `#define HAVE_MEMSET 1`

Definition at line 190 of file config.h.

7.18.1.34 `#define HAVE_NCURSES_H 1`

Definition at line 193 of file config.h.

7.18.1.35 `#define HAVE_PSIGN_OPT 1`

Definition at line 202 of file config.h.

7.18.1.36 `#define HAVE_SETLOCALE 1`

Definition at line 205 of file config.h.

7.18.1.37 `#define HAVE_SIGNAL_H 1`

Definition at line 208 of file config.h.

7.18.1.38 `#define HAVE_STDDEF_H 1`

Definition at line 211 of file config.h.

7.18.1.39 `#define HAVE_STDINT_H 1`

Definition at line 214 of file config.h.

7.18.1.40 `#define HAVE_STDLIB_H 1`

Definition at line 217 of file config.h.

7.18.1.41 `#define HAVE_STRCASECMP 1`

Definition at line 220 of file config.h.

7.18.1.42 `#define HAVE_STRCHR 1`

Definition at line 223 of file config.h.

7.18.1.43 `#define HAVE_STRDUP 1`

Definition at line 226 of file config.h.

7.18.1.44 `#define HAVE_STRERROR 1`

Definition at line 229 of file config.h.

7.18.1.45 `#define HAVE_STRING_H 1`

Definition at line 235 of file config.h.

7.18.1.46 `#define HAVE_STRINGS_H 1`

Definition at line 232 of file config.h.

7.18.1.47 `#define HAVE_STRCHR 1`

Definition at line 238 of file config.h.

7.18.1.48 `#define HAVE_STRSTR 1`

Definition at line 241 of file config.h.

7.18.1.49 `#define HAVE_STRTOL 1`

Definition at line 244 of file config.h.

7.18.1.50 `#define HAVE_SYS_STAT_H 1`

Definition at line 247 of file config.h.

7.18.1.51 `#define HAVE_SYS_TIME_H 1`

Definition at line 250 of file config.h.

7.18.1.52 `#define HAVE_SYS_TYPES_H 1`

Definition at line 253 of file config.h.

7.18.1.53 `#define HAVE_TIMEZONE 1`

Definition at line 256 of file config.h.

7.18.1.54 `#define HAVE_UNISTD_H 1`

Definition at line 259 of file config.h.

7.18.1.55 `#define HAVE_VPRINTF 1`

Definition at line 265 of file config.h.

7.18.1.56 `#define HAVE_WCHAR_H 1`

Definition at line 268 of file config.h.

7.18.1.57 `#define ICONV_CONST`

Definition at line 271 of file config.h.

7.18.1.58 `#define PACKAGE "open-cobol"`

Definition at line 274 of file config.h.

7.18.1.59 `#define PACKAGE_BUGREPORT "open-cobol-list@lists.sourceforge.net"`

Definition at line 277 of file config.h.

7.18.1.60 `#define PACKAGE_NAME "OpenCOBOL"`

Definition at line 280 of file config.h.

7.18.1.61 `#define PACKAGE_STRING "OpenCOBOL 1.1"`

Definition at line 283 of file config.h.

7.18.1.62 `#define PACKAGE_TARNAME "open-cobol"`

Definition at line 286 of file config.h.

7.18.1.63 `#define PACKAGE_VERSION "1.1"`

Definition at line 289 of file config.h.

7.18.1.64 `#define PATCH_LEVEL 0`

Definition at line 292 of file config.h.

7.18.1.65 `#define STDC_HEADERS 1`

Definition at line 303 of file config.h.

7.18.1.66 `#define USE_DB41 1`

Definition at line 309 of file config.h.

7.18.1.67 `#define USE_LIBDL 1`

Definition at line 312 of file config.h.

7.18.1.68 `#define VERSION "1.1"`

Definition at line 315 of file config.h.

7.18.1.69 `#define WITH_DB 3`

Definition at line 321 of file config.h.

7.18.1.70 `#define WITH_VARSEQ 0`

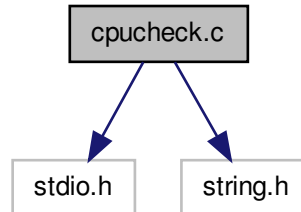
Definition at line 333 of file config.h.

## 7.19 cpucheck.c File Reference

```
#include <stdio.h>
```

```
#include <string.h>
```

Include dependency graph for cpucheck.c:



## Functions

- `int main ()`

### 7.19.1 Function Documentation

#### 7.19.1.1 `int main ( )`

Definition at line 5 of file cpucheck.c.

```

{
#ifdef __i386__
#if ( __GNUC__ == 3 && __GNUC_MINOR__ > 0 ) || __GNUC__ > 3
#if __GNUC__ > 3 || __GNUC_MINOR__ > 3
    char *ctune = "-mtune=";
#else
    char *ctune = "-mcpu=";
#endif
#endif
char vendor_string[16];
int eax, ebx, edx, ecx;
int i, hv;
int family, model, stepping;

__asm__ (".byte 0x0f,0xa2"
: "=a" (hv), "=b" (ebx), "=d" (edx), "=c" (ecx) : "0" (0));

*(int *) (vendor_string + 0) = ebx;
*(int *) (vendor_string + 4) = edx;
*(int *) (vendor_string + 8) = ecx;
vendor_string[12] = 0;
i = 1;

__asm__ (".byte 0x0f,0xa2"
: "=a" (eax), "=b" (ebx), "=d" (edx), "=c" (ecx) : "0" (i));

family = (eax >> 8) & 0xf;

```



```

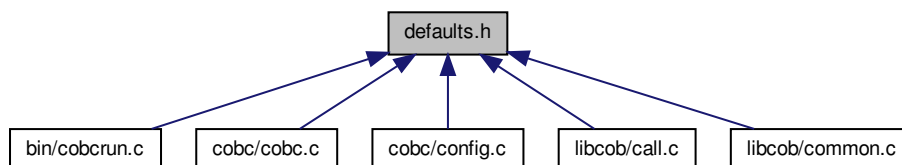
model = (eax >> 4) & 0xf;
stepping = eax & 0xf;
if (family == 0xf) {
    /* "extended" mode. */
    family += (eax >> 20) & 0xff;
    model += (eax >> 12) & 0xf0;
}

if (strcmp (vendor_string, "GenuineIntel") == 0) {
    if (family == 5) {
        printf ("-march=i686 %spentium", ctune);
    } else if (family == 6) {
        if (model <= 2) {
            printf ("-march=i686 %spentiumpro", ctune);
        } else if (model >= 3 && model <= 6) {
            printf ("-march=i686 %spentium2", ctune);
        } else if (model <= 11) {
            printf ("-march=i686 %spentium3", ctune);
        } else {
            printf ("-march=i686 %spentium4", ctune);
        }
    }
    else if (family == 15) {
        printf ("-march=i686 %spentium4", ctune);
    }
} else if (strcmp (vendor_string, "AuthenticAMD") == 0) {
    if (family == 6) {
        printf ("-march=i686 %sathlon", ctune);
    }
}
#endif
#endif
return 0;
}

```

## 7.20 defaults.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define `COB_CC` "gcc"

- `#define COB_CFLAGS "-I/usr/local/include "`
- `#define COB_LDFLAGS ""`
- `#define COB_LIBS "-L/usr/local/lib -lcob -lm -lgmp -Incurses -ldb -ldl"`
- `#define COB_CONFIG_DIR "/usr/local/share/open-cobol/config"`
- `#define COB_COPY_DIR "/usr/local/share/open-cobol/copy"`
- `#define COB_LIBRARY_PATH "/usr/local/lib/open-cobol"`
- `#define COB_MODULE_EXT "so"`
- `#define LOCALEDIR "/usr/local/share/locale"`

## 7.20.1 Define Documentation

### 7.20.1.1 `#define COB_CC "gcc"`

Definition at line 2 of file defaults.h.

### 7.20.1.2 `#define COB_CFLAGS "-I/usr/local/include "`

Definition at line 3 of file defaults.h.

### 7.20.1.3 `#define COB_CONFIG_DIR "/usr/local/share/open-cobol/config"`

Definition at line 6 of file defaults.h.

### 7.20.1.4 `#define COB_COPY_DIR "/usr/local/share/open-cobol/copy"`

Definition at line 7 of file defaults.h.

### 7.20.1.5 `#define COB_LDFLAGS ""`

Definition at line 4 of file defaults.h.

### 7.20.1.6 `#define COB_LIBRARY_PATH "/usr/local/lib/open-cobol"`

Definition at line 8 of file defaults.h.

### 7.20.1.7 `#define COB_LIBS "-L/usr/local/lib -lcob -lm -lgmp -Incurses -ldb -ldl"`

Definition at line 5 of file defaults.h.

### 7.20.1.8 `#define COB_MODULE_EXT "so"`

Definition at line 9 of file defaults.h.

7.20.1.9 `#define LOCALEDIR "/usr/local/share/locale"`

Definition at line 10 of file defaults.h.

## 7.21 lib/dummysmac.c File Reference

### Functions

- void [dummysmacfix](#) (void)

### 7.21.1 Function Documentation

7.21.1.1 void [dummysmacfix](#) ( void )

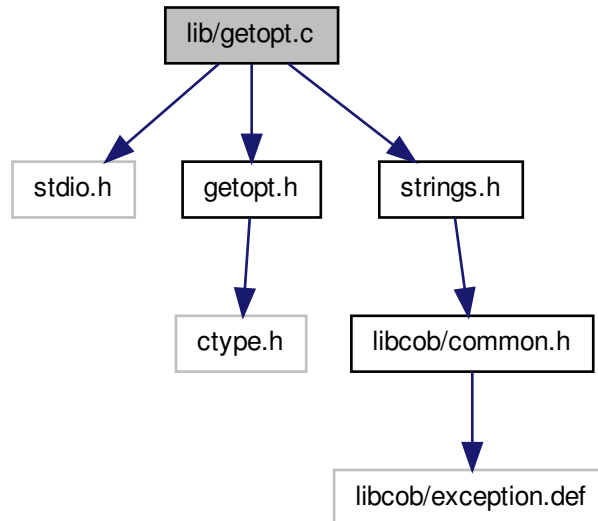
Definition at line 3 of file dummysmac.c.

```
{  
}
```

## 7.22 lib/getopt.c File Reference

```
#include <stdio.h>  
#include "getopt.h"  
#include <strings.h>
```

Include dependency graph for getopt.c:



## Defines

- `#define _NO_PROTO`
- `#define const`
- `#define GETOPT_INTERFACE_VERSION 2`
- `#define _(msgid) (msgid)`
- `#define attribute_hidden`
- `#define SWAP_FLAGS(ch1, ch2)`
- `#define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')`

## Enumerations

- `enum { REQUIRE_ORDER, PERMUTE, RETURN_IN_ORDER }`

## Functions

- `char * getopt ()`
- `int _getopt_internal (int argc, char *const *argv, const char *optstring, const struct option *longopts, int *longind, int long_only)`
- `int getopt (int argc, char *const *argv, const char *optstring)`

## Variables

- char \* `optarg`
- int `optind` = 1
- int `__getopt_initialized` `attribute_hidden`
- int `opterr` = 1
- int `optopt` = '?'

### 7.22.1 Define Documentation

#### 7.22.1.1 `#define _( msgid ) (msgid)`

Definition at line 89 of file getopt.c.

#### 7.22.1.2 `#define _NO_PROTO`

Definition at line 27 of file getopt.c.

#### 7.22.1.3 `#define attribute_hidden`

Definition at line 97 of file getopt.c.

#### 7.22.1.4 `#define const`

Definition at line 38 of file getopt.c.

#### 7.22.1.5 `#define GETOPT_INTERFACE_VERSION 2`

Definition at line 52 of file getopt.c.

#### 7.22.1.6 `#define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')`

#### 7.22.1.7 `#define SWAP_FLAGS( ch1, ch2 )`

Definition at line 291 of file getopt.c.

### 7.22.2 Enumeration Type Documentation

#### 7.22.2.1 anonymous enum

**Enumerator:**

***REQUIRE\_ORDER***

***PERMUTE***

**RETURN\_IN\_ORDER**

Definition at line 194 of file getopt.c.

```
{
    REQUIRE_ORDER, PERMUTE, RETURN_IN_ORDER
} ordering;
```

**7.22.3 Function Documentation****7.22.3.1 int \_getopt\_internal ( int argc, char \*const \* argv, const char \* optstring, const struct option \* longopts, int \* longind, int long\_only )**

Definition at line 515 of file getopt.c.

```
{
    int print_errors = opterr;
    if (optstring[0] == ':')
        print_errors = 0;

    if (argc < 1)
        return -1;

    optarg = NULL;

    if (optind == 0 || !__getopt_initialized)
    {
        if (optind == 0)
            optind = 1; /* Don't scan ARGV[0], the program name. */
        optstring = _getopt_initialize (argc, argv, optstring);
        __getopt_initialized = 1;
    }

    /* Test whether ARGV[optind] points to a non-option argument.
       Either it does not have option syntax, or there is an environment flag
       from the shell indicating it is not an option. The later information
       is only used when the used in the GNU libc. */
    #if defined _LIBC && defined USE_NONOPTION_FLAGS
    # define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0' \
        || (optind < nonoption_flags_len \
            && __getopt_nonoption_flags[optind] == '1'))
    #else
    # define NONOPTION_P (argv[optind][0] != '-' || argv[optind][1] == '\0')
    #endif

    if (nextchar == NULL || *nextchar == '\0')
    {
        /* Advance to the next ARGV-element. */

        /* Give FIRST_NONOPT & LAST_NONOPT rational values if OPTIND has been
           moved back by the user (who may also have changed the arguments). */
        if (last_nonopt > optind)
            last_nonopt = optind;
        if (first_nonopt > optind)
            first_nonopt = optind;

        if (ordering == PERMUTE)
        {
```

```
/* If we have just processed some options following some non-options,
   exchange them so that the options come first. */

if (first_nonopt != last_nonopt && last_nonopt != optind)
    exchange ((char **) argv);
else if (last_nonopt != optind)
    first_nonopt = optind;

/* Skip any additional non-options
   and extend the range of non-options previously skipped. */

while (optind < argc && NONOPTION_P)
    optind++;
last_nonopt = optind;
}

/* The special ARGV-element '--' means premature end of options.
   Skip it like a null option,
   then exchange with previous non-options as if it were an option,
   then skip everything else like a non-option. */

if (optind != argc && !strcmp (argv[optind], "--"))
{
    optind++;

    if (first_nonopt != last_nonopt && last_nonopt != optind)
        exchange ((char **) argv);
    else if (first_nonopt == last_nonopt)
        first_nonopt = optind;
    last_nonopt = argc;

    optind = argc;
}

/* If we have done all the ARGV-elements, stop the scan
   and back over any non-options that we skipped and permuted. */

if (optind == argc)
{
    /* Set the next-arg-index to point at the non-options
       that we previously skipped, so the caller will digest them. */
    if (first_nonopt != last_nonopt)
        optind = first_nonopt;
    return -1;
}

/* If we have come to a non-option and did not permute it,
   either stop the scan or describe it to the caller and pass it by. */

if (NONOPTION_P)
{
    if (ordering == REQUIRE_ORDER)
        return -1;
    optarg = argv[optind++];
    return 1;
}

/* We have found another option-ARGV-element.
   Skip the initial punctuation. */

nextchar = (argv[optind] + 1
            + (longopts != NULL && argv[optind][1] == '-'));
```

```

    }

    /* Decode the current option-ARGV-element. */

    /* Check whether the ARGV-element is a long option.

       If long_only and the ARGV-element has the form "-f", where f is
       a valid short option, don't consider it an abbreviated form of
       a long option that starts with f. Otherwise there would be no
       way to give the -f short option.

       On the other hand, if there's a long option "fubar" and
       the ARGV-element is "-fu", do consider that an abbreviation of
       the long option, just like "--fu", and not "-f" with arg "u".

       This distinction seems to be the most useful approach. */

if (longopts != NULL
    && (argv[optind][1] == '-'
        || (long_only && (argv[optind][2] || !my_index (optstring, argv[optind]
            [1])))))
{
    char *nameend;
    const struct option *p;
    const struct option *pfound = NULL;
    int exact = 0;
    int ambig = 0;
    int indfound = -1;
    int option_index;

    for (nameend = nextchar; *nameend && *nameend != '='; nameend++)
        /* Do nothing. */ ;

    /* Test all long options for either exact match
       or abbreviated matches. */
    for (p = longopts, option_index = 0; p->name; p++, option_index++)
        if (!strncmp (p->name, nextchar, nameend - nextchar))
            {
                if ((unsigned int) (nameend - nextchar)
                    == (unsigned int) strlen (p->name))
                    {
                        /* Exact match found. */
                        pfound = p;
                        indfound = option_index;
                        exact = 1;
                        break;
                    }
                else if (pfound == NULL)
                    {
                        /* First nonexact match found. */
                        pfound = p;
                        indfound = option_index;
                    }
                else if (long_only
                        || pfound->has_arg != p->has_arg
                        || pfound->flag != p->flag
                        || pfound->val != p->val)
                    /* Second or later nonexact match found. */
                        ambig = 1;
            }
        }

    if (ambig && !exact)

```



```

        {
            if (print_errors)
            {
#ifdef defined _LIBC && defined USE_IN_LIBIO
                char *buf;

                if (__asprintf (&buf, _("%s: option '%s' is ambiguous\n"),
                                argv[0], argv[optind]) >= 0)
                {
                    if (_IO_fwide (stderr, 0) > 0)
                        __fwprintf (stderr, L"%s", buf);
                    else
                        fputs (buf, stderr);

                    free (buf);
                }
#else
                fprintf (stderr, _("%s: option '%s' is ambiguous\n"),
                        argv[0], argv[optind]);
#endif
            }
            nextchar += strlen (nextchar);
            optind++;
            optopt = 0;
            return '?';
        }

    if (pfound != NULL)
    {
        option_index = indfound;
        optind++;
        if (*nameend)
        {
            /* Don't test has_arg with >, because some C compilers don't
               allow it to be used on enums. */
            if (pfound->has_arg)
                optarg = nameend + 1;
            else
            {
                if (print_errors)
                {
#ifdef defined _LIBC && defined USE_IN_LIBIO
                    char *buf;
                    int n;
#endif

                    if (argv[optind - 1][1] == '-')
                    {
                        /* --option */
#ifdef defined _LIBC && defined USE_IN_LIBIO
                            n = __asprintf (&buf, _("\
%s: option '--%s' doesn't allow an argument\n"),
                                            argv[0], pfound->name);
#else
                            fprintf (stderr, _("\
%s: option '--%s' doesn't allow an argument\n"),
                                    argv[0], pfound->name);
#endif
                        #endif
                    }
                    else
                    {

```

```

/* +option or -option */
#if defined _LIBC && defined USE_IN_LIBIO
    n = __asprintf (&buf, _("\
%s: option '%c%s' doesn't allow an argument\n"),
                    argv[0], argv[optind - 1][0],
                    pfound->name);
#else
    fprintf (stderr, _("\
%s: option '%c%s' doesn't allow an argument\n"),
            argv[0], argv[optind - 1][0], pfound->name);
#endif
    }

#if defined _LIBC && defined USE_IN_LIBIO
    if (n >= 0)
    {
        if (_IO_fwide (stderr, 0) > 0)
            __fwprintf (stderr, L"%s", buf);
        else
            fputs (buf, stderr);

        free (buf);
    }
#endif

    nextchar += strlen (nextchar);

    optopt = pfound->val;
    return '?';
}
else if (pfound->has_arg == 1)
{
    if (optind < argc)
        optarg = argv[optind++];
    else
    {
        if (print_errors)
        {
#if defined _LIBC && defined USE_IN_LIBIO
            char *buf;

            if (__asprintf (&buf, _("\
%s: option '%s' requires an argument\n"),
                            argv[0], argv[optind - 1]) >= 0)
            {
                if (_IO_fwide (stderr, 0) > 0)
                    __fwprintf (stderr, L"%s", buf);
                else
                    fputs (buf, stderr);

                free (buf);
            }
#endif
        }
    }
}
else
    fprintf (stderr,
            _("%s: option '%s' requires an argument\n"),
            argv[0], argv[optind - 1]);
#endif
    }
    nextchar += strlen (nextchar);
    optopt = pfound->val;

```

```

        return optstring[0] == ':' ? ':' : '?';
    }
}
nextchar += strlen (nextchar);
if (longind != NULL)
    *longind = option_index;
if (pfound->flag)
    {
        *(pfound->flag) = pfound->val;
        return 0;
    }
return pfound->val;
}

/* Can't find it as a long option.  If this is not getopt_long_only,
or the option starts with '--' or is not a valid short
option, then it's an error.
Otherwise interpret it as a short option.  */
if (!long_only || argv[optind][1] == '-')
    || my_index (optstring, *nextchar) == NULL)
    {
        if (print_errors)
            {
#ifdef _LIBC && defined USE_IN_LIBIO
                char *buf;
                int n;
#endif

                if (argv[optind][1] == '-')
                    {
                        /* --option */
#ifdef _LIBC && defined USE_IN_LIBIO
                            n = __asprintf (&buf, _("%s: unrecognized option '--%s'\n"),
                                argv[0], nextchar);
                        #else
                            fprintf (stderr, _("%s: unrecognized option '--%s'\n"),
                                argv[0], nextchar);
                        #endif
                    }
                else
                    {
                        /* +option or -option */
#ifdef _LIBC && defined USE_IN_LIBIO
                            n = __asprintf (&buf, _("%s: unrecognized option '%c%s'\n"),
                                argv[0], argv[optind][0], nextchar);
                        #else
                            fprintf (stderr, _("%s: unrecognized option '%c%s'\n"),
                                argv[0], argv[optind][0], nextchar);
                        #endif
                    }
            }
        }

#ifdef _LIBC && defined USE_IN_LIBIO
        if (n >= 0)
            {
                if (_IO_fwide (stderr, 0) > 0)
                    __fwprintf (stderr, L"%s", buf);
                else
                    fputs (buf, stderr);

                free (buf);
            }
#endif
    }
}
#endif

```

```

    }
    nextchar = (char *) "";
    optind++;
    optopt = 0;
    return '?';
}

/* Look at and handle the next short option-character. */

{
    char c = *nextchar++;
    char *temp = my_index (optstring, c);

    /* Increment 'optind' when we start to process its last character. */
    if (*nextchar == '\0')
        ++optind;

    if (temp == NULL || c == ':')
    {
        if (print_errors)
        {
            #if defined _LIBC && defined USE_IN_LIBIO
                char *buf;
                int n;
            #endif

            if (posixly_correct)
            {
                /* 1003.2 specifies the format of this message. */
                #if defined _LIBC && defined USE_IN_LIBIO
                    n = __asprintf (&buf, _("%s: illegal option -- %c\n"),
                                   argv[0], c);
                #else
                    fprintf (stderr, _("%s: illegal option -- %c\n"), argv[0], c);
                #endif
            }
            else
            {
                #if defined _LIBC && defined USE_IN_LIBIO
                    n = __asprintf (&buf, _("%s: invalid option -- %c\n"),
                                   argv[0], c);
                #else
                    fprintf (stderr, _("%s: invalid option -- %c\n"), argv[0], c);
                #endif
            }
        }

        #if defined _LIBC && defined USE_IN_LIBIO
            if (n >= 0)
            {
                if (_IO_fwide (stderr, 0) > 0)
                    __fwprintf (stderr, L"%s", buf);
                else
                    fputs (buf, stderr);

                free (buf);
            }
        #endif
    }
    optopt = c;
    return '?';
}

```

```

/* Convenience. Treat POSIX -W foo same as long option --foo */
if (temp[0] == 'W' && temp[1] == ';')
{
    char *nameend;
    const struct option *p;
    const struct option *pfound = NULL;
    int exact = 0;
    int ambig = 0;
    int indfound = 0;
    int option_index;

    /* This is an option that requires an argument. */
    if (*nextchar != '\0')
    {
        optarg = nextchar;
        /* If we end this ARGV-element by taking the rest as an arg,
           we must advance to the next element now. */
        optind++;
    }
    else if (optind == argc)
    {
        if (print_errors)
        {
            /* 1003.2 specifies the format of this message. */
#ifdef defined _LIBC && defined USE_IN_LIBIO
            char *buf;

            if (__asprintf (&buf,
                           _("%s: option requires an argument -- %c\n"),
                           argv[0], c) >= 0)
            {
                if (_IO_fwide (stderr, 0) > 0)
                    __fwprintf (stderr, L"%s", buf);
                else
                    fputs (buf, stderr);

                free (buf);
            }
#endif
        }
        fprintf (stderr, _("%s: option requires an argument -- %c\n"),
                argv[0], c);
    }
#ifdef endif
    }
    optopt = c;
    if (optstring[0] == ':')
        c = ':';
    else
        c = '?';
    return c;
}
else
    /* We already incremented 'optind' once;
       increment it again when taking next ARGV-elt as argument. */
    optarg = argv[optind++];

/* optarg is now the argument, see if it's in the
   table of longopts. */

for (nextchar = nameend = optarg; *nameend && *nameend != '='; nameend++)

    /* Do nothing. */ ;

```

```

/* Test all long options for either exact match
or abbreviated matches. */
for (p = longopts, option_index = 0; p->name; p++, option_index++)
if (!strcmp (p->name, nextchar, nameend - nextchar))
{
    if ((unsigned int) (nameend - nextchar) == strlen (p->name))
    {
        /* Exact match found. */
        pfound = p;
        indfound = option_index;
        exact = 1;
        break;
    }
    else if (pfound == NULL)
    {
        /* First nonexact match found. */
        pfound = p;
        indfound = option_index;
    }
    else
        /* Second or later nonexact match found. */
        ambig = 1;
}
if (ambig && !exact)
{
    if (print_errors)
    {
#if defined _LIBC && defined USE_IN_LIBIO
        char *buf;

        if (__asprintf (&buf, _("%s: option '-W %s' is ambiguous\n"),
                        argv[0], argv[optind]) >= 0)
        {
            if (_IO_fwide (stderr, 0) > 0)
                __fwprintf (stderr, L"%s", buf);
            else
                fputs (buf, stderr);

            free (buf);
        }
#else
        fprintf (stderr, _("%s: option '-W %s' is ambiguous\n"),
                argv[0], argv[optind]);
#endif
    }
    nextchar += strlen (nextchar);
    optind++;
    return '?';
}
if (pfound != NULL)
{
    option_index = indfound;
    if (*nameend)
    {
        /* Don't test has_arg with >, because some C compilers don't
allow it to be used on enums. */
        if (pfound->has_arg)
            optarg = nameend + 1;
        else
        {
            if (print_errors)
            {

```

```

#if defined _LIBC && defined USE_IN_LIBIO
    char *buf;

    if (__asprintf (&buf, _("\
%s: option '-W %s' doesn't allow an argument\n"),
                    argv[0], pfound->name) >= 0)
    {
        if (_IO_fwide (stderr, 0) > 0)
            __fwprintf (stderr, L"%s", buf);
        else
            fputs (buf, stderr);

        free (buf);
    }
#else
    fprintf (stderr, _("\
%s: option '-W %s' doesn't allow an argument\n"),
            argv[0], pfound->name);
#endif

    }

    nextchar += strlen (nextchar);
    return '?';
}
else if (pfound->has_arg == 1)
{
    if (optind < argc)
        optarg = argv[optind++];
    else
    {
        if (print_errors)
        {
#if defined _LIBC && defined USE_IN_LIBIO
            char *buf;

            if (__asprintf (&buf, _("\
%s: option '%s' requires an argument\n"),
                            argv[0], argv[optind - 1]) >= 0)
            {
                if (_IO_fwide (stderr, 0) > 0)
                    __fwprintf (stderr, L"%s", buf);
                else
                    fputs (buf, stderr);

                free (buf);
            }
        }
        fprintf (stderr,
                _("\
%s: option '%s' requires an argument\n"),
                argv[0], argv[optind - 1]);
    }
    nextchar += strlen (nextchar);
    return optstring[0] == ':' ? ':' : '?';
}
}
nextchar += strlen (nextchar);
if (longind != NULL)
    *longind = option_index;
if (pfound->flag)
{

```

```

        *(pfound->flag) = pfound->val;
        return 0;
    }
    return pfound->val;
}
nextchar = NULL;
return 'W'; /* Let the application handle it. */
}
if (temp[1] == ':')
{
    if (temp[2] == ':')
    {
        /* This is an option that accepts an argument optionally. */
        if (*nextchar != '\0')
        {
            optarg = nextchar;
            optind++;
        }
        else
            optarg = NULL;
        nextchar = NULL;
    }
    else
    {
        /* This is an option that requires an argument. */
        if (*nextchar != '\0')
        {
            optarg = nextchar;
            /* If we end this ARGV-element by taking the rest as an arg,
             we must advance to the next element now. */
            optind++;
        }
        else if (optind == argc)
        {
            if (print_errors)
            {
                /* 1003.2 specifies the format of this message. */
#ifdef _LIBC && defined USE_IN_LIBIO
                char *buf;

                if (__asprintf (&buf, _("\
%s: option requires an argument -- %c\n"),
                                argv[0], c) >= 0)
                {
                    if (_IO_fwide (stderr, 0) > 0)
                        __fwprintf (stderr, L"%s", buf);
                    else
                        fputs (buf, stderr);

                    free (buf);
                }
#endif
            }
            fprintf (stderr,
                    _("%s: option requires an argument -- %c\n"),
                    argv[0], c);
#endif
        }
        optopt = c;
        if (optstring[0] == ':')
            c = ':';
        else
            c = '?';
    }
}

```



```
    }
    else
        /* We already incremented 'optind' once;
           increment it again when taking next ARGV-elt as argument. */
        optarg = argv[optind++];
        nextchar = NULL;
    }
}
return c;
}
}
```

### 7.22.3.2 char\* getenv ( )

### 7.22.3.3 int getopt ( int argc, char \*const \* argv, const char \* optstring )

Definition at line 1196 of file getopt.c.

```
{
    return _getopt_internal (argc, argv, optstring,
                             (const struct option *) 0,
                             (int *) 0,
                             0);
}
```

## 7.22.4 Variable Documentation

### 7.22.4.1 int \_\_getopt\_initialized attribute\_hidden

Definition at line 143 of file getopt.c.

### 7.22.4.2 char\* optarg

Definition at line 122 of file getopt.c.

### 7.22.4.3 int opterr = 1

Definition at line 157 of file getopt.c.

### 7.22.4.4 int optind = 1

Definition at line 137 of file getopt.c.

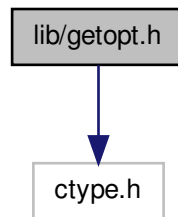
### 7.22.4.5 int optopt = '?'

Definition at line 163 of file getopt.c.

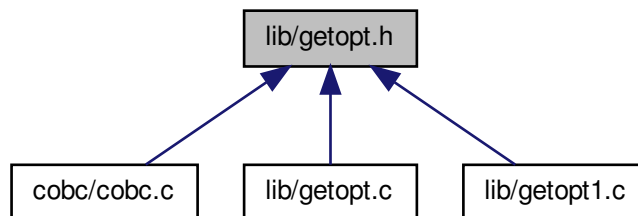
## 7.23 lib/getopt.h File Reference

```
#include <ctype.h>
```

Include dependency graph for getopt.h:



This graph shows which files directly or indirectly include this file:



### Classes

- struct [option](#)

### Defines

- #define [\\_GETOPT\\_H](#) 1
- #define [no\\_argument](#) 0
- #define [required\\_argument](#) 1

- #define [optional\\_argument](#) 2

## Functions

- int [getopt](#) ( )
- int [getopt\\_long](#) ( )
- int [getopt\\_long\\_only](#) ( )
- int [\\_getopt\\_internal](#) ( )

## Variables

- char \* [optarg](#)
- int [optind](#)
- int [opterr](#)
- int [optopt](#)

### 7.23.1 Define Documentation

#### 7.23.1.1 #define [\\_GETOPT\\_H](#) 1

Definition at line 23 of file getopt.h.

#### 7.23.1.2 #define [no\\_argument](#) 0

Definition at line 110 of file getopt.h.

#### 7.23.1.3 #define [optional\\_argument](#) 2

Definition at line 112 of file getopt.h.

#### 7.23.1.4 #define [required\\_argument](#) 1

Definition at line 111 of file getopt.h.

### 7.23.2 Function Documentation

#### 7.23.2.1 int [\\_getopt\\_internal](#) ( )

#### 7.23.2.2 int [getopt](#) ( )

#### 7.23.2.3 int [getopt\\_long](#) ( )

7.23.2.4 `int getopt_long_only ( )`

### 7.23.3 Variable Documentation

7.23.3.1 `char* optarg`

Definition at line 122 of file `getopt.c`.

7.23.3.2 `int opterr`

Definition at line 157 of file `getopt.c`.

7.23.3.3 `int optind`

Definition at line 137 of file `getopt.c`.

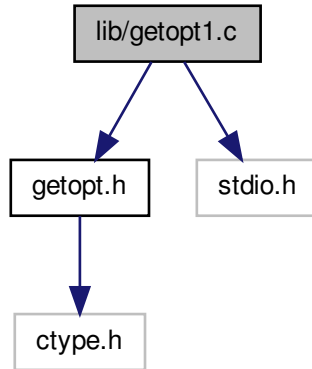
7.23.3.4 `int optopt`

Definition at line 163 of file `getopt.c`.

## 7.24 `lib/getopt1.c` File Reference

```
#include "getopt.h"  
#include <stdio.h>
```

Include dependency graph for getopt1.c:



## Defines

- #define [const](#)
- #define [GETOPT\\_INTERFACE\\_VERSION](#) 2
- #define [NULL](#) 0

## Functions

- int [getopt\\_long](#) (int argc, char \*const \*argv, const char \*options, const struct [option](#) \*long\_options, int \*opt\_index)
- int [getopt\\_long\\_only](#) (int argc, char \*const \*argv, const char \*options, const struct [option](#) \*long\_options, int \*opt\_index)

### 7.24.1 Define Documentation

#### 7.24.1.1 #define const

Definition at line 35 of file getopt1.c.

#### 7.24.1.2 #define GETOPT\_INTERFACE\_VERSION 2

Definition at line 49 of file getopt1.c.

### 7.24.1.3 #define NULL 0

Definition at line 67 of file getopt1.c.

## 7.24.2 Function Documentation

### 7.24.2.1 int getopt\_long ( int argc, char \*const \* argv, const char \* options, const struct option \* long\_options, int \* opt\_index )

Definition at line 71 of file getopt1.c.

```
{  
    return _getopt_internal (argc, argv, options, long_options, opt_index, 0);  
}
```

### 7.24.2.2 int getopt\_long\_only ( int argc, char \*const \* argv, const char \* options, const struct option \* long\_options, int \* opt\_index )

Definition at line 87 of file getopt1.c.

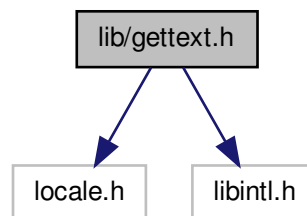
```
{  
    return _getopt_internal (argc, argv, options, long_options, opt_index, 1);  
}
```

## 7.25 lib/gettext.h File Reference

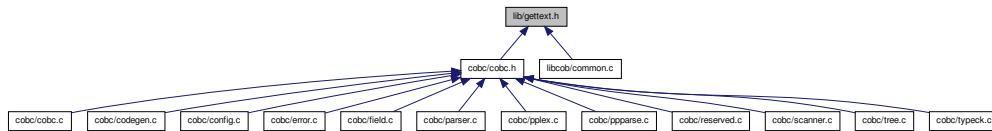
```
#include <locale.h>
```

```
#include <libintl.h>
```

Include dependency graph for gettext.h:



This graph shows which files directly or indirectly include this file:



## Defines

- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)
- #define [gettext](#)(Msgid) ((const char \*) (Msgid))
- #define [dgettext](#)(Domainname, Msgid) ((const char \*) (Msgid))
- #define [dcgettext](#)(Domainname, Msgid, Category) ((const char \*) (Msgid))
- #define [ngettext](#)(Msgid1, Msgid2, N) ((N) == 1 ? (const char \*) (Msgid1) : (const char \*) (Msgid2))
- #define [dngettext](#)(Domainname, Msgid1, Msgid2, N) ((N) == 1 ? (const char \*) (Msgid1) : (const char \*) (Msgid2))
- #define [dcngettext](#)(Domainname, Msgid1, Msgid2, N, Category) ((N) == 1 ? (const char \*) (Msgid1) : (const char \*) (Msgid2))
- #define [textdomain](#)(Domainname) ((const char \*) (Domainname))
- #define [bindtextdomain](#)(Domainname, Dirname) ((const char \*) (Dirname))
- #define [bind\\_textdomain\\_codeset](#)(Domainname, Codeset) ((const char \*) (Codeset))
- #define [gettext\\_noop](#)(String) String
- #define [\\_\(s\)](#) gettext(s)
- #define [N\\_\(s\)](#) gettext\_noop(s)

## 7.25.1 Define Documentation

7.25.1.1 `#define _( s ) gettext(s)`

7.25.1.2 `#define _( s ) gettext(s)`

7.25.1.3 `#define _( s ) gettext(s)`

7.25.1.4 `#define _( s ) gettext(s)`

7.25.1.5 `#define _( s ) gettext(s)`

Definition at line 60 of file gettext.h.

7.25.1.6 `#define _( s ) gettext(s)`

7.25.1.7 `#define _( s ) gettext(s)`

7.25.1.8 `#define _( s ) gettext(s)`

7.25.1.9 `#define bind_textdomain_codeset( Domainname, Codeset )((const char *) (Codeset))`

Definition at line 47 of file gettext.h.

7.25.1.10 `#define bindtextdomain( Domainname, Dirname )((const char *) (Dirname))`

Definition at line 46 of file gettext.h.

7.25.1.11 `#define dcgettext( Domainname, Msgid, Category )((const char *) (Msgid))`

Definition at line 38 of file gettext.h.

7.25.1.12 `#define dcngettext( Domainname, Msgid1, Msgid2, N, Category )((N) == 1 ?  
(const char *) (Msgid1) : (const char *) (Msgid2))`

Definition at line 43 of file gettext.h.

7.25.1.13 `#define dgettext( Domainname, Msgid )((const char *) (Msgid))`

Definition at line 37 of file gettext.h.

7.25.1.14 `#define dngettext( Domainname, Msgid1, Msgid2, N )((N) == 1 ? (const char *)  
(Msgid1) : (const char *) (Msgid2))`

Definition at line 41 of file gettext.h.



7.25.1.15 `#define gettext( Msgid )((const char *) (Msgid))`

Definition at line 36 of file gettext.h.

7.25.1.16 `#define gettext_noop( String ) String`

Definition at line 58 of file gettext.h.

7.25.1.17 `#define gettext_noop( String ) String`

7.25.1.18 `#define gettext_noop( String ) String`

7.25.1.19 `#define gettext_noop( String ) String`

7.25.1.20 `#define gettext_noop( String ) String`

7.25.1.21 `#define gettext_noop( String ) String`

7.25.1.22 `#define gettext_noop( String ) String`

7.25.1.23 `#define gettext_noop( String ) String`

7.25.1.24 `#define N_( s ) gettext_noop(s)`

7.25.1.25 `#define N_( s ) gettext_noop(s)`

7.25.1.26 `#define N_( s ) gettext_noop(s)`

7.25.1.27 `#define N_( s ) gettext_noop(s)`

7.25.1.28 `#define N_( s ) gettext_noop(s)`

Definition at line 61 of file gettext.h.

7.25.1.29 `#define N_( s ) gettext_noop(s)`

7.25.1.30 `#define N_( s ) gettext_noop(s)`

7.25.1.31 `#define N_( s ) gettext_noop(s)`

7.25.1.32 `#define ngettext( Msgid1, Msgid2, N )((N) == 1 ? (const char *) (Msgid1) : (const char *) (Msgid2))`

Definition at line 39 of file gettext.h.

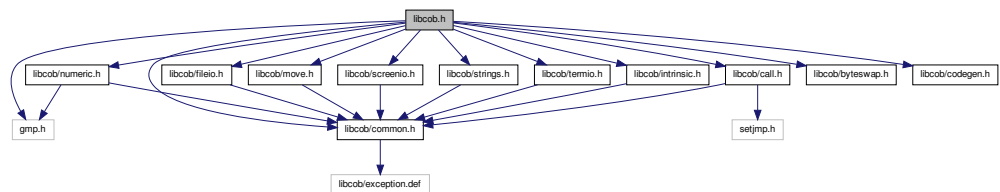
### 7.25.1.33 #define textdomain( *Domainname* )((const char \*) (Domainname))

Definition at line 45 of file gettext.h.

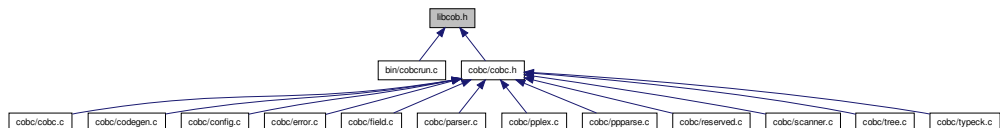
## 7.26 libcob.h File Reference

```
#include <gmp.h>
#include <libcob/byteswap.h>
#include <libcob/call.h>
#include <libcob/common.h>
#include <libcob/fileio.h>
#include <libcob/move.h>
#include <libcob/numeric.h>
#include <libcob/screenio.h>
#include <libcob/strings.h>
#include <libcob/termio.h>
#include <libcob/intrinsic.h>
#include <libcob/codegen.h>
```

Include dependency graph for libcob.h:

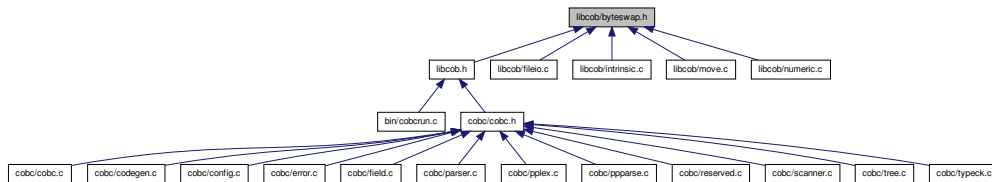


This graph shows which files directly or indirectly include this file:



## 7.27 libcob/byteswap.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define [COB\\_BSWAP\\_16\\_CONSTANT\(val\)](#)
- #define [COB\\_BSWAP\\_32\\_CONSTANT\(val\)](#)
- #define [COB\\_BSWAP\\_64\\_CONSTANT\(val\)](#)
- #define [COB\\_BSWAP\\_16\(val\)](#) (COB\_BSWAP\_16\_CONSTANT (val))
- #define [COB\\_BSWAP\\_32\(val\)](#) (COB\_BSWAP\_32\_CONSTANT (val))
- #define [COB\\_BSWAP\\_64\(val\)](#) (COB\_BSWAP\_64\_CONSTANT (val))

#### 7.27.1 Define Documentation

##### 7.27.1.1 #define COB\_BSWAP\_16( val ) (COB\_BSWAP\_16\_CONSTANT (val))

Definition at line 183 of file byteswap.h.

##### 7.27.1.2 #define COB\_BSWAP\_16\_CONSTANT( val )

###### Value:

```

((unsigned short) (
    (unsigned short) ((unsigned short) (val) >> 8) |
    (unsigned short) ((unsigned short) (val) << 8)))

```

Definition at line 32 of file byteswap.h.

##### 7.27.1.3 #define COB\_BSWAP\_32( val ) (COB\_BSWAP\_32\_CONSTANT (val))

Definition at line 184 of file byteswap.h.

**7.27.1.4 #define COB\_BSWAP\_32.CONSTANT( val )****Value:**

```
((unsigned int) (
    ((unsigned int) (val) & (unsigned int) 0x000000ffU) << 24) | \
    (((unsigned int) (val) & (unsigned int) 0x0000ff00U) << 8) | \
    (((unsigned int) (val) & (unsigned int) 0x00ff0000U) >> 8) | \
    (((unsigned int) (val) & (unsigned int) 0xff000000U) >> 24)))
```

Definition at line 36 of file byteswap.h.

**7.27.1.5 #define COB\_BSWAP\_64( val )(COB\_BSWAP\_64.CONSTANT(val))**

Definition at line 185 of file byteswap.h.

**7.27.1.6 #define COB\_BSWAP\_64.CONSTANT( val )****Value:**

```
((unsigned long long) ( \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x00000000000000ffULL) << 56) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x000000000000ff00ULL) << 40) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x000000000ff0000ULL) << 24) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x0000000ff000000ULL) << 8) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x000000ff0000000ULL) >> 8) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x0000ff000000000ULL) >> 24) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0x00ff00000000000ULL) >> 40) | \
    (((unsigned long long) (val) & \
    (unsigned long long) 0xff0000000000000ULL) >> 56)))
```

Definition at line 42 of file byteswap.h.

**7.28 libcob/call.c File Reference**

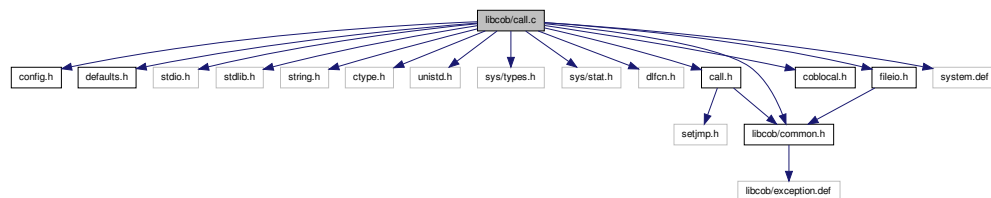
```
#include "config.h"
#include "defaults.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```

#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dlfcn.h>
#include "call.h"
#include "common.h"
#include "coblocal.h"
#include "fileio.h"
#include "system.def"

```

Include dependency graph for call.c:



## Classes

- struct [struct\\_handle](#)
- struct [call\\_hash](#)
- struct [system\\_table](#)

## Defines

- #define [\\_\\_USE\\_GNU](#) 1
- #define [lt\\_dlopen\(x\)](#) dlopen(x, RTLD\_LAZY | RTLD\_GLOBAL)
- #define [lt\\_dlsym\(x, y\)](#) dlsym(x, y)
- #define [lt\\_dlclose\(x\)](#) dlclose(x)
- #define [lt\\_dLError\(\)](#) dlerror()
- #define [lt\\_dlhandle](#) void \*
- #define [PATHSEPC](#) ':'
- #define [PATHSEPS](#) ":",
- #define [COB\\_MAX\\_COBCALL\\_PARMs](#) 16
- #define [CALL\\_BUFF\\_SIZE](#) 256
- #define [CALL\\_FILEBUFF\\_SIZE](#) 2048
- #define [HASH\\_SIZE](#) 131
- #define [COB\\_SYSTEM\\_GEN\(x, y, z\)](#) { x, (void \*)z },

## Functions

- const char \* [cob\\_resolve\\_error](#) (void)
- void [cob\\_call\\_error](#) (void)
- void [cob\\_set\\_cancel](#) (const char \*name, void \*entry, void \*cancel)
- void \* [cob\\_resolve](#) (const char \*name)
- void \* [cob\\_resolve\\_1](#) (const char \*name)
- void \* [cob\\_call\\_resolve](#) (const [cob\\_field](#) \*f)
- void \* [cob\\_call\\_resolve\\_1](#) (const [cob\\_field](#) \*f)
- void [cobcancel](#) (const char \*name)
- void [cob\\_field\\_cancel](#) (const [cob\\_field](#) \*f)
- void [cob\\_init\\_call](#) (void)
- int [cobcall](#) (const char \*name, const int argc, void \*\*argv)
- int [cobfunc](#) (const char \*name, const int argc, void \*\*argv)
- void \* [cobsavenv](#) (struct [cobjmp\\_buf](#) \*jbuf)
- void \* [cobsavenv2](#) (struct [cobjmp\\_buf](#) \*jbuf, const int jsz)
- void [coblongjmp](#) (struct [cobjmp\\_buf](#) \*jbuf)

### 7.28.1 Define Documentation

#### 7.28.1.1 #define \_\_USE\_GNU 1

Definition at line 44 of file call.c.

#### 7.28.1.2 #define CALL\_BUFF\_SIZE 256

Definition at line 100 of file call.c.

#### 7.28.1.3 #define CALL\_FILEBUFF\_SIZE 2048

Definition at line 101 of file call.c.

#### 7.28.1.4 #define COB\_MAX\_COBCALL\_PARMS 16

Definition at line 99 of file call.c.

#### 7.28.1.5 #define COB\_SYSTEM\_GEN( x, y, z ) { x, (void \*)z },

Definition at line 152 of file call.c.

#### 7.28.1.6 #define HASH\_SIZE 131

Definition at line 130 of file call.c.

**7.28.1.7 #define lt\_dlclose( x ) dlclose(x)**

Definition at line 48 of file call.c.

**7.28.1.8 #define lt\_dLError( ) dlerror()**

Definition at line 49 of file call.c.

**7.28.1.9 #define lt\_dlhandle void \***

Definition at line 50 of file call.c.

**7.28.1.10 #define lt\_dlopen( x ) dlopen(x, RTLD\_LAZY | RTLD\_GLOBAL)**

Definition at line 46 of file call.c.

**7.28.1.11 #define lt\_dlsym( x, y ) dlsym(x, y)**

Definition at line 47 of file call.c.

**7.28.1.12 #define PATHSEPC ':'**

Definition at line 95 of file call.c.

**7.28.1.13 #define PATHSEPS ":"**

Definition at line 96 of file call.c.

**7.28.2 Function Documentation****7.28.2.1 void cob\_call\_error( void )**

Definition at line 286 of file call.c.

```
{
    const char    *s;

    s = cob_resolve_error ();
    if (!s) {
        s = "Unknown error";
    }
    cob_runtime_error ("%s", s);
    cob_stop_run (1);
}
```

**7.28.2.2 void\* cob\_call\_resolve ( const cob\_field \* f )**

Definition at line 454 of file call.c.

```
{
    char    *buff;

    buff = cob_get_buff (f->size + 1);
    cob_field_to_string (f, buff);
    return cob_resolve (buff);
}
```

**7.28.2.3 void\* cob\_call\_resolve\_1 ( const cob\_field \* f )**

Definition at line 464 of file call.c.

```
{
    void    *p;

    p = cob_call_resolve (f);
    if (unlikely(!p)) {
        cob_call_error ();
    }
    return p;
}
```

**7.28.2.4 void cob\_field\_cancel ( const cob\_field \* f )**

Definition at line 504 of file call.c.

```
{
    char    *name;

    name = cob_get_buff (f->size + 1);
    cob_field_to_string (f, name);
    cobcancel (name);
}
```

**7.28.2.5 void cob\_init\_call ( void )**

Definition at line 514 of file call.c.

```
{
    char    *buff;
    char    *s;
    char    *p;
    const struct system_table *psyst;
#ifdef defined (_WIN32) || !defined (RTLD_DEFAULT)
    lt_dlhandle libhandle;
#endif
}
```



```

        size_t                i;
        struct stat           st;

#ifdef USE_LIBDL
        lt_dlinit ();
#endif

        /* big enough for anything from libdl/libltdl */
        resolve_error_buff = cob_malloc (CALL_BUFF_SIZE);

#ifdef COB_ALT_HASH
        call_table = cob_malloc (sizeof (struct call_hash *) * HASH_SIZE);
#endif

        call_filename_buff = cob_malloc (CALL_FILEBUFF_SIZE);
        call_entry_buff = cob_malloc (COB_SMALL_BUFF);
        call_entry2_buff = cob_malloc (COB_SMALL_BUFF);
        s = getenv ("COB_LOAD_CASE");
        if (s != NULL) {
            if (strcasecmp (s, "LOWER") == 0) {
                name_convert = 1;
            } else if (strcasecmp (s, "UPPER") == 0) {
                name_convert = 2;
            }
        }

        buff = cob_malloc (COB_MEDIUM_BUFF);
        s = getenv ("COB_LIBRARY_PATH");
        if (s == NULL) {
            snprintf (buff, COB_MEDIUM_MAX, "%s%s",
                PATHSEPS, COB_LIBRARY_PATH);
        } else {
            snprintf (buff, COB_MEDIUM_MAX, "%s%s.%s%s",
                s, PATHSEPS, PATHSEPS, COB_LIBRARY_PATH);
        }
        cob_set_library_path (buff);

#ifdef COB_BORKED_DLOPEN
        mainhandle = lt_dlopen (NULL);
#endif

        s = getenv ("COB_PRE_LOAD");
        if (s != NULL) {
            p = cob_strdup (s);
            s = strtok (p, PATHSEPS);
            for (; s; s = strtok (NULL, PATHSEPS)) {
                for (i = 0; i < resolve_size; ++i) {
                    buff[COB_MEDIUM_MAX] = 0;
                    snprintf (buff, COB_MEDIUM_MAX, "%s/%s.%s",
                        resolve_path[i], s, COB_MODULE_EXT);
                    if (stat (buff, &st) == 0) {
#ifdef _WIN32 || !defined (RTL_DEFAULT)
                        if ((libhandle = lt_dlopen (buff)) !=
                            NULL) {
                                cache_handle (libhandle);
                            }
                        else
                            if (lt_dlopen (buff) != NULL) {
                                break;
                            }
                        }
                    }
                }
            }
        }

```

```

        }
        free (p);
    }
    free (buff);
    call_buffer = cob_malloc (CALL_BUFFER_SIZE);
    call_lastsize = CALL_BUFFER_SIZE;
    for (psyst = (struct system_table *)&system_tab[0]; psyst->syst_name; ++p
syst) {
        insert (psyst->syst_name, psyst->syst_call, NULL);
    }
}

```

### 7.28.2.6 void\* cob\_resolve ( const char \* name )

Definition at line 317 of file call.c.

```

{
    unsigned char      *p;
    const unsigned char *s;
    void              *func;
#if defined (_WIN32) || !defined (RTLD_DEFAULT)
    struct struct_handle *chkhandle;
#endif
    lt_dlhandle        handle;
    size_t             i;
    struct stat        st;

    /* Checked in generated code
    if (!cob_initialized) {
        fputs ("cob_init() must be called before cob_resolve()", stderr);

        cob_stop_run (1);
    }
    */

    /* search the cache */
    cob_exception_code = 0;
    func = lookup (name);
    if (func) {
        return func;
    }

    /* encode program name */
    p = (unsigned char *)call_entry_buff;
    s = (const unsigned char *)name;
    if (unlikely(isdigit (*s))) {
        p += sprintf ((char *)p, "%02X", *s++);
    }
    for (; *s; ++s) {
        if (likely(isalnum (*s) || *s == '_')) {
            *p++ = *s;
        } else if (*s == '-') {
            *p++ = '_';
            *p++ = '_';
        } else {
            p += sprintf ((char *)p, "%02X", *s);
        }
    }
    *p = 0;
}

```

```

    /* search the main program */
    if (mainhandle != NULL && (func = lt_dlsym (mainhandle, call_entry_buff)
    != NULL) {
        insert (name, func, NULL);
        resolve_error = NULL;
        return func;
    }

    /* Search preloaded modules */
    #if defined (_WIN32) || !defined (RTLD_DEFAULT)
    for (chkhandle = pre_handle; chkhandle; chkhandle = chkhandle->next) {
        if ((func = lt_dlsym (chkhandle->preload_handle, call_entry_buff)
    ) != NULL) {
            insert (name, func, NULL);
            resolve_error = NULL;
            return func;
        }
    }
    #endif
    #if defined (USE_LIBDL) && defined (RTLD_DEFAULT)
    if ((func = lt_dlsym (RTLD_DEFAULT, call_entry_buff)) != NULL) {
        insert (name, func, NULL);
        resolve_error = NULL;
        return func;
    }
    #endif

    s = (const unsigned char *)name;
    if (unlikely(name_convert != 0)) {
        s = (const unsigned char *)name;
        p = call_entry2_buff;
        for (; *s; ++s) {
            if (name_convert == 1 && isupper (*s)) {
                *p++ = tolower (*s);
            } else if (name_convert == 2 && islower (*s)) {
                *p++ = toupper (*s);
            } else {
                *p++ = *s;
            }
        }
        *p = 0;
        s = (const unsigned char *)call_entry2_buff;
    }

    /* search external modules */
    for (i = 0; i < resolve_size; ++i) {
        call_filename_buff[CALL_FILEBUFF_SIZE - 1] = 0;
        if (resolve_path[i] == NULL) {
            snprintf (call_filename_buff, CALL_FILEBUFF_SIZE - 1,
                "%s.%s", s, COB_MODULE_EXT);
        } else {
            snprintf (call_filename_buff, CALL_FILEBUFF_SIZE - 1,
                "%s/%s.%s", resolve_path[i], s, COB_MODULE_EXT)
;
        }
        if (stat (call_filename_buff, &st) == 0) {
            if ((handle = lt_dlopen (call_filename_buff)) != NULL) {
                #if defined (_WIN32) || !defined (RTLD_DEFAULT)
                /* Candidate for future calls */
                cache_handle (handle);
                #endif
            }
        }
    }
}

```

```

        if ((func = lt_dlsym (handle, call_entry_buff)) !=
= NULL) {
            insert (name, func, NULL);
            resolve_error = NULL;
            return func;
        }
        resolve_error_buff[CALL_BUFF_SIZE - 1] = 0;
        strncpy (resolve_error_buff, lt_dlerror (),
                CALL_BUFF_SIZE - 1);
        resolve_error = resolve_error_buff;
        cob_set_exception (COB_EC_PROGRAM_NOT_FOUND);
        return NULL;
    }
}
resolve_error_buff[CALL_BUFF_SIZE - 1] = 0;
snprintf (resolve_error_buff, CALL_BUFF_SIZE - 1,
          "Cannot find module '%s'", name);
resolve_error = resolve_error_buff;
cob_set_exception (COB_EC_PROGRAM_NOT_FOUND);
return NULL;
}

```

#### 7.28.2.7 void\* cob\_resolve.1 ( const char \* name )

Definition at line 442 of file call.c.

```

{
    void *p;

    p = cob_resolve (name);
    if (unlikely(!p)) {
        cob_call_error ();
    }
    return p;
}

```

#### 7.28.2.8 const char\* cob\_resolve\_error ( void )

Definition at line 277 of file call.c.

```

{
    const char *p = resolve_error;

    resolve_error = NULL;
    return p;
}

```

#### 7.28.2.9 void cob\_set\_cancel ( const char \* name, void \* entry, void \* cancel )

Definition at line 299 of file call.c.

```

{
    struct call_hash      *p;

#ifdef COB_ALT_HASH
    for (p = call_table; p; p = p->next) {
#else
    for (p = call_table[hash ((const unsigned char *)name)]; p; p = p->next)
    {
#endif
        if (strcmp (name, p->name) == 0) {
            p->cancel = cancel;
            return;
        }
    }
    insert (name, entry, cancel);
}

```

#### 7.28.2.10 int cobcall ( const char \* name, const int argc, void \*\* argv )

Definition at line 596 of file call.c.

```

{
    int    i;
    union {
        void    *(*funcptr) ();
        int     (*funcint) ();
        void    *func_void;
    } unifunc;
    void    *pargv[16];

    if (unlikely(!cob_initialized)) {
        cob_runtime_error ("cobcall' - Runtime has not been initialized"
);
        cob_stop_run (1);
    }
    if (argc < 0 || argc > 16) {
        cob_runtime_error ("Invalid number of arguments to 'cobcall'");
        cob_stop_run (1);
    }
    if (unlikely(!name)) {
        cob_runtime_error ("NULL name parameter passed to 'cobcall'");
        cob_stop_run (1);
    }
    unifunc.func_void = cob_resolve_1 (name);
    memset (pargv, 0, sizeof(pargv));
    /* Set number of parameters */
    cob_call_params = argc;
    for (i = 0; i < argc; ++i) {
        pargv[i] = argv[i];
    }
    return unifunc.funcint (pargv[0], pargv[1], pargv[2], pargv[3],
        pargv[4], pargv[5], pargv[6], pargv[7],
        pargv[8], pargv[9], pargv[10], pargv[11],
        pargv[12], pargv[13], pargv[14], pargv[15]);
}

```

**7.28.2.11 void cobcancel ( const char \* name )**

Definition at line 476 of file call.c.

```

{
    struct call_hash      *p;
    union {
        int      (*cancel_func)(int, ...);
        void     *cancel_void;
    } unicanc;

    if (unlikely(!name)) {
        cob_runtime_error ("NULL name parameter passed to 'cobcancel'");
        cob_stop_run (1);
    }
#ifdef COB_ALT_HASH
    for (p = call_table; p; p = p->next) {
#else
    for (p = call_table[hash ((const unsigned char *)name)]; p; p = p->next)
    {
#endif
        if (strcmp (name, p->name) == 0) {
            if (p->cancel && !p->flag_is_active) {
                unicanc.cancel_void = p->cancel;
                unicanc.cancel_func (-1, NULL, NULL, NULL, NULL,
                                     NULL, NULL, NULL, NULL);
            }
        }
    }
}

```

**7.28.2.12 int cobfunc ( const char \* name, const int argc, void \*\* argv )**

Definition at line 632 of file call.c.

```

{
    int      ret;

    if (unlikely(!cob_initialized)) {
        cob_runtime_error ("'cobfunc' - Runtime has not been initialized"
);
        cob_stop_run (1);
    }
    ret = cobcall (name, argc, argv);
    cobcancel (name);
    return ret;
}

```

**7.28.2.13 void coblongjmp ( struct cobjmp\_buf \* jbuf )**

Definition at line 671 of file call.c.

```

{
    if (unlikely(!jbuf)) {

```

```

        cob_runtime_error ("NULL name parameter passed to 'coblongjmp'");

        cob_stop_run (1);
    }
    if (!cobjmp_primed) {
        cob_runtime_error ("Call to 'coblongjmp' with no prior 'cobsetjmp'");
        cob_stop_run (1);
    }
    cobjmp_primed = 0;
    longjmp (jbuf->cbj_jmp_buf, 1);
}

```

#### 7.28.2.14 void\* cobsavenv ( struct cobjmp\_buf \* jbuf )

Definition at line 646 of file call.c.

```

{
    if (unlikely(!jbuf)) {
        cob_runtime_error ("NULL name parameter passed to 'cobsavenv'");
        cob_stop_run (1);
    }
    if (cobjmp_primed) {
        cob_runtime_error ("Multiple call to 'cobsetjmp'");
        cob_stop_run (1);
    }
    cobjmp_primed = 1;
    return jbuf->cbj_jmp_buf;
}

```

#### 7.28.2.15 void\* cobsavenv2 ( struct cobjmp\_buf \* jbuf, const int jsize )

Definition at line 661 of file call.c.

```

{
    int    jtemp;

    /* Shut up compiler */
    jtemp = jsize;
    return cobsavenv (jbuf);
}

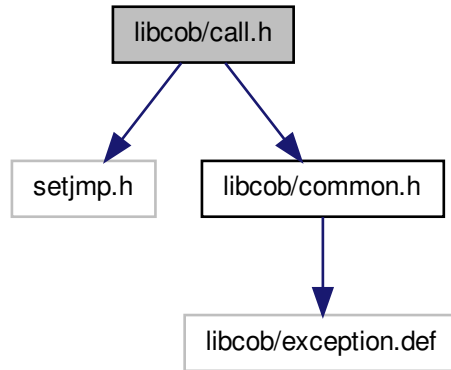
```

## 7.29 libcob/call.h File Reference

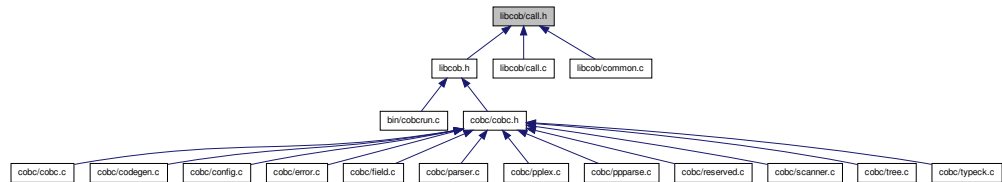
```
#include <setjmp.h>
```

```
#include <libcob/common.h>
```

Include dependency graph for call.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct `cobjmp_buf`

## Defines

- #define `cobsetjmp(x)` `setjmp (cobsavenv (x))`

## Functions

- void `cob_set_cancel` (const char \*, void \*, void \*)
- void \* `cob_resolve` (const char \*)



- void \* [cob\\_resolve\\_1](#) (const char \*)
- const char \* [cob\\_resolve\\_error](#) (void)
- void \* [cob\\_call\\_resolve](#) (const [cob\\_field](#) \*)
- void \* [cob\\_call\\_resolve\\_1](#) (const [cob\\_field](#) \*)
- void [cob\\_field\\_cancel](#) (const [cob\\_field](#) \*)
- void [cobcancel](#) (const char \*)
- int [cobcall](#) (const char \*, const int, void \*\*)
- int [cobfunc](#) (const char \*, const int, void \*\*)
- void \* [cobsavenv](#) (struct [cobjmp\\_buf](#) \*)
- void \* [cobsavenv2](#) (struct [cobjmp\\_buf](#) \*, const int)
- void [coblongjmp](#) (struct [cobjmp\\_buf](#) \*)
- void [cob\\_call\\_error](#) (void)

### 7.29.1 Define Documentation

#### 7.29.1.1 #define [cobsetjmp\( x \) setjmp \(cobsavenv \(x\)\)](#)

Definition at line 49 of file call.h.

### 7.29.2 Function Documentation

#### 7.29.2.1 void [cob\\_call\\_error](#) ( void )

Definition at line 286 of file call.c.

```
{
    const char      *s;

    s = cob_resolve_error ();
    if (!s) {
        s = "Unknown error";
    }
    cob_runtime_error ("%s", s);
    cob_stop_run (1);
}
```

#### 7.29.2.2 void\* [cob\\_call\\_resolve](#) ( const [cob\\_field](#) \* )

Definition at line 454 of file call.c.

```
{
    char      *buff;

    buff = cob_get_buff (f->size + 1);
    cob_field_to_string (f, buff);
    return cob_resolve (buff);
}
```

**7.29.2.3 void\* cob\_call\_resolve\_1 ( const cob\_field \* )**

Definition at line 464 of file call.c.

```
{
    void    *p;

    p = cob_call_resolve (f);
    if (unlikely(!p)) {
        cob_call_error ();
    }
    return p;
}
```

**7.29.2.4 void cob\_field\_cancel ( const cob\_field \* )**

Definition at line 504 of file call.c.

```
{
    char    *name;

    name = cob_get_buff (f->size + 1);
    cob_field_to_string (f, name);
    cobcancel (name);
}
```

**7.29.2.5 void\* cob\_resolve ( const char \* )**

Definition at line 317 of file call.c.

```
{
    unsigned char    *p;
    const unsigned char    *s;
    void            *func;
#ifdef _WIN32 || !defined (RTLD_DEFAULT)
    struct struct_handle    *chkhandle;
#endif
    lt_dlhandle        handle;
    size_t            i;
    struct stat        st;

    /* Checked in generated code
       if (!cob_initialized) {
           fputs ("cob_init() must be called before cob_resolve()", stderr);

           cob_stop_run (1);
       }
    */

    /* search the cache */
    cob_exception_code = 0;
    func = lookup (name);
    if (func) {
        return func;
    }
}
```

```

    }

    /* encode program name */
    p = (unsigned char *)call_entry_buff;
    s = (const unsigned char *)name;
    if (unlikely(isdigit (*s))) {
        p += sprintf ((char *)p, "%02X", *s++);
    }
    for (; *s; ++s) {
        if (likely(isalnum (*s) || *s == '_')) {
            *p++ = *s;
        } else if (*s == '-') {
            *p++ = '_';
            *p++ = '_';
        } else {
            p += sprintf ((char *)p, "%02X", *s);
        }
    }
    *p = 0;

    /* search the main program */
    if (mainhandle != NULL && (func = lt_dlsym (mainhandle, call_entry_buff)
    != NULL) {
        insert (name, func, NULL);
        resolve_error = NULL;
        return func;
    }

    /* Search preloaded modules */
    #if defined (_WIN32) || !defined (RTLD_DEFAULT)
    for (chkhandle = pre_handle; chkhandle; chkhandle = chkhandle->next) {
        if ((func = lt_dlsym (chkhandle->preload_handle, call_entry_buff)
    != NULL) {
            insert (name, func, NULL);
            resolve_error = NULL;
            return func;
        }
    }
    #endif
    #if defined (USE_LIBDL) && defined (RTLD_DEFAULT)
    if ((func = lt_dlsym (RTLD_DEFAULT, call_entry_buff)) != NULL) {
        insert (name, func, NULL);
        resolve_error = NULL;
        return func;
    }
    #endif

    s = (const unsigned char *)name;
    if (unlikely(name_convert != 0)) {
        s = (const unsigned char *)name;
        p = call_entry2_buff;
        for (; *s; ++s) {
            if (name_convert == 1 && isupper (*s)) {
                *p++ = tolower (*s);
            } else if (name_convert == 2 && islower (*s)) {
                *p++ = toupper (*s);
            } else {
                *p++ = *s;
            }
        }
        *p = 0;
        s = (const unsigned char *)call_entry2_buff;
    }

```

```

    }

    /* search external modules */
    for (i = 0; i < resolve_size; ++i) {
        call_filename_buff[CALL_FILEBUFF_SIZE - 1] = 0;
        if (resolve_path[i] == NULL) {
            snprintf (call_filename_buff, CALL_FILEBUFF_SIZE - 1,
                    "%s.%s", s, COB_MODULE_EXT);
        } else {
            snprintf (call_filename_buff, CALL_FILEBUFF_SIZE - 1,
                    "%s/%s.%s", resolve_path[i], s, COB_MODULE_EXT)
        }
    }
    if (stat (call_filename_buff, &st) == 0) {
        if ((handle = lt_dlopen (call_filename_buff)) != NULL) {
#if defined (_WIN32) || !defined (RTL_D_DEFAULT)
            /* Candidate for future calls */
            cache_handle (handle);
#endif
            if ((func = lt_dlsym (handle, call_entry_buff)) !=
                = NULL) {
                insert (name, func, NULL);
                resolve_error = NULL;
                return func;
            }
            resolve_error_buff[CALL_BUFF_SIZE - 1] = 0;
            strncpy (resolve_error_buff, lt_dlerror (),
                    CALL_BUFF_SIZE - 1);
            resolve_error = resolve_error_buff;
            cob_set_exception (COB_EC_PROGRAM_NOT_FOUND);
            return NULL;
        }
    }
    resolve_error_buff[CALL_BUFF_SIZE - 1] = 0;
    snprintf (resolve_error_buff, CALL_BUFF_SIZE - 1,
            "Cannot find module '%s'", name);
    resolve_error = resolve_error_buff;
    cob_set_exception (COB_EC_PROGRAM_NOT_FOUND);
    return NULL;
}

```

### 7.29.2.6 void\* cob\_resolve\_1 ( const char \* )

Definition at line 442 of file call.c.

```

{
    void    *p;

    p = cob_resolve (name);
    if (unlikely(!p)) {
        cob_call_error ();
    }
    return p;
}

```

**7.29.2.7 const char\* cob\_resolve\_error ( void )**

Definition at line 277 of file call.c.

```

{
    const char    *p = resolve_error;

    resolve_error = NULL;
    return p;
}

```

**7.29.2.8 void cob\_set\_cancel ( const char \*, void \*, void \* )**

Definition at line 299 of file call.c.

```

{
    struct call_hash    *p;

#ifdef COB_ALT_HASH
    for (p = call_table; p; p = p->next) {
#else
    for (p = call_table[hash ((const unsigned char *)name)]; p; p = p->next)
    {
#endif
        if (strcmp (name, p->name) == 0) {
            p->cancel = cancel;
            return;
        }
    }
    insert (name, entry, cancel);
}

```

**7.29.2.9 int cobcall ( const char \*, const int, void \*\* )**

Definition at line 596 of file call.c.

```

{
    int    i;
    union {
        void    *(*funcptr) ();
        int    (*funcint) ();
        void    *func_void;
    } unifunc;
    void    *pargv[16];

    if (unlikely(!cob_initialized)) {
        cob_runtime_error ("cobcall' - Runtime has not been initialized"
);
        cob_stop_run (1);
    }
    if (argc < 0 || argc > 16) {
        cob_runtime_error ("Invalid number of arguments to 'cobcall'");
        cob_stop_run (1);
    }
}

```

```

    if (unlikely(!name)) {
        cob_runtime_error ("NULL name parameter passed to 'cobcall'");
        cob_stop_run (1);
    }
    unifunc.func_void = cob_resolve_1 (name);
    memset (pargv, 0, sizeof(pargv));
    /* Set number of parameters */
    cob_call_params = argc;
    for (i = 0; i < argc; ++i) {
        pargv[i] = argv[i];
    }
    return unifunc.funcint (pargv[0], pargv[1], pargv[2], pargv[3],
                           pargv[4], pargv[5], pargv[6], pargv[7],
                           pargv[8], pargv[9], pargv[10], pargv[11],
                           pargv[12], pargv[13], pargv[14], pargv[15]);
}

```

#### 7.29.2.10 void cobcancel ( const char \* )

Definition at line 476 of file call.c.

```

{
    struct call_hash      *p;
    union {
        int      (*cancel_func)(int, ...);
        void     *cancel_void;
    } unicanc;

    if (unlikely(!name)) {
        cob_runtime_error ("NULL name parameter passed to 'cobcancel'");
        cob_stop_run (1);
    }
#ifdef COB_ALT_HASH
    for (p = call_table; p; p = p->next) {
#else
    for (p = call_table[hash ((const unsigned char *)name)]; p; p = p->next)
#endif
    {
        if (strcmp (name, p->name) == 0) {
            if (p->cancel && !p->flag_is_active) {
                unicanc.cancel_void = p->cancel;
                unicanc.cancel_func (-1, NULL, NULL, NULL, NULL,
                                     NULL, NULL, NULL, NULL);
            }
        }
    }
}

```

#### 7.29.2.11 int cobfunc ( const char \*, const int , void \*\* )

Definition at line 632 of file call.c.

```

{
    int     ret;

```

```
    if (unlikely(!cob_initialized)) {
        cob_runtime_error ("cobfunc' - Runtime has not been initialized"
);
        cob_stop_run (1);
    }
    ret = cobcall (name, argc, argv);
    cobcancel (name);
    return ret;
}
```

#### 7.29.2.12 void coblongjmp ( struct cobjmp\_buf \* )

Definition at line 671 of file call.c.

```
{
    if (unlikely(!jbuf)) {
        cob_runtime_error ("NULL name parameter passed to 'coblongjmp'");
        cob_stop_run (1);
    }
    if (!cobjmp_primed) {
        cob_runtime_error ("Call to 'coblongjmp' with no prior 'cobsetjmp'
");
        cob_stop_run (1);
    }
    cobjmp_primed = 0;
    longjmp (jbuf->cbj_jmp_buf, 1);
}
```

#### 7.29.2.13 void\* cobsavenv ( struct cobjmp\_buf \* )

Definition at line 646 of file call.c.

```
{
    if (unlikely(!jbuf)) {
        cob_runtime_error ("NULL name parameter passed to 'cobsavenv'");
        cob_stop_run (1);
    }
    if (cobjmp_primed) {
        cob_runtime_error ("Multiple call to 'cobsetjmp'");
        cob_stop_run (1);
    }
    cobjmp_primed = 1;
    return jbuf->cbj_jmp_buf;
}
```

#### 7.29.2.14 void\* cobsavenv2 ( struct cobjmp\_buf \*, const int )

Definition at line 661 of file call.c.

```
{
    int    jtemp;
```

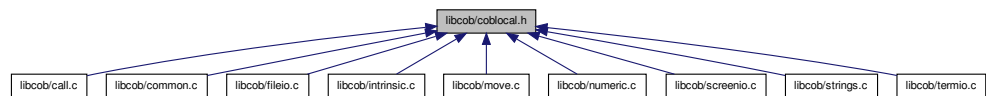
```

    /* Shut up compiler */
    jtemp = jsize;
    return cobsavenv (jbuf);
}

```

## 7.30 libcob/coblocal.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define [COB\\_HIDDEN](#)
- #define [COB\\_ATTR\\_INIT](#)(v, w, x, y, z)
- #define [COB\\_CHK\\_PARMS](#)(x, z)

### Functions

- COB\_HIDDEN void [cob\\_memcpy](#) (cob\_field \*, unsigned char \*, const int)
- COB\_HIDDEN void [cob\\_exit\\_fileio](#) (void)
- COB\_HIDDEN void [cob\\_field\\_to\\_string](#) (const cob\_field \*, char \*)
- COB\_HIDDEN void [cob\\_init\\_numeric](#) (void)
- COB\_HIDDEN void [cob\\_init\\_termio](#) (void)
- COB\_HIDDEN void [cob\\_init\\_fileio](#) (void)
- COB\_HIDDEN void [cob\\_init\\_call](#) (void)
- COB\_HIDDEN void [cob\\_init\\_intrinsic](#) (void)
- COB\_HIDDEN void [cob\\_init\\_strings](#) (void)
- COB\_HIDDEN void [cob\\_init\\_move](#) (void)
- COB\_HIDDEN void [cob\\_screen\\_terminate](#) (void)
- COB\_HIDDEN void [cob\\_screen\\_set\\_mode](#) (size\_t)
- COB\_HIDDEN int [cob\\_real\\_get\\_sign](#) (cob\_field \*)
- COB\_HIDDEN void [cob\\_real\\_put\\_sign](#) (cob\_field \*, const int)
- COB\_HIDDEN long long [cob\\_get\\_long\\_long](#) (cob\_field \*)



## Variables

- COB\_HIDDEN int [cob\\_screen\\_initialized](#)
- COB\_HIDDEN int [cob\\_got\\_exception](#)
- COB\_HIDDEN unsigned int [cob\\_orig\\_line](#)
- COB\_HIDDEN const char \* [cob\\_orig\\_statement](#)
- COB\_HIDDEN const char \* [cob\\_orig\\_program\\_id](#)
- COB\_HIDDEN const char \* [cob\\_orig\\_section](#)
- COB\_HIDDEN const char \* [cob\\_orig\\_paragraph](#)

### 7.30.1 Define Documentation

#### 7.30.1.1 #define COB\_ATTR\_INIT( v, w, x, y, z )

##### Value:

```
do { \  
    attr.type = v; \  
    attr.digits = w; \  
    attr.scale = x; \  
    attr.flags = y; \  
    attr.pic = z; \  
} while (0)
```

Definition at line 33 of file coblocal.h.

#### 7.30.1.2 #define COB\_CHK\_PARAMS( x, z )

Definition at line 50 of file coblocal.h.

#### 7.30.1.3 #define COB\_HIDDEN

Definition at line 30 of file coblocal.h.

### 7.30.2 Function Documentation

#### 7.30.2.1 COB\_HIDDEN void cob\_exit\_fileio ( void )

Definition at line 4426 of file fileio.c.

```
{  
    struct file_list    *l;  
  
    for (l = file_cache; l; l = l->next) {  
        if (l->file->open_mode != COB_OPEN_CLOSED &&  
            l->file->open_mode != COB_OPEN_LOCKED) {  
            cob_field_to_string (l->file->assign, runtime_buffer);  
            cob_close (l->file, 0, NULL);  
            fprintf (stderr, "WARNING - Implicit CLOSE of %s (\"%s\")  
}
```

```

        \n",
                                l->file->select_name, runtime_buffer);
                                fflush (stderr);
                                }
                                }
#ifdef USE_DB41
    free (record_lock_object);
    if (bdb_env) {
        bdb_env->lock_id_free (bdb_env, bdb_lock_id);
        bdb_env->close (bdb_env, 0);
    }
#endif
#if defined(WITH_INDEX_EXTFH) || defined(WITH_SEQRA_EXTFH)
    extfh_cob_exit_fileio ();
#endif
}

```

**7.30.2.2 COB\_HIDDEN void cob.field.to.string ( const cob\_field \*, char \* )**

**7.30.2.3 COB\_HIDDEN long long cob.get.long.long ( cob\_field \* )**

Definition at line 1329 of file move.c.

```

{
    long long      n;
    cob_field      temp;
    cob_field_attr attr;

    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_display_get_long_long (f);
    case COB_TYPE_NUMERIC_BINARY:
        return cob_binary_mget_int64 (f);
    case COB_TYPE_NUMERIC_PACKED:
        return cob_packed_get_long_long (f);
    default:
        COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0,
                      COB_FLAG_HAVE_SIGN, NULL);
        temp.size = 8;
        temp.data = (unsigned char *)&n;
        temp.attr = &attr;
        cob_move (f, &temp);
        return n;
    }
}

```

**7.30.2.4 COB\_HIDDEN void cob.init.call ( void )**

Definition at line 514 of file call.c.

```

{
    char            *buff;
    char            *s;
    char            *p;
    const struct system_table *psyst;
}

```

```

#if defined (_WIN32) || !defined (RTLD_DEFAULT)
    lt_dlhandle          libhandle;
#endif

    size_t                i;
    struct stat           st;

#ifndef USE_LIBDL
    lt_dlinit ();
#endif

    /* big enough for anything from libdl/libltdl */
    resolve_error_buff = cob_malloc (CALL_BUFF_SIZE);

#ifndef COB_ALT_HASH
    call_table = cob_malloc (sizeof (struct call_hash *) * HASH_SIZE);
#endif

    call_filename_buff = cob_malloc (CALL_FILEBUFF_SIZE);
    call_entry_buff = cob_malloc (COB_SMALL_BUFF);
    call_entry2_buff = cob_malloc (COB_SMALL_BUFF);
    s = getenv ("COB_LOAD_CASE");
    if (s != NULL) {
        if (strcasemp (s, "LOWER") == 0) {
            name_convert = 1;
        } else if (strcasemp (s, "UPPER") == 0) {
            name_convert = 2;
        }
    }

    buff = cob_malloc (COB_MEDIUM_BUFF);
    s = getenv ("COB_LIBRARY_PATH");
    if (s == NULL) {
        snprintf (buff, COB_MEDIUM_MAX, "%s%s",
            PATHSEPS, COB_LIBRARY_PATH);
    } else {
        snprintf (buff, COB_MEDIUM_MAX, "%s%s.%s%s",
            s, PATHSEPS, PATHSEPS, COB_LIBRARY_PATH);
    }
    cob_set_library_path (buff);

#ifndef COB_BORKED_DLOPEN
    mainhandle = lt_dlopen (NULL);
#endif

    s = getenv ("COB_PRE_LOAD");
    if (s != NULL) {
        p = cob_strdup (s);
        s = strtok (p, PATHSEPS);
        for (; s; s = strtok (NULL, PATHSEPS)) {
            for (i = 0; i < resolve_size; ++i) {
                buff[COB_MEDIUM_MAX] = 0;
                snprintf (buff, COB_MEDIUM_MAX, "%s/%s.%s",
                    resolve_path[i], s, COB_MODULE_EXT);
                if (stat (buff, &st) == 0) {
#if defined (_WIN32) || !defined (RTLD_DEFAULT)
                    if ((libhandle = lt_dlopen (buff)) !=
                        NULL) {
                        cache_handle (libhandle);
                    }
#else
                    if (lt_dlopen (buff) != NULL) {
#endif
                        break;
                    }
                }
            }
        }
    }

```

```

    }
    }
    }
    free (p);
}
free (buff);
call_buffer = cob_malloc (CALL_BUFF_SIZE);
call_lastsize = CALL_BUFF_SIZE;
for (psyst = (struct system_table *)&system_tab[0]; psyst->syst_name; ++p
syst) {
    insert (psyst->syst_name, psyst->syst_call, NULL);
}
}

```

### 7.30.2.5 COB\_HIDDEN void cob\_init\_fileio ( void )

Definition at line 4379 of file fileio.c.

```

{
    char    *s;
    int     n;

    if ((s = getenv ("COB_SYNC")) != NULL) {
        if (*s == 'Y' || *s == 'y') {
            cob_do_sync = 1;
        }
        if (*s == 'P' || *s == 'p') {
            cob_do_sync = 2;
        }
    }
    if ((s = getenv ("COB_SORT_MEMORY")) != NULL) {
        n = atoi (s);
        if (n >= 1024*1024) {
            cob_sort_memory = n;
        }
    }
    cob_file_path = getenv ("COB_FILE_PATH");
    if (cob_file_path) {
        if (!*cob_file_path || *cob_file_path == ' ') {
            cob_file_path = NULL;
        }
    }
    cob_ls_nulls = getenv ("COB_LS_NULLS");
    cob_ls_fixed = getenv ("COB_LS_FIXED");

    file_open_env = cob_malloc (COB_SMALL_BUFF);
    file_open_name = cob_malloc (COB_SMALL_BUFF);
    file_open_buff = cob_malloc (COB_SMALL_BUFF);

#ifdef USE_DB41
    bdb_home = getenv ("DB_HOME");
    join_environment ();
    record_lock_object = cob_malloc (1024);
    bdb_buff = cob_malloc (COB_SMALL_BUFF);
    rlo_size = 1024;
#endif

    #if defined(WITH_INDEX_EXTFH) || defined(WITH_SEQRA_EXTFH)

```

```

        extfh_cob_init_fileio (&sequential_funcs, &lineseq_funcs, &relative_funcs
        , &cob_file_write_opt);
#endif
}

```

### 7.30.2.6 COB\_HIDDEN void cob\_init\_intrinsic ( void )

Definition at line 3269 of file intrinsic.c.

```

{
    size_t      i;

    cob_decimal_init (&d1);
    cob_decimal_init (&d2);
    cob_decimal_init (&d3);
    cob_decimal_init (&d4);
    cob_decimal_init (&d5);
    /* mpz_init2 (mp, 256); */
    memset ((char *)&calc_field[0], 0, sizeof (calc_field));
    memset ((char *)&calc_attr[0], 0, sizeof (calc_attr));
    for (i = 0; i < DEPTH_LEVEL; ++i) {
        calc_field[i].data = cob_malloc (256);
        calc_field[i].size = 256;
        calc_size[i] = 256;
    }
    locale_buff = cob_malloc (COB_SMALL_BUFF);
}

```

### 7.30.2.7 COB\_HIDDEN void cob\_init\_move ( void )

Definition at line 1354 of file move.c.

```

{
    lastdata = cob_malloc (COB_SMALL_BUFF);
    lastsize = COB_SMALL_BUFF;
}

```

### 7.30.2.8 COB\_HIDDEN void cob\_init\_numeric ( void )

Definition at line 1396 of file numeric.c.

```

{
    size_t  i;

    cob_decimal_init (&cob_d1);
    cob_decimal_init (&cob_d2);
    cob_decimal_init (&cob_d3);
    cob_decimal_init (&cob_d4);
    mpz_init2 (cob_mpzt, 256);
    mpz_init2 (cob_mexp, 512);
    for (i = 0; i < COB_MAX_BINARY; i++) {

```

```

        mpz_init (cob_mpze10[i]);
        mpz_ui_pow_ui (cob_mpze10[i], 10, i);
    }
    num_buff_ptr = cob_malloc (2048);
    memset (packed_value, 0, sizeof(packed_value));
}

```

### 7.30.2.9 COB\_HIDDEN void cob\_init\_strings ( void )

Definition at line 595 of file strings.c.

```

{
    inspect_mark = cob_malloc (COB_MEDIUM_BUFF);
    lastsize = COB_MEDIUM_BUFF;
    alpha_attr.type = COB_TYPE_ALPHANUMERIC;
    alpha_attr.digits = 0;
    alpha_attr.scale = 0;
    alpha_attr.flags = 0;
    alpha_attr.pic = NULL;
    alpha_fld.size = 0;
    alpha_fld.data = NULL;
    alpha_fld.attr = &alpha_attr;
}

```

### 7.30.2.10 COB\_HIDDEN void cob\_init\_termio ( void )

Definition at line 279 of file termio.c.

```

{
    term_buff = cob_malloc (COB_MEDIUM_BUFF);
}

```

### 7.30.2.11 COB\_HIDDEN void cob\_memcpy ( cob\_field \*, unsigned char \*, const int )

### 7.30.2.12 COB\_HIDDEN int cob\_real\_get\_sign ( cob\_field \* )

### 7.30.2.13 COB\_HIDDEN void cob\_real\_put\_sign ( cob\_field \*, const int )

### 7.30.2.14 COB\_HIDDEN void cob\_screen\_set\_mode ( size\_t )

Definition at line 1058 of file screenio.c.

```

{
    if (!smode) {
        refresh ();
        def_prog_mode ();
        endwin ();
    } else {
        reset_prog_mode ();
        refresh ();
    }
}

```

**7.30.2.15 COB\_HIDDEN void cob\_screen\_terminate ( void )**

Definition at line 297 of file screenio.c.

```
{
    if (cob_screen_initialized) {
        cob_screen_initialized = 0;
        endwin ();
    }
}
```

**7.30.3 Variable Documentation****7.30.3.1 COB\_HIDDEN int cob\_got\_exception**

Definition at line 166 of file common.c.

**7.30.3.2 COB\_HIDDEN unsigned int cob\_orig\_line**

Definition at line 172 of file common.c.

**7.30.3.3 COB\_HIDDEN const char\* cob\_orig\_paragraph**

Definition at line 171 of file common.c.

**7.30.3.4 COB\_HIDDEN const char\* cob\_orig\_program\_id**

Definition at line 169 of file common.c.

**7.30.3.5 COB\_HIDDEN const char\* cob\_orig\_section**

Definition at line 170 of file common.c.

**7.30.3.6 COB\_HIDDEN const char\* cob\_orig\_statement**

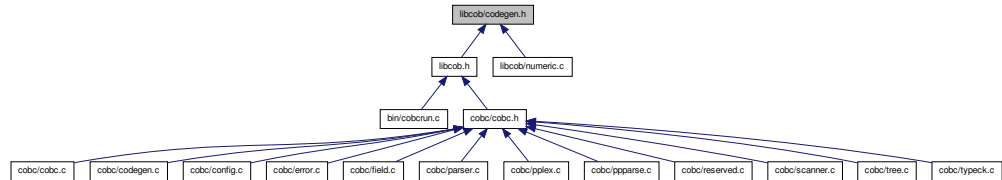
Definition at line 168 of file common.c.

**7.30.3.7 COB\_HIDDEN int cob\_screen\_initialized**

Definition at line 51 of file screenio.c.

## 7.31 libcob/codegen.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define [COB\\_STATIC](#)

### Functions

- int [cob\\_get\\_numdisp](#) (const unsigned char \*, const size\_t)
- int [cob\\_cmp\\_packed\\_int](#) (const [cob\\_field](#) \*, const int)
- int [cob\\_get\\_packed\\_int](#) (const [cob\\_field](#) \*)
- int [cob\\_add\\_packed\\_int](#) ([cob\\_field](#) \*, const int)
- int [cob\\_cmp\\_u8\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s8\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u24\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s24\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u40\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s40\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u48\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s48\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u56\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s56\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_u64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_s64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_u16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_s16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_u32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_s32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_u64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmp\\_align\\_s64\\_binary](#) (const unsigned char \*, const int)



- void [cob\\_add\\_u8\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s8\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_s64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_add\\_align\\_s64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u8\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s8\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_s64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_sub\\_align\\_s64\\_binary](#) (unsigned char \*, const int)
- int [cob\\_cmpswp\\_u16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s16\\_binary](#) (const unsigned char \*, const int)

- int [cob\\_cmpswp\\_u24\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s24\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_u32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_u40\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s40\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_u48\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s48\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_u56\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s56\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_u64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_s64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_u16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_s16\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_u32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_s32\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_u64\\_binary](#) (const unsigned char \*, const int)
- int [cob\\_cmpswp\\_align\\_s64\\_binary](#) (const unsigned char \*, const int)
- void [cob\\_addswp\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_addswp\\_s64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_subswp\\_s64\\_binary](#) (unsigned char \*, const int)

- void [cob\\_setswp\\_u16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s16\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s24\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s32\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s40\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s48\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s56\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_u64\\_binary](#) (unsigned char \*, const int)
- void [cob\\_setswp\\_s64\\_binary](#) (unsigned char \*, const int)

### 7.31.1 Define Documentation

#### 7.31.1.1 #define COB\_STATIC

Definition at line 26 of file codegen.h.

### 7.31.2 Function Documentation

7.31.2.1 void [cob\\_add\\_align\\_s16\\_binary](#) ( unsigned char \*, const int )

7.31.2.2 void [cob\\_add\\_align\\_s32\\_binary](#) ( unsigned char \*, const int )

7.31.2.3 void [cob\\_add\\_align\\_s64\\_binary](#) ( unsigned char \*, const int )

7.31.2.4 void [cob\\_add\\_align\\_u16\\_binary](#) ( unsigned char \*, const int )

7.31.2.5 void [cob\\_add\\_align\\_u32\\_binary](#) ( unsigned char \*, const int )

7.31.2.6 void [cob\\_add\\_align\\_u64\\_binary](#) ( unsigned char \*, const int )

7.31.2.7 int [cob\\_add\\_packed\\_int](#) ( [cob\\_field](#) \*, const int )

7.31.2.8 void [cob\\_add\\_s16\\_binary](#) ( unsigned char \*, const int )

7.31.2.9 void [cob\\_add\\_s24\\_binary](#) ( unsigned char \*, const int )

7.31.2.10 void [cob\\_add\\_s32\\_binary](#) ( unsigned char \*, const int )

7.31.2.11 void [cob\\_add\\_s40\\_binary](#) ( unsigned char \*, const int )

- 7.31.2.12 void cob\_add\_s48\_binary ( unsigned char \* , const int )
- 7.31.2.13 void cob\_add\_s56\_binary ( unsigned char \* , const int )
- 7.31.2.14 void cob\_add\_s64\_binary ( unsigned char \* , const int )
- 7.31.2.15 void cob\_add\_s8\_binary ( unsigned char \* , const int )
- 7.31.2.16 void cob\_add\_u16\_binary ( unsigned char \* , const int )
- 7.31.2.17 void cob\_add\_u24\_binary ( unsigned char \* , const int )
- 7.31.2.18 void cob\_add\_u32\_binary ( unsigned char \* , const int )
- 7.31.2.19 void cob\_add\_u40\_binary ( unsigned char \* , const int )
- 7.31.2.20 void cob\_add\_u48\_binary ( unsigned char \* , const int )
- 7.31.2.21 void cob\_add\_u56\_binary ( unsigned char \* , const int )
- 7.31.2.22 void cob\_add\_u64\_binary ( unsigned char \* , const int )
- 7.31.2.23 void cob\_add\_u8\_binary ( unsigned char \* , const int )
- 7.31.2.24 void cob\_addswp\_s16\_binary ( unsigned char \* , const int )
- 7.31.2.25 void cob\_addswp\_s24\_binary ( unsigned char \* , const int )
- 7.31.2.26 void cob\_addswp\_s32\_binary ( unsigned char \* , const int )
- 7.31.2.27 void cob\_addswp\_s40\_binary ( unsigned char \* , const int )
- 7.31.2.28 void cob\_addswp\_s48\_binary ( unsigned char \* , const int )
- 7.31.2.29 void cob\_addswp\_s56\_binary ( unsigned char \* , const int )
- 7.31.2.30 void cob\_addswp\_s64\_binary ( unsigned char \* , const int )
- 7.31.2.31 void cob\_addswp\_u16\_binary ( unsigned char \* , const int )
- 7.31.2.32 void cob\_addswp\_u24\_binary ( unsigned char \* , const int )
- 7.31.2.33 void cob\_addswp\_u32\_binary ( unsigned char \* , const int )
- 7.31.2.34 void cob\_addswp\_u40\_binary ( unsigned char \* , const int )
- 7.31.2.35 void cob\_addswp\_u48\_binary ( unsigned char \* , const int )

- 7.31.2.36 void cob\_addswp\_u56\_binary ( unsigned char \* , const int )
- 7.31.2.37 void cob\_addswp\_u64\_binary ( unsigned char \* , const int )
- 7.31.2.38 int cob\_cmp\_align\_s16\_binary ( const unsigned char \* , const int )
- 7.31.2.39 int cob\_cmp\_align\_s32\_binary ( const unsigned char \* , const int )
- 7.31.2.40 int cob\_cmp\_align\_s64\_binary ( const unsigned char \* , const int )
- 7.31.2.41 int cob\_cmp\_align\_u16\_binary ( const unsigned char \* , const int )
- 7.31.2.42 int cob\_cmp\_align\_u32\_binary ( const unsigned char \* , const int )
- 7.31.2.43 int cob\_cmp\_align\_u64\_binary ( const unsigned char \* , const int )
- 7.31.2.44 int cob\_cmp\_packed\_int ( const cob\_field \* , const int )
- 7.31.2.45 int cob\_cmp\_s16\_binary ( const unsigned char \* , const int )
- 7.31.2.46 int cob\_cmp\_s24\_binary ( const unsigned char \* , const int )
- 7.31.2.47 int cob\_cmp\_s32\_binary ( const unsigned char \* , const int )
- 7.31.2.48 int cob\_cmp\_s40\_binary ( const unsigned char \* , const int )
- 7.31.2.49 int cob\_cmp\_s48\_binary ( const unsigned char \* , const int )
- 7.31.2.50 int cob\_cmp\_s56\_binary ( const unsigned char \* , const int )
- 7.31.2.51 int cob\_cmp\_s64\_binary ( const unsigned char \* , const int )
- 7.31.2.52 int cob\_cmp\_s8\_binary ( const unsigned char \* , const int )
- 7.31.2.53 int cob\_cmp\_u16\_binary ( const unsigned char \* , const int )
- 7.31.2.54 int cob\_cmp\_u24\_binary ( const unsigned char \* , const int )
- 7.31.2.55 int cob\_cmp\_u32\_binary ( const unsigned char \* , const int )
- 7.31.2.56 int cob\_cmp\_u40\_binary ( const unsigned char \* , const int )
- 7.31.2.57 int cob\_cmp\_u48\_binary ( const unsigned char \* , const int )
- 7.31.2.58 int cob\_cmp\_u56\_binary ( const unsigned char \* , const int )
- 7.31.2.59 int cob\_cmp\_u64\_binary ( const unsigned char \* , const int )

- 7.31.2.60 `int cob_cmp_u8_binary ( const unsigned char * , const int )`
- 7.31.2.61 `int cob_cmpswp_align_s16_binary ( const unsigned char * , const int )`
- 7.31.2.62 `int cob_cmpswp_align_s32_binary ( const unsigned char * , const int )`
- 7.31.2.63 `int cob_cmpswp_align_s64_binary ( const unsigned char * , const int )`
- 7.31.2.64 `int cob_cmpswp_align_u16_binary ( const unsigned char * , const int )`
- 7.31.2.65 `int cob_cmpswp_align_u32_binary ( const unsigned char * , const int )`
- 7.31.2.66 `int cob_cmpswp_align_u64_binary ( const unsigned char * , const int )`
- 7.31.2.67 `int cob_cmpswp_s16_binary ( const unsigned char * , const int )`
- 7.31.2.68 `int cob_cmpswp_s24_binary ( const unsigned char * , const int )`
- 7.31.2.69 `int cob_cmpswp_s32_binary ( const unsigned char * , const int )`
- 7.31.2.70 `int cob_cmpswp_s40_binary ( const unsigned char * , const int )`
- 7.31.2.71 `int cob_cmpswp_s48_binary ( const unsigned char * , const int )`
- 7.31.2.72 `int cob_cmpswp_s56_binary ( const unsigned char * , const int )`
- 7.31.2.73 `int cob_cmpswp_s64_binary ( const unsigned char * , const int )`
- 7.31.2.74 `int cob_cmpswp_u16_binary ( const unsigned char * , const int )`
- 7.31.2.75 `int cob_cmpswp_u24_binary ( const unsigned char * , const int )`
- 7.31.2.76 `int cob_cmpswp_u32_binary ( const unsigned char * , const int )`
- 7.31.2.77 `int cob_cmpswp_u40_binary ( const unsigned char * , const int )`
- 7.31.2.78 `int cob_cmpswp_u48_binary ( const unsigned char * , const int )`
- 7.31.2.79 `int cob_cmpswp_u56_binary ( const unsigned char * , const int )`
- 7.31.2.80 `int cob_cmpswp_u64_binary ( const unsigned char * , const int )`
- 7.31.2.81 `int cob_get_numdisp ( const unsigned char * , const size_t )`
- 7.31.2.82 `int cob_get_packed_int ( const cob_field * )`
- 7.31.2.83 `void cob_setswp_s16_binary ( unsigned char * , const int )`

- 7.31.2.84 void cob\_setswp\_s24\_binary ( unsigned char \* , const int )
- 7.31.2.85 void cob\_setswp\_s32\_binary ( unsigned char \* , const int )
- 7.31.2.86 void cob\_setswp\_s40\_binary ( unsigned char \* , const int )
- 7.31.2.87 void cob\_setswp\_s48\_binary ( unsigned char \* , const int )
- 7.31.2.88 void cob\_setswp\_s56\_binary ( unsigned char \* , const int )
- 7.31.2.89 void cob\_setswp\_s64\_binary ( unsigned char \* , const int )
- 7.31.2.90 void cob\_setswp\_u16\_binary ( unsigned char \* , const int )
- 7.31.2.91 void cob\_setswp\_u24\_binary ( unsigned char \* , const int )
- 7.31.2.92 void cob\_setswp\_u32\_binary ( unsigned char \* , const int )
- 7.31.2.93 void cob\_setswp\_u40\_binary ( unsigned char \* , const int )
- 7.31.2.94 void cob\_setswp\_u48\_binary ( unsigned char \* , const int )
- 7.31.2.95 void cob\_setswp\_u56\_binary ( unsigned char \* , const int )
- 7.31.2.96 void cob\_setswp\_u64\_binary ( unsigned char \* , const int )
- 7.31.2.97 void cob\_sub\_align\_s16\_binary ( unsigned char \* , const int )
- 7.31.2.98 void cob\_sub\_align\_s32\_binary ( unsigned char \* , const int )
- 7.31.2.99 void cob\_sub\_align\_s64\_binary ( unsigned char \* , const int )
- 7.31.2.100 void cob\_sub\_align\_u16\_binary ( unsigned char \* , const int )
- 7.31.2.101 void cob\_sub\_align\_u32\_binary ( unsigned char \* , const int )
- 7.31.2.102 void cob\_sub\_align\_u64\_binary ( unsigned char \* , const int )
- 7.31.2.103 void cob\_sub\_s16\_binary ( unsigned char \* , const int )
- 7.31.2.104 void cob\_sub\_s24\_binary ( unsigned char \* , const int )
- 7.31.2.105 void cob\_sub\_s32\_binary ( unsigned char \* , const int )
- 7.31.2.106 void cob\_sub\_s40\_binary ( unsigned char \* , const int )
- 7.31.2.107 void cob\_sub\_s48\_binary ( unsigned char \* , const int )

7.31.2.108 void cob\_sub\_s56\_binary ( unsigned char \*, const int )

7.31.2.109 void cob\_sub\_s64\_binary ( unsigned char \*, const int )

7.31.2.110 void cob\_sub\_s8\_binary ( unsigned char \*, const int )

7.31.2.111 void cob\_sub\_u16\_binary ( unsigned char \*, const int )

7.31.2.112 void cob\_sub\_u24\_binary ( unsigned char \*, const int )

7.31.2.113 void cob\_sub\_u32\_binary ( unsigned char \*, const int )

7.31.2.114 void cob\_sub\_u40\_binary ( unsigned char \*, const int )

7.31.2.115 void cob\_sub\_u48\_binary ( unsigned char \*, const int )

7.31.2.116 void cob\_sub\_u56\_binary ( unsigned char \*, const int )

7.31.2.117 void cob\_sub\_u64\_binary ( unsigned char \*, const int )

7.31.2.118 void cob\_sub\_u8\_binary ( unsigned char \*, const int )

7.31.2.119 void cob\_subswp\_s16\_binary ( unsigned char \*, const int )

7.31.2.120 void cob\_subswp\_s24\_binary ( unsigned char \*, const int )

7.31.2.121 void cob\_subswp\_s32\_binary ( unsigned char \*, const int )

7.31.2.122 void cob\_subswp\_s40\_binary ( unsigned char \*, const int )

7.31.2.123 void cob\_subswp\_s48\_binary ( unsigned char \*, const int )

7.31.2.124 void cob\_subswp\_s56\_binary ( unsigned char \*, const int )

7.31.2.125 void cob\_subswp\_s64\_binary ( unsigned char \*, const int )

7.31.2.126 void cob\_subswp\_u16\_binary ( unsigned char \*, const int )

7.31.2.127 void cob\_subswp\_u24\_binary ( unsigned char \*, const int )

7.31.2.128 void cob\_subswp\_u32\_binary ( unsigned char \*, const int )

7.31.2.129 void cob\_subswp\_u40\_binary ( unsigned char \*, const int )

7.31.2.130 void cob\_subswp\_u48\_binary ( unsigned char \*, const int )

7.31.2.131 void cob\_subswp\_u56\_binary ( unsigned char \*, const int )

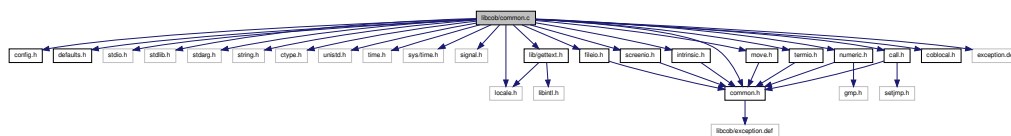


7.31.2.132 void cob\_subswp\_u64\_binary ( unsigned char \*, const int )

## 7.32 libcob/common.c File Reference

```
#include "config.h"
#include "defaults.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#include <signal.h>
#include <locale.h>
#include "common.h"
#include "coblocal.h"
#include "move.h"
#include "numeric.h"
#include "termio.h"
#include "fileio.h"
#include "call.h"
#include "screenio.h"
#include "intrinsic.h"
#include "lib/gettext.h"
#include "exception.def"
```

Include dependency graph for common.c:



## Classes

- struct [cob\\_exception](#)
- struct [cob\\_alloc\\_cache](#)
- struct [exit\\_handlerlist](#)
- struct [handlerlist](#)

## Defines

- #define [COB\\_ERRBUF\\_SIZE](#) 256
- #define [COB\\_EXCEPTION](#)(code, tag, [name](#), critical) [name](#),
- #define [COB\\_EXCEPTION](#)(code, tag, [name](#), critical) 0x##code,
- #define [EXCEPTION\\_TAB\\_SIZE](#) sizeof(cob\_exception\_tab\_code) / sizeof(int)

## Typedefs

- typedef void(\* [cob\\_sighandler\\_t](#))(int)

## Variables

- struct [cob\\_module](#) \* [cob\\_current\\_module](#) = NULL
- int [cob\\_initialized](#) = 0
- int [cob\\_exception\\_code](#) = 0
- int [cob\\_call\\_params](#) = 0
- int [cob\\_save\\_call\\_params](#) = 0
- int [cob\\_initial\\_external](#) = 0
- int [cob\\_got\\_exception](#) = 0
- const char \* [cob\\_orig\\_statement](#) = NULL
- const char \* [cob\\_orig\\_program\\_id](#) = NULL
- const char \* [cob\\_orig\\_section](#) = NULL
- const char \* [cob\\_orig\\_paragraph](#) = NULL
- unsigned int [cob\\_orig\\_line](#) = 0
- [cob\\_field](#) [cob\\_zero](#) = { 1, (ucharptr)"0", &all\_attr }
- [cob\\_field](#) [cob\\_space](#) = { 1, (ucharptr)" ", &all\_attr }
- [cob\\_field](#) [cob\\_high](#) = { 1, (ucharptr)"\xff", &all\_attr }
- [cob\\_field](#) [cob\\_low](#) = { 1, (ucharptr)"\0", &all\_attr }
- [cob\\_field](#) [cob\\_quote](#) = { 1, (ucharptr)"\"", &all\_attr }
- [cob\\_field](#) [cob\\_one](#) = { 1, (ucharptr)"1", &one\_attr }

### 7.32.1 Define Documentation

#### 7.32.1.1 #define COB\_ERRBUF\_SIZE 256

Definition at line 75 of file common.c.

7.32.1.2 `#define COB_EXCEPTION( code, tag, name, critical ) name,`

Definition at line 122 of file common.c.

7.32.1.3 `#define COB_EXCEPTION( code, tag, name, critical ) 0x##code,`

Definition at line 122 of file common.c.

7.32.1.4 `#define EXCEPTION_TAB_SIZE sizeof(cob_exception_tab_code) / sizeof(int)`

Definition at line 131 of file common.c.

## 7.32.2 Typedef Documentation

7.32.2.1 `typedef void(* cob_sighandler_t)(int)`

Definition at line 103 of file common.c.

## 7.32.3 Variable Documentation

7.32.3.1 `int cob_call_params = 0`

Definition at line 163 of file common.c.

7.32.3.2 `struct cob_module* cob_current_module = NULL`

Definition at line 158 of file common.c.

7.32.3.3 `int cob_exception_code = 0`

Definition at line 161 of file common.c.

7.32.3.4 `int cob_got_exception = 0`

Definition at line 166 of file common.c.

7.32.3.5 `cob_field cob_high = { 1, (ucharptr)"\xff", &all_attr }`

Definition at line 176 of file common.c.

**7.32.3.6 int cob\_initial\_external = 0**

Definition at line 165 of file common.c.

**7.32.3.7 int cob\_initialized = 0**

Definition at line 160 of file common.c.

**7.32.3.8 cob\_field cob\_low = { 1, (ucharptr)"\0", &all\_attr }**

Definition at line 177 of file common.c.

**7.32.3.9 cob\_field cob\_one = { 1, (ucharptr)"1", &one\_attr }**

Definition at line 179 of file common.c.

**7.32.3.10 unsigned int cob\_orig\_line = 0**

Definition at line 172 of file common.c.

**7.32.3.11 const char\* cob\_orig\_paragraph = NULL**

Definition at line 171 of file common.c.

**7.32.3.12 const char\* cob\_orig\_program\_id = NULL**

Definition at line 169 of file common.c.

**7.32.3.13 const char\* cob\_orig\_section = NULL**

Definition at line 170 of file common.c.

**7.32.3.14 const char\* cob\_orig\_statement = NULL**

Definition at line 168 of file common.c.

**7.32.3.15 cob\_field cob\_quote = { 1, (ucharptr)"\"", &all\_attr }**

Definition at line 178 of file common.c.

7.32.3.16 `int cob_save_call_params = 0`

Definition at line 164 of file common.c.

7.32.3.17 `cob_field cob_space = { 1, (ucharptr)" ", &all_attr }`

Definition at line 175 of file common.c.

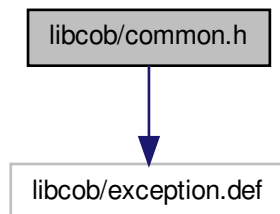
7.32.3.18 `cob_field cob_zero = { 1, (ucharptr)"0", &all_attr }`

Definition at line 174 of file common.c.

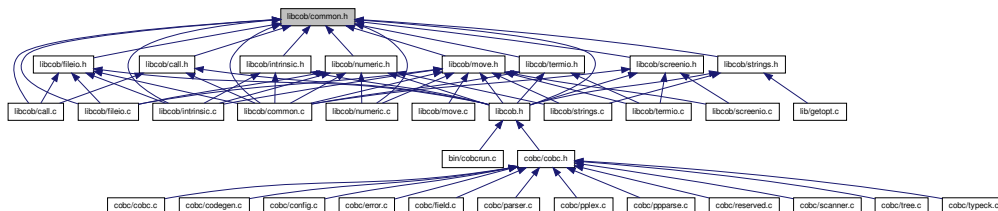
## 7.33 libcob/common.h File Reference

```
#include <libcob/exception.def>
```

Include dependency graph for common.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [cob\\_external](#)
- struct [cob\\_field\\_attr](#)
- struct [cob\\_field](#)
- struct [cob\\_module](#)

## Defines

- #define [DLL\\_EXPIMP](#)
- #define [COB\\_INLINE](#)
- #define [likely\(x\)](#) (x)
- #define [unlikely\(x\)](#) (x)
- #define [COB\\_NOINLINE](#)
- #define [COB\\_MINI\\_BUFF](#) 256
- #define [COB\\_SMALL\\_BUFF](#) 1024
- #define [COB\\_NORMAL\\_BUFF](#) 2048
- #define [COB\\_MEDIUM\\_BUFF](#) 8192
- #define [COB\\_LARGE\\_BUFF](#) 16384
- #define [COB\\_MINI\\_MAX](#) (COB\_MINI\_BUFF - 1)
- #define [COB\\_SMALL\\_MAX](#) (COB\_SMALL\_BUFF - 1)
- #define [COB\\_NORMAL\\_MAX](#) (COB\_NORMAL\_BUFF - 1)
- #define [COB\\_MEDIUM\\_MAX](#) (COB\_MEDIUM\_BUFF - 1)
- #define [COB\\_LARGE\\_MAX](#) (COB\_LARGE\_BUFF - 1)
- #define [COB\\_STACK\\_SIZE](#) 255
- #define [COB\\_MAX\\_FIELD\\_PARAMS](#) 64
- #define [COB\\_TYPE\\_UNKNOWN](#) 0x00
- #define [COB\\_TYPE\\_GROUP](#) 0x01
- #define [COB\\_TYPE\\_BOOLEAN](#) 0x02
- #define [COB\\_TYPE\\_NUMERIC](#) 0x10
- #define [COB\\_TYPE\\_NUMERIC\\_DISPLAY](#) 0x10
- #define [COB\\_TYPE\\_NUMERIC\\_BINARY](#) 0x11
- #define [COB\\_TYPE\\_NUMERIC\\_PACKED](#) 0x12
- #define [COB\\_TYPE\\_NUMERIC\\_FLOAT](#) 0x13
- #define [COB\\_TYPE\\_NUMERIC\\_DOUBLE](#) 0x14
- #define [COB\\_TYPE\\_NUMERIC\\_EDITED](#) 0x24
- #define [COB\\_TYPE\\_ALPHANUMERIC](#) 0x21
- #define [COB\\_TYPE\\_ALPHANUMERIC\\_ALL](#) 0x22
- #define [COB\\_TYPE\\_ALPHANUMERIC\\_EDITED](#) 0x23
- #define [COB\\_TYPE\\_NATIONAL](#) 0x40
- #define [COB\\_TYPE\\_NATIONAL\\_EDITED](#) 0x41
- #define [COB\\_FLAG\\_HAVE\\_SIGN](#) 0x01
- #define [COB\\_FLAG\\_SIGN\\_SEPARATE](#) 0x02
- #define [COB\\_FLAG\\_SIGN\\_LEADING](#) 0x04
- #define [COB\\_FLAG\\_BLANK\\_ZERO](#) 0x08
- #define [COB\\_FLAG\\_JUSTIFIED](#) 0x10

- #define `COB_FLAG_BINARY_SWAP` 0x20
- #define `COB_FLAG_REAL_BINARY` 0x40
- #define `COB_FLAG_IS_POINTER` 0x80
- #define `COB_FIELD_HAVE_SIGN`(f) ((f)->attr->flags & COB\_FLAG\_HAVE\_SIGN)
- #define `COB_FIELD_SIGN_SEPARATE`(f) ((f)->attr->flags & COB\_FLAG\_SIGN\_SEPARATE)
- #define `COB_FIELD_SIGN_LEADING`(f) ((f)->attr->flags & COB\_FLAG\_SIGN\_LEADING)
- #define `COB_FIELD_BLANK_ZERO`(f) ((f)->attr->flags & COB\_FLAG\_BLANK\_ZERO)
- #define `COB_FIELD_JUSTIFIED`(f) ((f)->attr->flags & COB\_FLAG\_JUSTIFIED)
- #define `COB_FIELD_BINARY_SWAP`(f) ((f)->attr->flags & COB\_FLAG\_BINARY\_SWAP)
- #define `COB_FIELD_REAL_BINARY`(f) ((f)->attr->flags & COB\_FLAG\_REAL\_BINARY)
- #define `COB_FIELD_IS_POINTER`(f) ((f)->attr->flags & COB\_FLAG\_IS\_POINTER)
- #define `cob_get_sign`(f) (COB\_FIELD\_HAVE\_SIGN (f) ? cob\_real\_get\_sign (f) : 0)
- #define `cob_put_sign`(f, s) if (COB\_FIELD\_HAVE\_SIGN (f)) cob\_real\_put\_sign (f, s)
- #define `COB_FIELD_TYPE`(f) ((f)->attr->type)
- #define `COB_FIELD_DIGITS`(f) ((f)->attr->digits)
- #define `COB_FIELD_SCALE`(f) ((f)->attr->scale)
- #define `COB_FIELD_FLAGS`(f) ((f)->attr->flags)
- #define `COB_FIELD_PIC`(f) ((f)->attr->pic)
- #define `COB_FIELD_DATA`(f)
- #define `COB_FIELD_SIZE`(f) ((f)->size - (COB\_FIELD\_SIGN\_SEPARATE (f) ? 1 : 0))
- #define `COB_FIELD_IS_NUMERIC`(f) (COB\_FIELD\_TYPE (f) & COB\_TYPE\_NUMERIC)
- #define `GET_SIGN_ASCII`(x) x -= 0x40
- #define `PUT_SIGN_ASCII`(x) x += 0x40
- #define `COB_DISPLAY_SIGN_ASCII` 0
- #define `COB_DISPLAY_SIGN_EBCDIC` 1
- #define `COB_EXCEPTION`(code, tag, name, critical) tag,
- #define `COB_ERROR_INITIALIZED` 0
- #define `COB_ERROR_CODEGEN` 1
- #define `COB_ERROR_CHAINING` 2
- #define `COB_ERROR_STACK` 3
- #define `cob_d2i`(x) ((x) - '0')
- #define `cob_i2d`(x) ((x) + '0')

## Typedefs

- typedef unsigned char \* `ucharptr`

## Enumerations

- enum `cob_exception_id` { `COB_EC_ZERO`, `COB_EC_MAX` }

## Functions

- void `cob_init` (int, char \*\*)
- void `cob_module_enter` (struct `cob_module` \*)
- void `cob_module_leave` (struct `cob_module` \*)
- void `cobexit` (int)
- void `cob_stop_run` (const int)
- void `cob_fatal_error` (const unsigned int)
- void `cob_runtime_error` (const char \*,...)
- void \* `cob_malloc` (const size\_t)
- const char \* `cob_get_exception_name` (const int)
- void `cob_set_exception` (const int)
- void `cob_check_version` (const char \*, const char \*, const int)
- void `cob_accept_date` (`cob_field` \*)
- void `cob_accept_date_yyyymmdd` (`cob_field` \*)
- void `cob_accept_day` (`cob_field` \*)
- void `cob_accept_day_yyyddd` (`cob_field` \*)
- void `cob_accept_day_of_week` (`cob_field` \*)
- void `cob_accept_time` (`cob_field` \*)
- void `cob_display_command_line` (`cob_field` \*)
- void `cob_accept_command_line` (`cob_field` \*)
- void `cob_set_environment` (`cob_field` \*, `cob_field` \*)
- void `cob_display_environment` (`cob_field` \*)
- void `cob_get_environment` (`cob_field` \*, `cob_field` \*)
- void `cob_accept_environment` (`cob_field` \*)
- void `cob_display_env_value` (`cob_field` \*)
- void `cob_display_arg_number` (`cob_field` \*)
- void `cob_accept_arg_number` (`cob_field` \*)
- void `cob_accept_arg_value` (`cob_field` \*)
- void `cob_chain_setup` (void \*, const size\_t, const size\_t)
- void `cob_allocate` (unsigned char \*\*, `cob_field` \*, `cob_field` \*)
- void `cob_free_alloc` (unsigned char \*\*, unsigned char \*)
- int `cobinit` (void)
- int `cobtidy` (void)
- void \* `cobcommandline` (int, int \*, char \*\*\*, char \*\*\*, char \*\*)
- char \* `cobgetenv` (const char \*)
- int `cobputenv` (char \*)
- int `CBL_ERROR_PROC` (unsigned char \*, unsigned char \*)
- int `CBL_EXIT_PROC` (unsigned char \*, unsigned char \*)
- int `SYSTEM` (const unsigned char \*)
- int `CBL_AND` (unsigned char \*, unsigned char \*, const int)
- int `CBL_OR` (unsigned char \*, unsigned char \*, const int)



- int [CBL\\_NOR](#) (unsigned char \*, unsigned char \*, const int)
- int [CBL\\_XOR](#) (unsigned char \*, unsigned char \*, const int)
- int [CBL\\_IMP](#) (unsigned char \*, unsigned char \*, const int)
- int [CBL\\_NIMP](#) (unsigned char \*, unsigned char \*, const int)
- int [CBL\\_EQ](#) (unsigned char \*, unsigned char \*, const int)
- int [CBL\\_NOT](#) (unsigned char \*, const int)
- int [CBL\\_XF4](#) (unsigned char \*, unsigned char \*)
- int [CBL\\_XF5](#) (unsigned char \*, unsigned char \*)
- int [CBL\\_X91](#) (unsigned char \*, const unsigned char \*, unsigned char \*)
- int [CBL\\_TOUPPER](#) (unsigned char \*, const int)
- int [CBL\\_TOLOWER](#) (unsigned char \*, const int)
- int [CBL\\_OC\\_NANOSLEEP](#) (unsigned char \*)
- int [cob\\_return\\_args](#) (unsigned char \*)
- int [cob\\_parameter\\_size](#) (unsigned char \*)
- int [cob\\_acuw\\_sleep](#) (unsigned char \*)
- int [cob\\_acuw\\_justify](#) (unsigned char \*,...)
- unsigned char \* [cob\\_external\\_addr](#) (const char \*, const int)
- unsigned char \* [cob\\_get\\_pointer](#) (const unsigned char \*)
- void \* [cob\\_get\\_prog\\_pointer](#) (const unsigned char \*)
- void [cob\\_set\\_location](#) (const char \*, const char \*, const unsigned int, const char \*, const char \*, const char \*)
- void [cob\\_ready\\_trace](#) (void)
- void [cob\\_reset\\_trace](#) (void)
- int [cob\\_get\\_switch](#) (const int)
- void [cob\\_set\\_switch](#) (const int, const int)
- int [cob\\_cmp](#) ([cob\\_field](#) \*, [cob\\_field](#) \*)
- int [cob\\_is\\_omitted](#) (const [cob\\_field](#) \*)
- int [cob\\_is\\_numeric](#) ([cob\\_field](#) \*)
- int [cob\\_is\\_alpha](#) (const [cob\\_field](#) \*)
- int [cob\\_is\\_upper](#) (const [cob\\_field](#) \*)
- int [cob\\_is\\_lower](#) (const [cob\\_field](#) \*)
- void [cob\\_table\\_sort\\_init](#) (const int, const unsigned char \*)
- void [cob\\_table\\_sort\\_init\\_key](#) (const int, [cob\\_field](#) \*, size\_t)
- void [cob\\_table\\_sort](#) ([cob\\_field](#) \*, const int)
- void [cob\\_check\\_numeric](#) ([cob\\_field](#) \*, const char \*)
- void [cob\\_check\\_based](#) (const unsigned char \*, const char \*)
- void [cob\\_check\\_odo](#) (const int, const int, const int, const char \*)
- void [cob\\_check\\_subscript](#) (const int, const int, const int, const char \*)
- void [cob\\_check\\_ref\\_mod](#) (const int, const int, const int, const char \*)
- int [cob\\_numeric\\_cmp](#) ([cob\\_field](#) \*, [cob\\_field](#) \*)

## Variables

- DLL\_EXPIMP int [cob\\_initialized](#)
- DLL\_EXPIMP int [cob\\_exception\\_code](#)
- DLL\_EXPIMP struct [cob\\_module](#) \* [cob\\_current\\_module](#)
- DLL\_EXPIMP int [cob\\_call\\_params](#)
- DLL\_EXPIMP int [cob\\_save\\_call\\_params](#)
- DLL\_EXPIMP int [cob\\_initial\\_external](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_zero](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_space](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_high](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_low](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_quote](#)
- DLL\_EXPIMP [cob\\_field](#) [cob\\_one](#)

### 7.33.1 Define Documentation

7.33.1.1 `#define cob_d2i( x ) ((x) - '0')`

Definition at line 263 of file common.h.

7.33.1.2 `#define COB_DISPLAY_SIGN_ASCII 0`

Definition at line 194 of file common.h.

7.33.1.3 `#define COB_DISPLAY_SIGN_EBCDIC 1`

Definition at line 195 of file common.h.

7.33.1.4 `#define COB_EXCEPTION( code, tag, name, critical ) tag,`

Definition at line 223 of file common.h.

7.33.1.5 `#define COB_ERROR_CHAINING 2`

Definition at line 239 of file common.h.

7.33.1.6 `#define COB_ERROR_CODEGEN 1`

Definition at line 238 of file common.h.

7.33.1.7 `#define COB_ERROR_INITIALIZED 0`

Definition at line 237 of file common.h.

**7.33.1.8 #define COB\_FERROR\_STACK 3**

Definition at line 240 of file common.h.

**7.33.1.9 #define COB\_FIELD\_BINARY\_SWAP( f ) ((f)->attr->flags & COB\_FLAG\_BINARY\_SWAP)**

Definition at line 146 of file common.h.

**7.33.1.10 #define COB\_FIELD\_BLANK\_ZERO( f ) ((f)->attr->flags & COB\_FLAG\_BLANK\_ZERO)**

Definition at line 144 of file common.h.

**7.33.1.11 #define COB\_FIELD\_DATA( f )****Value:**

```
((f)->data +  
  ((COB_FIELD_SIGN_SEPARATE (f) && COB_FIELD_SIGN_LEADING (f)) ? 1 : 0))
```

Definition at line 176 of file common.h.

**7.33.1.12 #define COB\_FIELD\_DIGITS( f ) ((f)->attr->digits)**

Definition at line 172 of file common.h.

**7.33.1.13 #define COB\_FIELD\_FLAGS( f ) ((f)->attr->flags)**

Definition at line 174 of file common.h.

**7.33.1.14 #define COB\_FIELD\_HAVE\_SIGN( f ) ((f)->attr->flags & COB\_FLAG\_HAVE\_SIGN)**

Definition at line 141 of file common.h.

**7.33.1.15 #define COB\_FIELD\_IS\_NUMERIC( f ) (COB\_FIELD\_TYPE (f) & COB\_TYPE\_NUMERIC)**

Definition at line 182 of file common.h.

**7.33.1.16 #define COB\_FIELD\_IS\_POINTER( f ) ((f)->attr->flags & COB\_FLAG\_IS\_POINTER)**

Definition at line 148 of file common.h.

**7.33.1.17 #define COB\_FIELD\_JUSTIFIED( f ) ((f)->attr->flags & COB\_FLAG\_JUSTIFIED)**

Definition at line 145 of file common.h.

7.33.1.18 `#define COB_FIELD_PIC( f ) ((f)->attr->pic)`

Definition at line 175 of file common.h.

7.33.1.19 `#define COB_FIELD_REAL_BINARY( f ) ((f)->attr->flags & COB_FLAG_REAL_BINARY)`

Definition at line 147 of file common.h.

7.33.1.20 `#define COB_FIELD_SCALE( f ) ((f)->attr->scale)`

Definition at line 173 of file common.h.

7.33.1.21 `#define COB_FIELD_SIGN_LEADING( f ) ((f)->attr->flags & COB_FLAG_SIGN_LEADING)`

Definition at line 143 of file common.h.

7.33.1.22 `#define COB_FIELD_SIGN_SEPARATE( f ) ((f)->attr->flags & COB_FLAG_SIGN_SEPARATE)`

Definition at line 142 of file common.h.

7.33.1.23 `#define COB_FIELD_SIZE( f ) ((f)->size - (COB_FIELD_SIGN_SEPARATE (f) ? 1 : 0))`

Definition at line 179 of file common.h.

7.33.1.24 `#define COB_FIELD_TYPE( f ) ((f)->attr->type)`

Definition at line 171 of file common.h.

7.33.1.25 `#define COB_FLAG_BINARY_SWAP 0x20`

Definition at line 137 of file common.h.

7.33.1.26 `#define COB_FLAG_BLANK_ZERO 0x08`

Definition at line 135 of file common.h.

7.33.1.27 `#define COB_FLAG_HAVE_SIGN 0x01`

Definition at line 132 of file common.h.

7.33.1.28 `#define COB_FLAG_IS_POINTER 0x80`

Definition at line 139 of file common.h.

7.33.1.29 `#define COB_FLAG_JUSTIFIED 0x10`

Definition at line 136 of file common.h.

7.33.1.30 `#define COB_FLAG_REAL_BINARY 0x40`

Definition at line 138 of file common.h.

7.33.1.31 `#define COB_FLAG_SIGN_LEADING 0x04`

Definition at line 134 of file common.h.

7.33.1.32 `#define COB_FLAG_SIGN_SEPARATE 0x02`

Definition at line 133 of file common.h.

7.33.1.33 `#define cob_get_sign( f ) (COB_FIELD_HAVE_SIGN (f) ? cob_real_get_sign (f) : 0)`

Definition at line 150 of file common.h.

7.33.1.34 `#define cob_i2d( x ) ((x) + '0')`

Definition at line 266 of file common.h.

7.33.1.35 `#define COB_INLINE`

Definition at line 54 of file common.h.

7.33.1.36 `#define COB_LARGE_BUFF 16384`

Definition at line 83 of file common.h.

7.33.1.37 `#define COB_LARGE_MAX (COB_LARGE_BUFF - 1)`

Definition at line 88 of file common.h.

7.33.1.38 `#define COB_MAX_FIELD_PARAMS 64`

Definition at line 92 of file common.h.

7.33.1.39 `#define COB_MEDIUM_BUFF 8192`

Definition at line 82 of file common.h.

7.33.1.40 `#define COB_MEDIUM_MAX (COB_MEDIUM_BUFF - 1)`

Definition at line 87 of file common.h.

7.33.1.41 `#define COB_MINI_BUFF 256`

Definition at line 79 of file common.h.

7.33.1.42 `#define COB_MINI_MAX (COB_MINI_BUFF - 1)`

Definition at line 84 of file common.h.

7.33.1.43 `#define COB_NOINLINE`

Definition at line 70 of file common.h.

7.33.1.44 `#define COB_NORMAL_BUFF 2048`

Definition at line 81 of file common.h.

7.33.1.45 `#define COB_NORMAL_MAX (COB_NORMAL_BUFF - 1)`

Definition at line 86 of file common.h.

7.33.1.46 `#define cob_put_sign( f, s ) if (COB_FIELD_HAVE_SIGN (f)) cob_real_put_sign (f, s)`

Definition at line 151 of file common.h.

7.33.1.47 `#define COB_SMALL_BUFF 1024`

Definition at line 80 of file common.h.

7.33.1.48 `#define COB_SMALL_MAX (COB_SMALL_BUFF - 1)`

Definition at line 85 of file common.h.

7.33.1.49 `#define COB_STACK_SIZE 255`

Definition at line 90 of file common.h.

7.33.1.50 `#define COB_TYPE_ALPHANUMERIC 0x21`

Definition at line 123 of file common.h.

7.33.1.51 `#define COB_TYPE_ALPHANUMERIC_ALL 0x22`

Definition at line 124 of file common.h.

7.33.1.52 `#define COB_TYPE_ALPHANUMERIC_EDITED 0x23`

Definition at line 125 of file common.h.

7.33.1.53 `#define COB_TYPE_BOOLEAN 0x02`

Definition at line 113 of file common.h.

7.33.1.54 `#define COB_TYPE_GROUP 0x01`

Definition at line 112 of file common.h.

7.33.1.55 `#define COB_TYPE_NATIONAL 0x40`

Definition at line 127 of file common.h.

7.33.1.56 `#define COB_TYPE_NATIONAL_EDITED 0x41`

Definition at line 128 of file common.h.

7.33.1.57 `#define COB_TYPE_NUMERIC 0x10`

Definition at line 115 of file common.h.

7.33.1.58 `#define COB_TYPE_NUMERIC_BINARY 0x11`

Definition at line 117 of file common.h.

7.33.1.59 `#define COB_TYPE_NUMERIC_DISPLAY 0x10`

Definition at line 116 of file common.h.

7.33.1.60 `#define COB_TYPE_NUMERIC_DOUBLE 0x14`

Definition at line 120 of file common.h.

7.33.1.61 `#define COB_TYPE_NUMERIC_EDITED 0x24`

Definition at line 121 of file common.h.

7.33.1.62 `#define COB_TYPE_NUMERIC_FLOAT 0x13`

Definition at line 119 of file common.h.

7.33.1.63 `#define COB_TYPE_NUMERIC_PACKED 0x12`

Definition at line 118 of file common.h.

7.33.1.64 `#define COB_TYPE_UNKNOWN 0x00`

Definition at line 111 of file common.h.

7.33.1.65 `#define DLL_EXPIMP`

Definition at line 46 of file common.h.

7.33.1.66 `#define GET_SIGN_ASCII( x ) x -= 0x40`

Definition at line 191 of file common.h.

7.33.1.67 `#define likely( x ) (x)`

Definition at line 68 of file common.h.



7.33.1.68 `#define PUT_SIGN_ASCII( x ) x += 0x40`

Definition at line 192 of file common.h.

7.33.1.69 `#define unlikely( x )(x)`

Definition at line 69 of file common.h.

### 7.33.2 Typedef Documentation

7.33.2.1 `typedef unsigned char* ucharptr`

Definition at line 77 of file common.h.

### 7.33.3 Enumeration Type Documentation

7.33.3.1 `enum cob_exception_id`

Enumerator:

**`COB_EC_ZERO`**

**`COB_EC_MAX`**

Definition at line 225 of file common.h.

```
        {  
            COB_EC_ZERO,  
#include <libcob/exception.def>  
            COB_EC_MAX  
};
```

### 7.33.4 Function Documentation

7.33.4.1 `int CBL_AND ( unsigned char *, unsigned char *, const int )`

7.33.4.2 `int CBL_EQ ( unsigned char *, unsigned char *, const int )`

7.33.4.3 `int CBL_ERROR_PROC ( unsigned char *, unsigned char * )`

7.33.4.4 `int CBL_EXIT_PROC ( unsigned char *, unsigned char * )`

7.33.4.5 `int CBL_IMP ( unsigned char *, unsigned char *, const int )`

7.33.4.6 `int CBL_NIMP ( unsigned char *, unsigned char *, const int )`

7.33.4.7 `int CBL_NOR ( unsigned char *, unsigned char *, const int )`

- 7.33.4.8 int CBL\_NOT ( unsigned char \* , const int )
- 7.33.4.9 int CBL\_OC\_NANOSLEEP ( unsigned char \* )
- 7.33.4.10 int CBL\_OR ( unsigned char \* , unsigned char \* , const int )
- 7.33.4.11 int CBL\_TOLOWER ( unsigned char \* , const int )
- 7.33.4.12 int CBL\_Toupper ( unsigned char \* , const int )
- 7.33.4.13 int CBL\_X91 ( unsigned char \* , const unsigned char \* , unsigned char \* )
- 7.33.4.14 int CBL\_XF4 ( unsigned char \* , unsigned char \* )
- 7.33.4.15 int CBL\_XF5 ( unsigned char \* , unsigned char \* )
- 7.33.4.16 int CBL\_XOR ( unsigned char \* , unsigned char \* , const int )
- 7.33.4.17 void cob\_accept\_arg\_number ( cob\_field \* )
- 7.33.4.18 void cob\_accept\_arg\_value ( cob\_field \* )
- 7.33.4.19 void cob\_accept\_command\_line ( cob\_field \* )
- 7.33.4.20 void cob\_accept\_date ( cob\_field \* )
- 7.33.4.21 void cob\_accept\_date\_yyyymmdd ( cob\_field \* )
- 7.33.4.22 void cob\_accept\_day ( cob\_field \* )
- 7.33.4.23 void cob\_accept\_day\_of\_week ( cob\_field \* )
- 7.33.4.24 void cob\_accept\_day\_yyyddd ( cob\_field \* )
- 7.33.4.25 void cob\_accept\_environment ( cob\_field \* )
- 7.33.4.26 void cob\_accept\_time ( cob\_field \* )
- 7.33.4.27 int cob\_acuw\_justify ( unsigned char \* , ... )
- 7.33.4.28 int cob\_acuw\_sleep ( unsigned char \* )
- 7.33.4.29 void cob\_allocate ( unsigned char \*\* , cob\_field \* , cob\_field \* )
- 7.33.4.30 void cob\_chain\_setup ( void \* , const size\_t , const size\_t )
- 7.33.4.31 void cob\_check\_based ( const unsigned char \* , const char \* )

- 7.33.4.32 void cob\_check\_numeric ( cob\_field \*, const char \* )
- 7.33.4.33 void cob\_check\_odo ( const int , const int , const int , const char \* )
- 7.33.4.34 void cob\_check\_ref\_mod ( const int , const int , const int , const char \* )
- 7.33.4.35 void cob\_check\_subscript ( const int , const int , const int , const char \* )
- 7.33.4.36 void cob\_check\_version ( const char \* , const char \* , const int )
- 7.33.4.37 int cob\_cmp ( cob\_field \* , cob\_field \* )
- 7.33.4.38 void cob\_display\_arg\_number ( cob\_field \* )
- 7.33.4.39 void cob\_display\_command\_line ( cob\_field \* )
- 7.33.4.40 void cob\_display\_env\_value ( cob\_field \* )
- 7.33.4.41 void cob\_display\_environment ( cob\_field \* )
- 7.33.4.42 unsigned char\* cob\_external\_addr ( const char \* , const int )
- 7.33.4.43 void cob\_fatal\_error ( const unsigned int )
- 7.33.4.44 void cob\_free\_alloc ( unsigned char \*\*, unsigned char \* )
- 7.33.4.45 void cob\_get\_environment ( cob\_field \* , cob\_field \* )
- 7.33.4.46 const char\* cob\_get\_exception\_name ( const int )
- 7.33.4.47 unsigned char\* cob\_get\_pointer ( const unsigned char \* )
- 7.33.4.48 void\* cob\_get\_prog\_pointer ( const unsigned char \* )
- 7.33.4.49 int cob\_get\_switch ( const int )
- 7.33.4.50 void cob\_init ( int , char \*\* )
- 7.33.4.51 int cob\_is\_alpha ( const cob\_field \* )
- 7.33.4.52 int cob\_is\_lower ( const cob\_field \* )
- 7.33.4.53 int cob\_is\_numeric ( cob\_field \* )
- 7.33.4.54 int cob\_is\_omitted ( const cob\_field \* )
- 7.33.4.55 int cob\_is\_upper ( const cob\_field \* )

7.33.4.56 void\* cob\_malloc ( const size\_t )

7.33.4.57 void cob\_module\_enter ( struct cob\_module \* )

7.33.4.58 void cob\_module\_leave ( struct cob\_module \* )

7.33.4.59 int cob\_numeric\_cmp ( cob\_field \*, cob\_field \* )

Definition at line 1324 of file numeric.c.

```
{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}
```

7.33.4.60 int cob\_parameter\_size ( unsigned char \* )

7.33.4.61 void cob\_ready\_trace ( void )

7.33.4.62 void cob\_reset\_trace ( void )

7.33.4.63 int cob\_return\_args ( unsigned char \* )

7.33.4.64 void cob\_runtime\_error ( const char \*, ... )

7.33.4.65 void cob\_set\_environment ( cob\_field \*, cob\_field \* )

7.33.4.66 void cob\_set\_exception ( const int )

7.33.4.67 void cob\_set\_location ( const char \*, const char \*, const unsigned int, const char \*  
, const char \*, const char \* )

7.33.4.68 void cob\_set\_switch ( const int , const int )

7.33.4.69 void cob\_stop\_run ( const int )

7.33.4.70 void cob\_table\_sort ( cob\_field \*, const int )

7.33.4.71 void cob\_table\_sort\_init ( const int , const unsigned char \* )

7.33.4.72 void cob\_table\_sort\_init\_key ( const int , cob\_field \*, size\_t )

7.33.4.73 void\* cobcommandline ( int , int \*, char \*\*\*, char \*\*\*, char \*\* )

7.33.4.74 void cobexit ( int )

7.33.4.75 `char* cobgetenv ( const char * )`

7.33.4.76 `int cobinit ( void )`

7.33.4.77 `int cobputenv ( char * )`

7.33.4.78 `int cobtidy ( void )`

7.33.4.79 `int SYSTEM ( const unsigned char * )`

### 7.33.5 Variable Documentation

7.33.5.1 `DLL_EXPIMP int cob_call_params`

Definition at line 163 of file common.c.

7.33.5.2 `DLL_EXPIMP struct cob_module* cob_current_module`

Definition at line 158 of file common.c.

7.33.5.3 `DLL_EXPIMP int cob_exception_code`

Definition at line 161 of file common.c.

7.33.5.4 `DLL_EXPIMP cob_field cob_high`

Definition at line 176 of file common.c.

7.33.5.5 `DLL_EXPIMP int cob_initial_external`

Definition at line 165 of file common.c.

7.33.5.6 `DLL_EXPIMP int cob_initialized`

Definition at line 160 of file common.c.

7.33.5.7 `DLL_EXPIMP cob_field cob_low`

Definition at line 177 of file common.c.

7.33.5.8 `DLL_EXPIMP cob_field cob_one`

Definition at line 179 of file common.c.

### 7.33.5.9 DLL\_EXPIMP cob\_field cob\_quote

Definition at line 178 of file common.c.

### 7.33.5.10 DLL\_EXPIMP int cob\_save\_call\_params

Definition at line 164 of file common.c.

### 7.33.5.11 DLL\_EXPIMP cob\_field cob\_space

Definition at line 175 of file common.c.

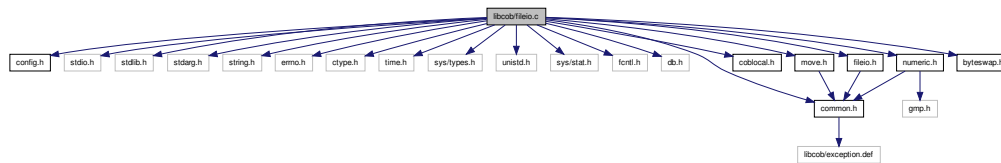
### 7.33.5.12 DLL\_EXPIMP cob\_field cob\_zero

Definition at line 174 of file common.c.

## 7.34 libcob/fileio.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <db.h>
#include "common.h"
#include "coblocal.h"
#include "move.h"
#include "numeric.h"
#include "fileio.h"
#include "byteswap.h"
```

Include dependency graph for fileio.c:



## Classes

- struct [cobitem](#)
- struct [memory\\_struct](#)
- struct [file\\_struct](#)
- struct [cobsort](#)
- struct [file\\_list](#)
- struct [indexed\\_file](#)

## Defines

- #define [\\_LFS64\\_LARGEFILE](#) 1
- #define [\\_LFS64\\_STDIO](#) 1
- #define [\\_FILE\\_OFFSET\\_BITS](#) 64
- #define [\\_LARGEFILE64\\_SOURCE](#) 1
- #define [SEEK\\_INIT](#)(f) fseek ((FILE \*)f->file, (off\_t)0, SEEK\_CUR)
- #define [O\\_BINARY](#) 0
- #define [O\\_LARGEFILE](#) 0
- #define [INITIAL\\_FLAGS](#) 0
- #define [COBSORTEND](#) 1
- #define [COBSORTABORT](#) 2
- #define [COBSORTFILEERR](#) 3
- #define [COBSORTNOTOPEN](#) 4
- #define [RETURN\\_STATUS](#)(x) do { save\_status (f, x, fnstatus); return; } while (0)
- #define [NUM\\_PREFIX](#) sizeof(prefix) / sizeof(char \*)
- #define [DB\\_PUT](#)(db, flags) db->put (db, NULL, &p->key, &p->data, flags)
- #define [DB\\_GET](#)(db, flags) db->get (db, NULL, &p->key, &p->data, flags)
- #define [DB\\_SEQ](#)(db, flags) db->c\_get (db, &p->key, &p->data, flags)
- #define [DB\\_DEL](#)(db, key, flags) db->del (db, NULL, key, flags)
- #define [DB\\_CLOSE](#)(db) db->close (db, 0)
- #define [DB\\_SYNC](#)(db) db->sync (db, 0)
- #define [cob\\_dbtsize\\_t](#) u\_int32\_t
- #define [DBT\\_SET](#)(key, fld)

## Functions

- void `cob_unlock_file` (`cob_file` \*f, `cob_field` \*fnstatus)
- void `cob_open` (`cob_file` \*f, const int mode, const int sharing, `cob_field` \*fnstatus)
- void `cob_close` (`cob_file` \*f, const int opt, `cob_field` \*fnstatus)
- void `cob_start` (`cob_file` \*f, const int cond, `cob_field` \*key, `cob_field` \*fnstatus)
- void `cob_read` (`cob_file` \*f, `cob_field` \*key, `cob_field` \*fnstatus, int read\_opts)
- void `cob_write` (`cob_file` \*f, `cob_field` \*rec, const int opt, `cob_field` \*fnstatus)
- void `cob_rewrite` (`cob_file` \*f, `cob_field` \*rec, const int opt, `cob_field` \*fnstatus)
- void `cob_delete` (`cob_file` \*f, `cob_field` \*fnstatus)
- void `cob_commit` (void)
- void `cob_rollback` (void)
- void `cob_default_error_handle` (void)
- void `cob_init_fileio` (void)
- void `cob_exit_fileio` (void)
- int `CBL_OPEN_FILE` (unsigned char \*file\_name, unsigned char \*file\_access, unsigned char \*file\_lock, unsigned char \*file\_dev, unsigned char \*file\_handle)
- int `CBL_CREATE_FILE` (unsigned char \*file\_name, unsigned char \*file\_access, unsigned char \*file\_lock, unsigned char \*file\_dev, unsigned char \*file\_handle)
- int `CBL_READ_FILE` (unsigned char \*file\_handle, unsigned char \*file\_offset, unsigned char \*file\_len, unsigned char \*flags, unsigned char \*buf)
- int `CBL_WRITE_FILE` (unsigned char \*file\_handle, unsigned char \*file\_offset, unsigned char \*file\_len, unsigned char \*flags, unsigned char \*buf)
- int `CBL_CLOSE_FILE` (unsigned char \*file\_handle)
- int `CBL_FLUSH_FILE` (unsigned char \*file\_handle)
- int `CBL_DELETE_FILE` (unsigned char \*file\_name)
- int `CBL_COPY_FILE` (unsigned char \*fname1, unsigned char \*fname2)
- int `CBL_CHECK_FILE_EXIST` (unsigned char \*file\_name, unsigned char \*file\_info)
- int `CBL_RENAME_FILE` (unsigned char \*fname1, unsigned char \*fname2)
- int `CBL_GET_CURRENT_DIR` (const int flags, const int dir\_length, unsigned char \*dir)
- int `CBL_CREATE_DIR` (unsigned char \*dir)
- int `CBL_CHANGE_DIR` (unsigned char \*dir)
- int `CBL_DELETE_DIR` (unsigned char \*dir)
- int `cob_acuw_mkdir` (unsigned char \*dir)
- int `cob_acuw_chdir` (unsigned char \*dir, unsigned char \*status)
- int `cob_acuw_copyfile` (unsigned char \*fname1, unsigned char \*fname2, unsigned char \*file\_type)
- int `cob_acuw_file_info` (unsigned char \*file\_name, unsigned char \*file\_info)
- int `cob_acuw_file_delete` (unsigned char \*file\_name, unsigned char \*file\_type)
- void `cob_file_sort_using` (`cob_file` \*sort\_file, `cob_file` \*data\_file)
- void `cob_file_sort_giving` (`cob_file` \*sort\_file, const size\_t varcnt,...)
- void `cob_file_sort_init` (`cob_file` \*f, const int nkeys, const unsigned char \*collating\_sequence, void \*sort\_return, `cob_field` \*fnstatus)
- void `cob_file_sort_init_key` (`cob_file` \*f, const int flag, `cob_field` \*field, size\_t offset)
- void `cob_file_sort_close` (`cob_file` \*f)
- void `cob_file_release` (`cob_file` \*f)
- void `cob_file_return` (`cob_file` \*f)



## Variables

- [cob\\_file](#) \* [cob\\_error\\_file](#)

### 7.34.1 Define Documentation

#### 7.34.1.1 #define \_FILE\_OFFSET\_BITS 64

Definition at line 25 of file fileio.c.

#### 7.34.1.2 #define \_LARGEFILE64\_SOURCE 1

Definition at line 26 of file fileio.c.

#### 7.34.1.3 #define \_LFS64\_LARGEFILE 1

Definition at line 23 of file fileio.c.

#### 7.34.1.4 #define \_LFS64\_STDIO 1

Definition at line 24 of file fileio.c.

#### 7.34.1.5 #define cob\_dbtsize\_t u\_int32\_t

Definition at line 288 of file fileio.c.

#### 7.34.1.6 #define COBSORTABORT 2

Definition at line 154 of file fileio.c.

#### 7.34.1.7 #define COBSORTEND 1

Definition at line 153 of file fileio.c.

#### 7.34.1.8 #define COBSORTFILEERR 3

Definition at line 155 of file fileio.c.

#### 7.34.1.9 #define COBSORTNOTOPEN 4

Definition at line 156 of file fileio.c.

7.34.1.10 `#define DB_CLOSE( db ) db->close (db, 0)`

Definition at line 286 of file fileio.c.

7.34.1.11 `#define DB_DEL( db, key, flags ) db->del (db, NULL, key, flags)`

Definition at line 285 of file fileio.c.

7.34.1.12 `#define DB_GET( db, flags ) db->get (db, NULL, &p->key, &p->data, flags)`

Definition at line 283 of file fileio.c.

7.34.1.13 `#define DB_PUT( db, flags ) db->put (db, NULL, &p->key, &p->data, flags)`

Definition at line 282 of file fileio.c.

7.34.1.14 `#define DB_SEQ( db, flags ) db->c_get (db, &p->key, &p->data, flags)`

Definition at line 284 of file fileio.c.

7.34.1.15 `#define DB_SYNC( db ) db->sync (db, 0)`

Definition at line 287 of file fileio.c.

7.34.1.16 `#define DBT_SET( key, fld )`

**Value:**

```
key.data = fld->data;
key.size = (cob_dbtsize_t) fld->size \
```

Definition at line 303 of file fileio.c.

7.34.1.17 `#define INITIAL_FLAGS 0`

Definition at line 148 of file fileio.c.

7.34.1.18 `#define NUM_PREFIX sizeof(prefix) / sizeof(char *)`

Definition at line 250 of file fileio.c.

7.34.1.19 `#define O_BINARY 0`

Definition at line 138 of file fileio.c.

#### 7.34.1.20 #define O\_LARGEFILE 0

Definition at line 142 of file fileio.c.

#### 7.34.1.21 #define RETURN\_STATUS( x ) do { save\_status (f, x, fnstatus); return; } while (0)

Definition at line 234 of file fileio.c.

#### 7.34.1.22 #define SEEK\_INIT( f ) fseek ((FILE \*)f->file, (off\_t)0, SEEK\_CUR)

Definition at line 132 of file fileio.c.

### 7.34.2 Function Documentation

#### 7.34.2.1 int CBL\_CHANGE\_DIR ( unsigned char \* *dir* )

Definition at line 4857 of file fileio.c.

```
{
    char    *fn;
    int     ret;

    COB_CHK_PARMS (CBL_CHANGE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = chdir (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}
```

#### 7.34.2.2 int CBL\_CHECK\_FILE\_EXIST ( unsigned char \* *file\_name*, unsigned char \* *file\_info* )

Definition at line 4722 of file fileio.c.

```
{
    char          *fn;
    struct tm     *tm;
    long long     sz;
    struct stat   st;
    short         y;
    char          d, m, hh, mm, ss;

    COB_CHK_PARMS (CBL_CHECK_FILE_EXIST, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
```

```

        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    if (stat (fn, &st) < 0) {
        free (fn);
        return 35;
    }
    free (fn);
    sz = st.st_size;
    tm = localtime (&st.st_mtime);
    d = (char) tm->tm_mday;
    m = (char) tm->tm_mon + 1;
    y = tm->tm_year + 1900;
    hh = (char) tm->tm_hour;
    mm = (char) tm->tm_min;
    ss = (char) tm->tm_sec;

#ifdef WORDS_BIGENDIAN
    sz = COB_BSWAP_64 (sz);
    y = COB_BSWAP_16 (y);
#endif
    memcpy (file_info, &sz, 8);
    file_info[8] = d;
    file_info[9] = m;
    memcpy (file_info+10, &y, 2);
    file_info[12] = hh;
    file_info[13] = mm;
    file_info[14] = ss;
    file_info[15] = 0;
    return 0;
}

```

### 7.34.2.3 int CBL\_CLOSE\_FILE ( unsigned char \* *file\_handle* )

Definition at line 4631 of file fileio.c.

```

{
    int    fd;

    COB_CHK_PARAMS (CBL_CLOSE_FILE, 1);

    memcpy (&fd, file_handle, 4);
    return close (fd);
}

```

### 7.34.2.4 int CBL\_COPY\_FILE ( unsigned char \* *fname1*, unsigned char \* *fname2* )

Definition at line 4670 of file fileio.c.

```

{
    char    *fn1;
    char    *fn2;
#ifdef O_BINARY
    int    flag = O_BINARY;
#else

```

```

        int      flag = 0;
#endif
    int      ret;
    int      i;
    int      fd1, fd2;

    COB_CHK_PARMS (CBL_COPY_FILE, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    if (!cob_current_module->cob_procedure_parameters[1]) {
        return -1;
    }
    fn1 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);

    flag |= O_RDONLY;
    fd1 = open (fn1, flag, 0);
    if (fd1 < 0) {
        free (fn1);
        return -1;
    }
    free (fn1);
    fn2 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[1]);

    flag &= ~O_RDONLY;
    flag |= O_CREAT | O_TRUNC | O_WRONLY;
    fd2 = open (fn2, flag, 0660);
    if (fd2 < 0) {
        close (fd1);
        free (fn2);
        return -1;
    }
    free (fn2);
    ret = 0;
    while ((i = read (fd1, fn1, sizeof(fn1))) > 0) {
        if (write (fd2, fn1, (size_t)i) < 0) {
            ret = -1;
            break;
        }
    }
    close (fd1);
    close (fd2);
    return ret;
}

```

#### 7.34.2.5 int CBL.CREATE\_DIR ( unsigned char \* *dir* )

Definition at line 4833 of file fileio.c.

```

{
    char      *fn;
    int      ret;

    COB_CHK_PARMS (CBL_CREATE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
}

```

```

        fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
#ifdef _WIN32
    ret = mkdir (fn);
#else
    ret = mkdir (fn, 0770);
#endif
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}

```

#### 7.34.2.6 int CBL\_CREATE\_FILE ( unsigned char \* *file\_name*, unsigned char \* *file\_access*, unsigned char \* *file\_lock*, unsigned char \* *file\_dev*, unsigned char \* *file\_handle* )

Definition at line 4548 of file fileio.c.

```

{
    COB_CHK_PARAMS (CBL_CREATE_FILE, 5);

    return open_cbl_file (file_name, file_access, file_handle, O_CREAT | O_TR
UNC);
}

```

#### 7.34.2.7 int CBL\_DELETE\_DIR ( unsigned char \* *dir* )

Definition at line 4877 of file fileio.c.

```

{
    char    *fn;
    int     ret;

    COB_CHK_PARAMS (CBL_DELETE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = rmdir (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}

```

#### 7.34.2.8 int CBL\_DELETE\_FILE ( unsigned char \* *file\_name* )

Definition at line 4650 of file fileio.c.

```

{
    char    *fn;
    int     ret;

    COB_CHK_PARMS (CBL_DELETE_FILE, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = unlink (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}

```

#### 7.34.2.9 int CBL\_FLUSH\_FILE ( unsigned char \* *file\_handle* )

Definition at line 4642 of file fileio.c.

```

{
    COB_CHK_PARMS (CBL_FLUSH_FILE, 1);

    return 0;
}

```

#### 7.34.2.10 int CBL\_GET\_CURRENT\_DIR ( const int *flags*, const int *dir\_length*, unsigned char \* *dir* )

Definition at line 4793 of file fileio.c.

```

{
    char    *dirname;
    int     dir_size;
    int     has_space;

    COB_CHK_PARMS (CBL_GET_CURRENT_DIR, 3);

    if (dir_length < 1) {
        return 128;
    }
    if (flags) {
        return 129;
    }
    memset (dir, ' ', (size_t)dir_length);
    dirname = getcwd (NULL, 0);
    if (dirname == NULL) {
        return 128;
    }
    dir_size = (int) strlen (dirname);
    has_space = 0;
    if (strchr (dirname, ' ')) {
        has_space = 2;
    }
}

```

```

}
if (dir_size + has_space > dir_length) {
    free (dirname);
    return 128;
}
if (has_space) {
    *dir = '\0';
    memcpy (&dir[1], dirname, (size_t)dir_size);
    dir[dir_size + 1] = '\0';
} else {
    memcpy (dir, dirname, (size_t)dir_size);
}
free (dirname);
return 0;
}

```

#### 7.34.2.11 int CBL\_OPEN\_FILE ( unsigned char \* *file\_name*, unsigned char \* *file\_access*, unsigned char \* *file\_lock*, unsigned char \* *file\_dev*, unsigned char \* *file\_handle* )

Definition at line 4538 of file fileio.c.

```

{
    COB_CHK_PARAMS (CBL_OPEN_FILE, 5);

    return open_cbl_file (file_name, file_access, file_handle, 0);
}

```

#### 7.34.2.12 int CBL\_READ\_FILE ( unsigned char \* *file\_handle*, unsigned char \* *file\_offset*, unsigned char \* *file\_len*, unsigned char \* *flags*, unsigned char \* *buf* )

Definition at line 4558 of file fileio.c.

```

{
    long long    off;
    int          fd;
    int          len;
    int          rc = 0;
    struct stat  st;

    COB_CHK_PARAMS (CBL_READ_FILE, 5);

    memcpy (&fd, file_handle, 4);
    memcpy (&off, file_offset, 8);
    memcpy (&len, file_len, 4);
#ifdef WORDS_BIGENDIAN
    off = COB_BSWAP_64 (off);
    len = COB_BSWAP_32 (len);
#endif
    if (lseek (fd, (off_t)off, SEEK_SET) < 0) {
        return -1;
    }
    if (len > 0) {
        rc = read (fd, buf, (size_t)len);
        if (rc < 0) {
            rc = -1;
        }
    }
}

```



```

        } else if (rc == 0) {
            rc = 10;
        } else {
            rc = 0;
        }
    }
    if ((*flags & 0x80) != 0) {
        if (fstat (fd, &st) < 0) {
            return -1;
        }
        off = st.st_size;
#ifdef WORDS_BIGENDIAN
        off = COB_BSWAP_64 (off);
#endif
        memcpy (file_offset, &off, 8);
    }
    return rc;
}

```

#### 7.34.2.13 int CBL\_RENAME\_FILE ( unsigned char \* *fname1*, unsigned char \* *fname2* )

Definition at line 4767 of file fileio.c.

```

{
    char    *fn1;
    char    *fn2;
    int     ret;

    COB_CHK_PARAMS (CBL_RENAME_FILE, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    if (!cob_current_module->cob_procedure_parameters[1]) {
        return -1;
    }
    fn1 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    fn2 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[1]);

    ret = rename (fn1, fn2);
    free (fn1);
    free (fn2);
    if (ret) {
        return 128;
    }
    return 0;
}

```

#### 7.34.2.14 int CBL\_WRITE\_FILE ( unsigned char \* *file\_handle*, unsigned char \* *file\_offset*, unsigned char \* *file\_len*, unsigned char \* *flags*, unsigned char \* *buf* )

Definition at line 4603 of file fileio.c.

```

{

```

```

    long long    off;
    int          fd;
    int          len;
    int          rc;

    COB_CHK_PARAMS (CBL_WRITE_FILE, 5);

    memcpy (&fd, file_handle, 4);
    memcpy (&off, file_offset, 8);
    memcpy (&len, file_len, 4);
#ifdef WORDS_BIGENDIAN
    off = COB_BSWAP_64 (off);
    len = COB_BSWAP_32 (len);
#endif
    if (lseek (fd, (off_t)off, SEEK_SET) < 0) {
        return -1;
    }
    rc = write (fd, buf, (size_t)len);
    if (rc < 0) {
        return 30;
    }
    return 0;
}

```

#### 7.34.2.15 int cob\_acuw\_chdir ( unsigned char \* *dir*, unsigned char \* *status* )

Definition at line 4911 of file fileio.c.

```

{
    int          ret;

    COB_CHK_PARAMS (C$CHDIR, 2);

    ret = CBL_CHANGE_DIR (dir);
    if (ret < 0) {
        ret = 128;
    }
    cob_set_int (cob_current_module->cob_procedure_parameters[1], ret);
    return ret;
}

```

#### 7.34.2.16 int cob\_acuw\_copyfile ( unsigned char \* *fname1*, unsigned char \* *fname2*, unsigned char \* *file\_type* )

Definition at line 4926 of file fileio.c.

```

{
    int          ret = 128;

    /* RXW - Type is not yet evaluated */

    COB_CHK_PARAMS (C$COPY, 3);

    if (cob_call_params < 3) {
        return 128;
    }
}

```

```

    }
    ret = CBL_COPY_FILE (fname1, fname2);
    if (ret < 0) {
        ret = 128;
    }
    return ret;
}

```

#### 7.34.2.17 int cob\_acuw\_file\_delete ( unsigned char \* *file\_name*, unsigned char \* *file\_type* )

Definition at line 4993 of file fileio.c.

```

{
    int    ret;

    /* RXW - Type is not yet evaluated */
    COB_CHK_PARMS (C$DELETE, 2);

    if (cob_call_params < 2 || !cob_current_module->cob_procedure_parameters[
0]) {
        return 128;
    }
    ret = CBL_DELETE_FILE (file_name);
    if (ret < 0) {
        ret = 128;
    }
    return ret;
}

```

#### 7.34.2.18 int cob\_acuw\_file\_info ( unsigned char \* *file\_name*, unsigned char \* *file\_info* )

Definition at line 4945 of file fileio.c.

```

{
    char                *fn;
    struct tm           *tm;
    unsigned long long  sz;
    unsigned int        dt;
    short               Y;
    short               d, m, hh, mm, ss;
    struct stat         st;

    COB_CHK_PARMS (C$FILEINFO, 2);

    if (cob_call_params < 2 || !cob_current_module->cob_procedure_parameters[
0]) {
        return 128;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    if (stat (fn, &st) < 0) {
        free (fn);
        return 35;
    }
    free (fn);
    sz = st.st_size;
}

```

```

        tm = localtime (&st.st_mtime);
        d = tm->tm_mday;
        m = tm->tm_mon + 1;
        y = tm->tm_year + 1900;
        hh = tm->tm_hour;
        mm = tm->tm_min;
        ss = tm->tm_sec;

#ifdef WORDS_BIGENDIAN
        sz = COB_BSWAP_64 (sz);
#endif
        memcpy (file_info, &sz, 8);
        dt = (y * 10000) + (m * 100) + d;
#ifdef WORDS_BIGENDIAN
        dt = COB_BSWAP_32 (dt);
#endif
        memcpy (file_info + 8, &dt, 4);
        dt = (hh * 1000000) + (mm * 10000) + (ss * 100);
#ifdef WORDS_BIGENDIAN
        dt = COB_BSWAP_32 (dt);
#endif
        memcpy (file_info + 12, &dt, 4);
        return 0;
}

```

#### 7.34.2.19 int cob\_acuw\_mkdir ( unsigned char \* dir )

Definition at line 4897 of file fileio.c.

```

{
    int          ret;

    COB_CHK_PARAMS (C$MAKEDIR, 1);

    ret = CBL_CREATE_DIR (dir);
    if (ret < 0) {
        ret = 128;
    }
    return ret;
}

```

#### 7.34.2.20 void cob\_close ( cob\_file \* f, const int opt, cob\_field \* fnstatus )

Definition at line 3987 of file fileio.c.

```

{
    int          ret;

    f->flag_read_done = 0;

    if (f->special) {
        f->open_mode = COB_OPEN_CLOSED;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    }
    if (f->open_mode == COB_OPEN_CLOSED) {

```

```

        RETURN_STATUS (COB_STATUS_42_NOT_OPEN);
    }

    if (f->flag_nonexistent) {
        ret = COB_STATUS_00_SUCCESS;
    } else {
#ifdef WITH_SEQRA_EXTFH
        if (f->organization != COB_ORG_INDEXED) {
            ret = extfh_cob_file_close (f, opt);
        } else {
#endif
            ret = fileio_funcs[(int)f->organization]->close (f, opt);

#ifdef WITH_SEQRA_EXTFH
        }
#endif
    }

    if (ret == COB_STATUS_00_SUCCESS) {
        switch (opt) {
            case COB_CLOSE_LOCK:
                f->open_mode = COB_OPEN_LOCKED;
                break;
            default:
                f->open_mode = COB_OPEN_CLOSED;
                break;
        }
    }

    RETURN_STATUS (ret);
}

```

#### 7.34.2.21 void cob\_commit ( void )

Definition at line 4273 of file fileio.c.

```

{
    struct file_list    *l;

    for (l = file_cache; l; l = l->next) {
        cob_file_unlock (l->file);
    }
}

```

#### 7.34.2.22 void cob\_default\_error\_handle ( void )

Definition at line 4293 of file fileio.c.

```

{
    const char    *msg;
    unsigned char *file_status;
    char          *filename;
    int           status;

    file_status = cob_error_file->file_status;
}

```

```
status = cob_d2i(file_status[0]) * 10 + cob_d2i(file_status[1]);
switch (status) {
case COB_STATUS_10_END_OF_FILE:
    msg = "End of file";
    break;
case COB_STATUS_14_OUT_OF_KEY_RANGE:
    msg = "Key out of range";
    break;
case COB_STATUS_21_KEY_INVALID:
    msg = "Key order not ascending";
    break;
case COB_STATUS_22_KEY_EXISTS:
    msg = "Record key already exists";
    break;
case COB_STATUS_23_KEY_NOT_EXISTS:
    msg = "Record key does not exist";
    break;
case COB_STATUS_30_PERMANENT_ERROR:
    msg = "Permanent file error";
    break;
case COB_STATUS_35_NOT_EXISTS:
    msg = "File does not exist";
    break;
case COB_STATUS_37_PERMISSION_DENIED:
    msg = "Permission denied";
    break;
case COB_STATUS_41_ALREADY_OPEN:
    msg = "File already open";
    break;
case COB_STATUS_42_NOT_OPEN:
    msg = "File not open";
    break;
case COB_STATUS_43_READ_NOT_DONE:
    msg = "READ must be executed first";
    break;
case COB_STATUS_44_RECORD_OVERFLOW:
    msg = "Record overflow";
    break;
case COB_STATUS_46_READ_ERROR:
    msg = "Failed to read";
    break;
case COB_STATUS_47_INPUT_DENIED:
    msg = "READ/START not allowed";
    break;
case COB_STATUS_48_OUTPUT_DENIED:
    msg = "WRITE not allowed";
    break;
case COB_STATUS_49_I_O_DENIED:
    msg = "DELETE/REWRITE not allowed";
    break;
case COB_STATUS_51_RECORD_LOCKED:
    msg = "Record locked by another file connector";
    break;
case COB_STATUS_52_EOP:
    msg = "A page overflow condition occurred";
    break;
case COB_STATUS_57_I_O_LINAGE:
    msg = "LINAGE values invalid";
    break;
case COB_STATUS_61_FILE_SHARING:
    msg = "File sharing conflict";
    break;
```

```

case COB_STATUS_91_NOT_AVAILABLE:
    msg = "Runtime library is not configured for this operation";
    break;
default:
    msg = "Unknown file error";
    break;
}

filename = cob_malloc (COB_MEDIUM_BUFF);
cob_field_to_string (cob_error_file->assign, filename);
cob_runtime_error ("%s (STATUS = %02d) File : '%s'", msg,
                  status, filename);
free (filename);
}

```

#### 7.34.2.23 void cob\_delete ( cob\_file \* f, cob\_field \* fnstatus )

Definition at line 4247 of file fileio.c.

```

{
    int    ret;
    int    read_done = f->flag_read_done;

    f->flag_read_done = 0;

    if (unlikely(f->open_mode == COB_OPEN_CLOSED ||
                f->open_mode != COB_OPEN_I_O)) {
        RETURN_STATUS (COB_STATUS_49_I_O_DENIED);
    }

    if (f->access_mode == COB_ACCESS_SEQUENTIAL && !read_done) {
        RETURN_STATUS (COB_STATUS_43_READ_NOT_DONE);
    }

    ret = fileio_funcs[(int)f->organization]->fdelete (f);

    if (unlikely(cob_do_sync && ret == 0)) {
        cob_sync (f, cob_do_sync);
    }

    RETURN_STATUS (ret);
}

```

#### 7.34.2.24 void cob\_exit\_fileio ( void )

Definition at line 4426 of file fileio.c.

```

{
    struct file_list    *l;

    for (l = file_cache; l; l = l->next) {
        if (l->file->open_mode != COB_OPEN_CLOSED &&
            l->file->open_mode != COB_OPEN_LOCKED) {
            cob_field_to_string (l->file->assign, runtime_buffer);
            cob_close (l->file, 0, NULL);
        }
    }
}

```

```

        fprintf (stderr, "WARNING - Implicit CLOSE of %s (%s)"
\n",
                l->file->select_name, runtime_buffer);
        fflush (stderr);
    }
}
#ifdef USE_DB41
free (record_lock_object);
if (bdb_env) {
    bdb_env->lock_id_free (bdb_env, bdb_lock_id);
    bdb_env->close (bdb_env, 0);
}
#endif
#if defined(WITH_INDEX_EXTFH) || defined(WITH_SEQRA_EXTFH)
extfh_cob_exit_fileio ();
#endif
}

```

#### 7.34.2.25 void cob\_file\_release ( cob\_file \* f )

Definition at line 5654 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field *fnstatus = NULL;
    int ret;

    hp = f->file;
    if (likely(hp)) {
        fnstatus = hp->fnstatus;
    }
    ret = cob_file_sort_submit (f, f->record->data);
    switch (ret) {
    case 0:
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
        break;
    default:
        if (likely(hp)) {
            *(int *) (hp->sort_return) = 16;
        }
        RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        break;
    }
}

```

#### 7.34.2.26 void cob\_file\_return ( cob\_file \* f )

Definition at line 5679 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field *fnstatus = NULL;
    int ret;

    hp = f->file;

```



```

    if (likely (hp)) {
        fnstatus = hp->fnstatus;
    }
    ret = cob_file_sort_retrieve (f, f->record->data);
    switch (ret) {
    case 0:
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
        break;
    case COBSORTEND:
        RETURN_STATUS (COB_STATUS_10_END_OF_FILE);
        break;
    default:
        if (likely (hp)) {
            *(int *) (hp->sort_return) = 16;
        }
        RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        break;
    }
}

```

#### 7.34.2.27 void cob\_file\_sort\_close ( cob\_file \* f )

Definition at line 5631 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field      *fnstatus = NULL;
    size_t        i;

    hp = f->file;
    if (likely (hp)) {
        fnstatus = hp->fnstatus;
        cob_free_list (hp->empty);
        for (i = 0; i < 4; i++) {
            cob_free_list (hp->queue[i].first);
            if (hp->file[i].fp != NULL) {
                fclose (hp->file[i].fp);
            }
        }
        free (hp);
    }
    f->file = NULL;
    RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

#### 7.34.2.28 void cob\_file\_sort\_giving ( cob\_file \* sort\_file, const size\_t varcnt, ... )

Definition at line 5544 of file fileio.c.

```

{
    cob_file      **fbase;
    struct cobsort *hp;
    size_t        i;
    int           ret;
    int           opt;
}

```

```

va_list      args;

fbase = cob_malloc (varcnt * sizeof(cob_file *));
va_start (args, varcnt);
for (i = 0; i < varcnt; i++) {
    fbase[i] = va_arg (args, cob_file *);
}
va_end (args);
for (i = 0; i < varcnt; i++) {
    cob_open (fbase[i], COB_OPEN_OUTPUT, 0, NULL);
}
while (1) {
    ret = cob_file_sort_retrieve (sort_file, sort_file->record->data)
;
    if (ret) {
        if (ret == COBSORTEND) {
            sort_file->file_status[0] = '1';
            sort_file->file_status[1] = '0';
        } else {
            hp = sort_file->file;
            *(int *) (hp->sort_return) = 16;
            sort_file->file_status[0] = '3';
            sort_file->file_status[1] = '0';
        }
        break;
    }
    for (i = 0; i < varcnt; i++) {
        if (fbase[i]->special ||
            fbase[i]->organization == COB_ORG_LINE_SEQUENTIAL) {
            opt = COB_WRITE_BEFORE | COB_WRITE_LINES | 1;
        } else {
            opt = 0;
        }
        cob_copy_check (fbase[i], sort_file);
        cob_write (fbase[i], fbase[i]->record, opt, NULL);
    }
}
for (i = 0; i < varcnt; i++) {
    cob_close (fbase[i], COB_CLOSE_NORMAL, NULL);
}
free (fbase);
}

```

#### 7.34.2.29 void cob\_file\_sort\_init ( cob\_file \* f, const int nkeys, const unsigned char \* collating\_sequence, void \* sort\_return, cob\_field \* fnstatus )

Definition at line 5594 of file fileio.c.

```

{
    struct cobsort *p;

    p = cob_malloc (sizeof (struct cobsort));
    p->fnstatus = fnstatus;
    p->size = f->record_max;
    p->r_size = f->record_max + sizeof (size_t);
    p->w_size = f->record_max + sizeof (size_t) + 1;
    p->pointer = f;
    p->sort_return = sort_return;
    *(int *) sort_return = 0;
}

```

```

p->memory = (size_t)cob_sort_memory / (p->size + sizeof(struct cobitem));

f->file = p;
f->keys = cob_malloc (sizeof (struct cob_file_key) * nkeys);
f->nkeys = 0;
if (collating_sequence) {
    f->sort_collating = collating_sequence;
} else {
    f->sort_collating = cob_current_module->collating_sequence;
}
RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

#### 7.34.2.30 void cob\_file\_sort\_init\_key ( cob\_file \* f, const int flag, cob\_field \* field, size\_t offset )

Definition at line 5621 of file fileio.c.

```

{
    f->keys[f->nkeys].flag = flag;
    f->keys[f->nkeys].field = field;
    f->keys[f->nkeys].offset = offset;
    f->nkeys++;
}

```

#### 7.34.2.31 void cob\_file\_sort\_using ( cob\_file \* sort\_file, cob\_file \* data\_file )

Definition at line 5524 of file fileio.c.

```

{
    int          ret;

    cob_open (data_file, COB_OPEN_INPUT, 0, NULL);
    while (1) {
        cob_read (data_file, NULL, NULL, COB_READ_NEXT);
        if (data_file->file_status[0] != '0') {
            break;
        }
        cob_copy_check (sort_file, data_file);
        ret = cob_file_sort_submit (sort_file, sort_file->record->data);
        if (ret) {
            break;
        }
    }
    cob_close (data_file, COB_CLOSE_NORMAL, NULL);
}

```

#### 7.34.2.32 void cob\_init\_fileio ( void )

Definition at line 4379 of file fileio.c.

```

{
    char    *s;
    int     n;

    if ((s = getenv ("COB_SYNC")) != NULL) {
        if (*s == 'Y' || *s == 'y') {
            cob_do_sync = 1;
        }
        if (*s == 'P' || *s == 'p') {
            cob_do_sync = 2;
        }
    }
    if ((s = getenv ("COB_SORT_MEMORY")) != NULL) {
        n = atoi (s);
        if (n >= 1024*1024) {
            cob_sort_memory = n;
        }
    }
    cob_file_path = getenv ("COB_FILE_PATH");
    if (cob_file_path) {
        if (!*cob_file_path || *cob_file_path == ' ') {
            cob_file_path = NULL;
        }
    }
    cob_ls_nulls = getenv ("COB_LS_NULLS");
    cob_ls_fixed = getenv ("COB_LS_FIXED");

    file_open_env = cob_malloc (COB_SMALL_BUFF);
    file_open_name = cob_malloc (COB_SMALL_BUFF);
    file_open_buff = cob_malloc (COB_SMALL_BUFF);

#ifdef USE_DB41
    bdb_home = getenv ("DB_HOME");
    join_environment ();
    record_lock_object = cob_malloc (1024);
    bdb_buff = cob_malloc (COB_SMALL_BUFF);
    rlo_size = 1024;
#endif

#ifdef WITH_INDEX_EXTFH || defined(WITH_SEQRA_EXTFH)
    extfh_cob_init_fileio (&sequential_funcs, &lineseq_funcs, &relative_funcs
, &cob_file_write_opt);
#endif
}

```

### 7.34.2.33 void cob\_open ( cob\_file \*f, const int mode, const int sharing, cob\_field \*fnstatus )

Definition at line 3734 of file fileio.c.

```

{
    char    *p;
    char    *src;
    char    *dst;
    size_t  i;
    size_t  simple;
    int     was_not_exist = 0;
    struct stat  st;

```

```

f->flag_read_done = 0;

/* file was previously closed with lock */
if (f->open_mode == COB_OPEN_LOCKED) {
    RETURN_STATUS (COB_STATUS_38_CLOSED_WITH_LOCK);
}

/* file is already open */
if (f->open_mode != COB_OPEN_CLOSED) {
    RETURN_STATUS (COB_STATUS_41_ALREADY_OPEN);
}

f->last_open_mode = mode;
f->flag_nonexistent = 0;
f->flag_end_of_file = 0;
f->flag_begin_of_file = 0;
f->flag_first_read = 2;

if (f->special) {
    if (f->special == 1) {
        if (mode != COB_OPEN_INPUT) {
            RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        }
        f->file = stdin;
        f->open_mode = mode;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    } else {
        if (mode != COB_OPEN_OUTPUT) {
            RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        }
        f->file = stdout;
        f->open_mode = mode;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    }
}

/* obtain the file name */
cob_field_to_string (f->assign, file_open_name);

#ifdef WITH_INDEX_EXTFH
if (f->organization == COB_ORG_INDEXED) {
    int    ret;

    ret = extfh_indexed_locate (f, file_open_name);
    switch (ret) {
        case COB_NOT_CONFIGURED:
            /* EXTFH requires OC to process the filename */
            break;
        case COB_STATUS_00_SUCCESS:
            /* EXTFH recognized the file */
            goto file_available;
        default:
            /* EXTFH detected an error */
            RETURN_STATUS (ret);
    }
}
#endif /* WITH_INDEX_EXTFH */

#ifdef WITH_SEQRA_EXTFH
if (f->organization != COB_ORG_INDEXED) {
    int    ret;

```

```

ret = extfh_seqra_locate (f, file_open_name);
switch (ret) {
case COB_NOT_CONFIGURED:
    /* EXT FH requires OC to process the filename */
    break;
case COB_STATUS_00_SUCCESS:
    /* EXT FH recognized the file */
    goto file_available;
default:
    /* EXT FH detected an error */
    RETURN_STATUS (ret);
}
}
#endif /* WITH_SEQRA_EXTFH */

if (cob_current_module->flag_filename_mapping) {
    src = file_open_name;
    dst = file_open_buff;
    simple = 1;
    /* expand environment variables */
    /* ex. "$TMPDIR/foo" -> "/tmp/foo" */
    while (*src) {
        if (!isalnum (*src) && *src != '_' && *src != '-') {
            simple = 0;
        }
        if (*src == '$') {
            for (i = 1; ; i++) {
                if (!isalnum (src[i]) && src[i] != '_' &&
                    *src != '-') {
                    break;
                }
            }
            memcpy (file_open_env, src + 1, i - 1);
            file_open_env[i - 1] = 0;
            if ((p = getenv (file_open_env)) != NULL) {
                strcpy (dst, p);
                dst += strlen (p);
            }
            src += i;
        } else {
            *dst++ = *src++;
        }
    }
    *dst = 0;
    strncpy (file_open_name, file_open_buff, COB_SMALL_MAX);

    /* resolve by environment variables */
    /* ex. "TMPFILE" -> DD_TMPFILE, dd_TMPFILE, or TMPFILE */
    if (simple) {
        for (i = 0; i < NUM_PREFIX; i++) {
            sprintf (file_open_buff, COB_SMALL_MAX, "%s%s",
                prefix[i], file_open_name);
            if ((p = getenv (file_open_buff)) != NULL) {
                strncpy (file_open_name, p,
                    COB_SMALL_MAX);
                break;
            }
        }
        if (i == NUM_PREFIX && cob_file_path) {
            sprintf (file_open_buff, COB_SMALL_MAX, "%s/%s",

```

```

                                cob_file_path, file_open_name);
                                strncpy (file_open_name, file_open_buff,
                                        COB_SMALL_MAX);
                                }
                                }
                                /* check if the file exists */
#ifdef USE_DB41
                                if (f->organization == COB_ORG_INDEXED) {
                                    if ((bdb_env && bdb_nofile (file_open_name)) ||
                                        (!bdb_env && stat (file_open_name, &st) == -1 && errno == EN
OENT)) {
                                        was_not_exist = 1;
                                        if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                        }
                                    }
                                } else if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#else
                                /* USE_DB41 */

#ifdef WITH_CISAM || defined(WITH_DISAM) || defined(WITH_VBISAM)
                                if (f->organization == COB_ORG_INDEXED) {
                                    strncpy (file_open_buff, file_open_name, COB_SMALL_MAX);
                                    strcat (file_open_buff, ".idx");
                                    if (stat (file_open_buff, &st) == -1 && errno == ENOENT) {
                                        was_not_exist = 1;
                                        if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                        }
                                    }
                                    strncpy (file_open_buff, file_open_name, COB_SMALL_MAX);
                                    strcat (file_open_buff, ".dat");
                                    if (stat (file_open_buff, &st) == -1 && errno == ENOENT) {
                                        was_not_exist = 1;
                                        if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                        }
                                    }
                                } else if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#else
                                /* WITH_CISAM || WITH_DISAM || WITH_VBISAM */
                                if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#endif
                                /* WITH_CISAM || WITH_DISAM || WITH_VBISAM */

#ifdef USE_DB41
                                was_not_exist = 1;
                                if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                    RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                }
                                }

#ifdef WITH_INDEX_EXTFH || defined(WITH_SEQRA_EXTFH)
                                file_available:
#endif
                                /* WITH_INDEX_EXTFH || WITH_SEQRA_EXTFH */

                                cob_cache_file (f);

                                /* open the file */
#ifdef WITH_SEQRA_EXTFH
                                if (f->organization != COB_ORG_INDEXED) {
                                    int     ret;

```

```

ret = extfh_cob_file_open (f, file_open_name, mode, sharing);
switch (ret) {
case COB_STATUS_00_SUCCESS:
    f->open_mode = mode;
    break;
case COB_STATUS_35_NOT_EXISTS:
    if (f->flag_optional) {
        f->open_mode = mode;
        f->flag_nonexistent = 1;
        f->flag_end_of_file = 1;
        f->flag_begin_of_file = 1;
        RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
    }
    break;
}
RETURN_STATUS (ret);
}
#endif
#if defined(WITH_CISAM) || defined(WITH_DISAM) || defined(WITH_VBISAM)
if (f->organization == COB_ORG_INDEXED) {
    /* Do this here to avoid mangling of the status in the 'switch' b
elow */
    RETURN_STATUS (fileio_funcs[(int)f->organization]->open (f, file_
open_name, mode, sharing));
}
#endif
switch (fileio_funcs[(int)f->organization]->open (f, file_open_name, mode
, sharing)) {
case 0:
    f->open_mode = mode;
    if (f->flag_optional && was_not_exist) {
        RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
    } else {
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    }
}
case ENOENT:
    if (mode == COB_OPEN_EXTEND || mode == COB_OPEN_OUTPUT) {
        RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
    }
    if (f->flag_optional) {
        f->open_mode = mode;
        f->flag_nonexistent = 1;
        f->flag_end_of_file = 1;
        f->flag_begin_of_file = 1;
        RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
    } else {
        RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
    }
}
case EACCES:
case EISDIR:
case EROFS:
    RETURN_STATUS (COB_STATUS_37_PERMISSION_DENIED);
case EAGAIN:
case COB_STATUS_61_FILE_SHARING:
    RETURN_STATUS (COB_STATUS_61_FILE_SHARING);
case COB_STATUS_91_NOT_AVAILABLE:
    RETURN_STATUS (COB_STATUS_91_NOT_AVAILABLE);
case COB_LINAGE_INVALID:
    RETURN_STATUS (COB_STATUS_57_I_O_LINAGE);
default:
    RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
}

```



```
}

```

**7.34.2.34** `void cob_read ( cob_file * f, cob_field * key, cob_field * fnstatus, int read_opts )`

Definition at line 4081 of file fileio.c.

```
{
    int    ret;

    f->flag_read_done = 0;

    if (unlikely(f->flag_nonexistent)) {
        if (f->flag_first_read == 0) {
            RETURN_STATUS (COB_STATUS_23_KEY_NOT_EXISTS);
        }
        f->flag_first_read = 0;
        RETURN_STATUS (COB_STATUS_10_END_OF_FILE);
    }

    /* sequential read at the end of file is an error */
    if (key == NULL) {
        if (f->flag_end_of_file && !(read_opts & COB_READ_PREVIOUS)) {
            RETURN_STATUS (COB_STATUS_46_READ_ERROR);
        }
        if (f->flag_begin_of_file && (read_opts & COB_READ_PREVIOUS)) {
            RETURN_STATUS (COB_STATUS_46_READ_ERROR);
        }
    }

    if (unlikely(f->open_mode == COB_OPEN_CLOSED
        || f->open_mode == COB_OPEN_OUTPUT
        || f->open_mode == COB_OPEN_EXTEND)) {
        RETURN_STATUS (COB_STATUS_47_INPUT_DENIED);
    }

#ifdef USE_DB41
    if (f->organization == COB_ORG_INDEXED && bdb_env != NULL) {
        if (f->open_mode != COB_OPEN_I_O ||
            (f->lock_mode & COB_LOCK_EXCLUSIVE)) {
            read_opts &= ~COB_READ_LOCK;
        } else if ((f->lock_mode & COB_LOCK_AUTOMATIC) &&
            !(read_opts & COB_READ_NO_LOCK)) {
            read_opts |= COB_READ_LOCK;
        }
    } else {
        read_opts &= ~COB_READ_LOCK;
    }
#endif

    if (key) {
        ret = fileio_funcs[(int)f->organization]->read (f, key, read_opts
    );
    } else {
        ret = fileio_funcs[(int)f->organization]->read_next (f, read_opts
    );
    }

    switch (ret) {
        case COB_STATUS_00_SUCCESS:

```

```

        f->flag_first_read = 0;
        f->flag_read_done = 1;
        f->flag_end_of_file = 0;
        f->flag_begin_of_file = 0;
        if (f->record_size && f->organization != COB_ORG_LINE_SEQUENTIAL)
        {
                cob_set_int (f->record_size, (int) f->record->size);
        }
        break;
case COB_STATUS_10_END_OF_FILE:
        if (read_opts & COB_READ_PREVIOUS) {
                f->flag_begin_of_file = 1;
        } else {
                f->flag_end_of_file = 1;
        }
        break;
}

RETURN_STATUS (ret);
}

```

#### 7.34.2.35 void cob\_rewrite ( cob\_file \* f, cob\_field \* rec, const int opt, cob\_field \* fnstatus )

Definition at line 4201 of file fileio.c.

```

{
        int      ret;
        int      read_done = f->flag_read_done;

        f->flag_read_done = 0;

        if (unlikely(f->open_mode == COB_OPEN_CLOSED ||
                f->open_mode != COB_OPEN_I_O)) {
                RETURN_STATUS (COB_STATUS_49_I_O_DENIED);
        }

        if (f->access_mode == COB_ACCESS_SEQUENTIAL && !read_done) {
                RETURN_STATUS (COB_STATUS_43_READ_NOT_DONE);
        }

        if (f->organization == COB_ORG_SEQUENTIAL) {
                if (f->record->size != rec->size) {
                        RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
                }

                if (f->record_size) {
                        if (f->record->size != (size_t)cob_get_int (f->
record_size)) {
                                RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
                        }
                }
        }

        /* RXW
#ifdef USE_DB41
        if (f->organization != COB_ORG_INDEXED || bdb_env == NULL) {
                opt &= ~COB_WRITE_LOCK;
        }

```

```

#endif
*/

    ret = fileio_funcs[(int)f->organization]->rewrite (f, opt);

    if (unlikely(cob_do_sync && ret == 0)) {
        cob_sync (f, cob_do_sync);
    }

    RETURN_STATUS (ret);
}

```

#### 7.34.2.36 void cob\_rollback ( void )

Definition at line 4283 of file fileio.c.

```

{
    struct file_list    *l;

    for (l = file_cache; l; l = l->next) {
        cob_file_unlock (l->file);
    }
}

```

#### 7.34.2.37 void cob\_start ( cob\_file \* f, const int cond, cob\_field \* key, cob\_field \* fnstatus )

Definition at line 4052 of file fileio.c.

```

{
    int    ret;

    f->flag_read_done = 0;
    f->flag_first_read = 0;

    if (f->flag_nonexistent) {
        RETURN_STATUS (COB_STATUS_23_KEY_NOT_EXISTS);
    }

    if (f->open_mode == COB_OPEN_CLOSED
        || f->open_mode == COB_OPEN_OUTPUT
        || f->open_mode == COB_OPEN_EXTEND
        || f->access_mode == COB_ACCESS_RANDOM) {
        RETURN_STATUS (COB_STATUS_47_INPUT_DENIED);
    }

    ret = fileio_funcs[(int)f->organization]->start (f, cond, key);
    if (ret == COB_STATUS_00_SUCCESS) {
        f->flag_end_of_file = 0;
        f->flag_begin_of_file = 0;
        f->flag_first_read = 1;
    }

    RETURN_STATUS (ret);
}

```

**7.34.2.38 void cob\_unlock\_file ( cob\_file \* f, cob\_field \* fnstatus )**

Definition at line 3727 of file fileio.c.

```

{
    cob_file_unlock (f);
    RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

**7.34.2.39 void cob\_write ( cob\_file \* f, cob\_field \* rec, const int opt, cob\_field \* fnstatus )**

Definition at line 4153 of file fileio.c.

```

{
    int    ret;

    f->flag_read_done = 0;

    if (f->access_mode == COB_ACCESS_SEQUENTIAL) {
        if (f->open_mode == COB_OPEN_CLOSED
            || f->open_mode == COB_OPEN_INPUT
            || f->open_mode == COB_OPEN_I_O) {
            RETURN_STATUS (COB_STATUS_48_OUTPUT_DENIED);
        }
    } else {
        if (f->open_mode == COB_OPEN_CLOSED
            || f->open_mode == COB_OPEN_INPUT
            || f->open_mode == COB_OPEN_EXTEND) {
            RETURN_STATUS (COB_STATUS_48_OUTPUT_DENIED);
        }
    }

    if (f->record_size) {
        f->record->size = cob_get_int (f->record_size);
    } else {
        f->record->size = rec->size;
    }

    if (f->record->size < f->record_min || f->record_max < f->record->size) {
        RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
    }

    /* RXW
    #ifdef USE_DB41
    if (f->organization != COB_ORG_INDEXED || bdb_env == NULL) {
        opt &= ~COB_WRITE_LOCK;
    }
    #endif
    */

    ret = fileio_funcs[(int)f->organization]->write (f, opt);

    if (unlikely(cob_do_sync && ret == 0)) {
        cob_sync (f, cob_do_sync);
    }
}

```

```

    RETURN_STATUS (ret);
}

```

### 7.34.3 Variable Documentation

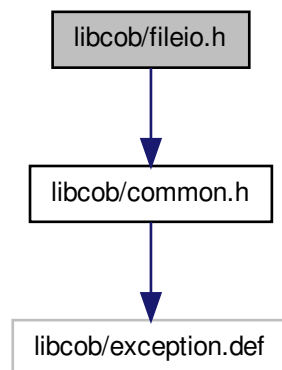
#### 7.34.3.1 cob\_file\* cob\_error\_file

Definition at line 197 of file fileio.c.

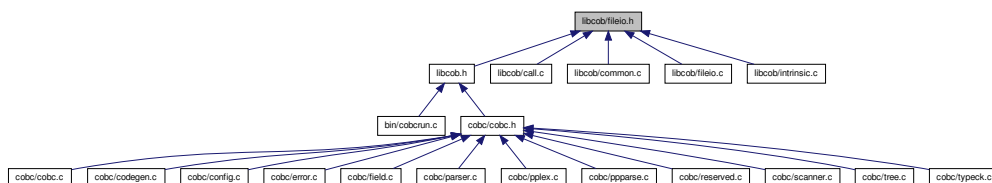
## 7.35 libcob/fileio.h File Reference

```
#include <libcob/common.h>
```

Include dependency graph for fileio.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [cob\\_file\\_key](#)
- struct [linage\\_struct](#)
- struct [cob\\_file](#)
- struct [cob\\_fileio\\_funcs](#)

## Defines

- #define [COB\\_FILE\\_VERSION](#) 0
- #define [COB\\_EQ](#) 1
- #define [COB\\_LT](#) 2
- #define [COB\\_LE](#) 3
- #define [COB\\_GT](#) 4
- #define [COB\\_GE](#) 5
- #define [COB\\_NE](#) 6
- #define [COB\\_ASCENDING](#) 0
- #define [COB\\_DESCENDING](#) 1
- #define [COB\\_FILE\\_MODE](#) 0644
- #define [COB\\_ORG\\_SEQUENTIAL](#) 0
- #define [COB\\_ORG\\_LINE\\_SEQUENTIAL](#) 1
- #define [COB\\_ORG\\_RELATIVE](#) 2
- #define [COB\\_ORG\\_INDEXED](#) 3
- #define [COB\\_ORG\\_SORT](#) 4
- #define [COB\\_ORG\\_MAX](#) 5
- #define [COB\\_ACCESS\\_SEQUENTIAL](#) 1
- #define [COB\\_ACCESS\\_DYNAMIC](#) 2
- #define [COB\\_ACCESS\\_RANDOM](#) 3
- #define [COB\\_SELECT\\_FILE\\_STATUS](#) 0x01
- #define [COB\\_SELECT\\_EXTERNAL](#) 0x02
- #define [COB\\_SELECT\\_LINAGE](#) 0x04
- #define [COB\\_SELECT\\_SPLITKEY](#) 0x08
- #define [COB\\_LOCK\\_EXCLUSIVE](#) 1
- #define [COB\\_LOCK\\_MANUAL](#) 2
- #define [COB\\_LOCK\\_AUTOMATIC](#) 4
- #define [COB\\_LOCK\\_MULTIPLE](#) 8
- #define [COB\\_LOCK\\_MASK](#) 0x7
- #define [COB\\_OPEN\\_CLOSED](#) 0
- #define [COB\\_OPEN\\_INPUT](#) 1
- #define [COB\\_OPEN\\_OUTPUT](#) 2
- #define [COB\\_OPEN\\_I\\_O](#) 3
- #define [COB\\_OPEN\\_EXTEND](#) 4
- #define [COB\\_OPEN\\_LOCKED](#) 5
- #define [COB\\_CLOSE\\_NORMAL](#) 0
- #define [COB\\_CLOSE\\_LOCK](#) 1
- #define [COB\\_CLOSE\\_NO\\_REWIND](#) 2

- #define COB\_CLOSE\_UNIT 3
- #define COB\_CLOSE\_UNIT\_REMOVAL 4
- #define COB\_WRITE\_MASK 0x0000ffff
- #define COB\_WRITE\_LINES 0x00010000
- #define COB\_WRITE\_PAGE 0x00020000
- #define COB\_WRITE\_CHANNEL 0x00040000
- #define COB\_WRITE\_AFTER 0x00100000
- #define COB\_WRITE\_BEFORE 0x00200000
- #define COB\_WRITE\_EOP 0x00400000
- #define COB\_WRITE\_LOCK 0x00800000
- #define COB\_READ\_NEXT 0x01
- #define COB\_READ\_PREVIOUS 0x02
- #define COB\_READ\_FIRST 0x04
- #define COB\_READ\_LAST 0x08
- #define COB\_READ\_LOCK 0x10
- #define COB\_READ\_NO\_LOCK 0x20
- #define COB\_READ\_KEPT\_LOCK 0x40
- #define COB\_READ\_WAIT\_LOCK 0x80
- #define COB\_READ\_IGNORE\_LOCK 0x100
- #define COB\_STATUS\_00\_SUCCESS 00
- #define COB\_STATUS\_02\_SUCCESS\_DUPLICATE 02
- #define COB\_STATUS\_04\_SUCCESS\_INCOMPLETE 04
- #define COB\_STATUS\_05\_SUCCESS\_OPTIONAL 05
- #define COB\_STATUS\_07\_SUCCESS\_NO\_UNIT 07
- #define COB\_STATUS\_10\_END\_OF\_FILE 10
- #define COB\_STATUS\_14\_OUT\_OF\_KEY\_RANGE 14
- #define COB\_STATUS\_21\_KEY\_INVALID 21
- #define COB\_STATUS\_22\_KEY\_EXISTS 22
- #define COB\_STATUS\_23\_KEY\_NOT\_EXISTS 23
- #define COB\_STATUS\_30\_PERMANENT\_ERROR 30
- #define COB\_STATUS\_31\_INCONSISTENT\_FILENAME 31
- #define COB\_STATUS\_34\_BOUNDARY\_VIOLATION 34
- #define COB\_STATUS\_35\_NOT\_EXISTS 35
- #define COB\_STATUS\_37\_PERMISSION\_DENIED 37
- #define COB\_STATUS\_38\_CLOSED\_WITH\_LOCK 38
- #define COB\_STATUS\_39\_CONFLICT\_ATTRIBUTE 39
- #define COB\_STATUS\_41\_ALREADY\_OPEN 41
- #define COB\_STATUS\_42\_NOT\_OPEN 42
- #define COB\_STATUS\_43\_READ\_NOT\_DONE 43
- #define COB\_STATUS\_44\_RECORD\_OVERFLOW 44
- #define COB\_STATUS\_46\_READ\_ERROR 46
- #define COB\_STATUS\_47\_INPUT\_DENIED 47
- #define COB\_STATUS\_48\_OUTPUT\_DENIED 48
- #define COB\_STATUS\_49\_I\_O\_DENIED 49
- #define COB\_STATUS\_51\_RECORD\_LOCKED 51
- #define COB\_STATUS\_52\_EOP 52

- #define `COB_STATUS_57_I_O_LINAGE` 57
- #define `COB_STATUS_61_FILE_SHARING` 61
- #define `COB_STATUS_91_NOT_AVAILABLE` 91
- #define `COB_LINAGE_INVALID` 16384
- #define `COB_NOT_CONFIGURED` 32768

## Functions

- void `cob_default_error_handle` (void)
- void `cob_open` (`cob_file` \*, const int, const int, `cob_field` \*)
- void `cob_close` (`cob_file` \*, const int, `cob_field` \*)
- void `cob_read` (`cob_file` \*, `cob_field` \*, `cob_field` \*, int)
- void `cob_write` (`cob_file` \*, `cob_field` \*, const int, `cob_field` \*)
- void `cob_rewrite` (`cob_file` \*, `cob_field` \*, const int, `cob_field` \*)
- void `cob_delete` (`cob_file` \*, `cob_field` \*)
- void `cob_start` (`cob_file` \*, const int, `cob_field` \*, `cob_field` \*)
- void `cob_unlock_file` (`cob_file` \*, `cob_field` \*)
- void `cob_commit` (void)
- void `cob_rollback` (void)
- int `CBL_OPEN_FILE` (unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*)
- int `CBL_CREATE_FILE` (unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*)
- int `CBL_READ_FILE` (unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*)
- int `CBL_WRITE_FILE` (unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*)
- int `CBL_CLOSE_FILE` (unsigned char \*)
- int `CBL_FLUSH_FILE` (unsigned char \*)
- int `CBL_DELETE_FILE` (unsigned char \*)
- int `CBL_COPY_FILE` (unsigned char \*, unsigned char \*)
- int `CBL_CHECK_FILE_EXIST` (unsigned char \*, unsigned char \*)
- int `CBL_RENAME_FILE` (unsigned char \*, unsigned char \*)
- int `CBL_GET_CURRENT_DIR` (const int, const int, unsigned char \*)
- int `CBL_CHANGE_DIR` (unsigned char \*)
- int `CBL_CREATE_DIR` (unsigned char \*)
- int `CBL_DELETE_DIR` (unsigned char \*)
- int `cob_acuw_chdir` (unsigned char \*, unsigned char \*)
- int `cob_acuw_mkdir` (unsigned char \*)
- int `cob_acuw_copyfile` (unsigned char \*, unsigned char \*, unsigned char \*)
- int `cob_acuw_file_info` (unsigned char \*, unsigned char \*)
- int `cob_acuw_file_delete` (unsigned char \*, unsigned char \*)
- void `cob_file_sort_init` (`cob_file` \*, const int, const unsigned char \*, void \*, `cob_field` \*)
- void `cob_file_sort_init_key` (`cob_file` \*, const int, `cob_field` \*, size\_t)
- void `cob_file_sort_close` (`cob_file` \*)



- void `cob_file_sort_using` (`cob_file *`, `cob_file *`)
- void `cob_file_sort_giving` (`cob_file *`, `const size_t,...`)
- void `cob_file_release` (`cob_file *`)
- void `cob_file_return` (`cob_file *`)

## Variables

- `DLL_EXPIMP` `cob_file *` `cob_error_file`

### 7.35.1 Define Documentation

#### 7.35.1.1 `#define COB_ACCESS_DYNAMIC 2`

Definition at line 53 of file fileio.h.

#### 7.35.1.2 `#define COB_ACCESS_RANDOM 3`

Definition at line 54 of file fileio.h.

#### 7.35.1.3 `#define COB_ACCESS_SEQUENTIAL 1`

Definition at line 52 of file fileio.h.

#### 7.35.1.4 `#define COB_ASCENDING 0`

Definition at line 36 of file fileio.h.

#### 7.35.1.5 `#define COB_CLOSE_LOCK 1`

Definition at line 83 of file fileio.h.

#### 7.35.1.6 `#define COB_CLOSE_NO_REWIND 2`

Definition at line 84 of file fileio.h.

#### 7.35.1.7 `#define COB_CLOSE_NORMAL 0`

Definition at line 82 of file fileio.h.

#### 7.35.1.8 `#define COB_CLOSE_UNIT 3`

Definition at line 85 of file fileio.h.

7.35.1.9 `#define COB_CLOSE_UNIT_REMOVAL 4`

Definition at line 86 of file fileio.h.

7.35.1.10 `#define COB_DESCENDING 1`

Definition at line 37 of file fileio.h.

7.35.1.11 `#define COB_EQ 1`

Definition at line 29 of file fileio.h.

7.35.1.12 `#define COB_FILE_MODE 0644`

Definition at line 39 of file fileio.h.

7.35.1.13 `#define COB_FILE_VERSION 0`

Definition at line 27 of file fileio.h.

7.35.1.14 `#define COB_GE 5`

Definition at line 33 of file fileio.h.

7.35.1.15 `#define COB_GT 4`

Definition at line 32 of file fileio.h.

7.35.1.16 `#define COB_LE 3`

Definition at line 31 of file fileio.h.

7.35.1.17 `#define COB_LINAGE_INVALID 16384`

Definition at line 146 of file fileio.h.

7.35.1.18 `#define COB_LOCK_AUTOMATIC 4`

Definition at line 67 of file fileio.h.

7.35.1.19 **#define COB\_LOCK\_EXCLUSIVE 1**

Definition at line 65 of file fileio.h.

7.35.1.20 **#define COB\_LOCK\_MANUAL 2**

Definition at line 66 of file fileio.h.

7.35.1.21 **#define COB\_LOCK\_MASK 0x7**

Definition at line 69 of file fileio.h.

7.35.1.22 **#define COB\_LOCK\_MULTIPLE 8**

Definition at line 68 of file fileio.h.

7.35.1.23 **#define COB\_LT 2**

Definition at line 30 of file fileio.h.

7.35.1.24 **#define COB\_NE 6**

Definition at line 34 of file fileio.h.

7.35.1.25 **#define COB\_NOT\_CONFIGURED 32768**

Definition at line 148 of file fileio.h.

7.35.1.26 **#define COB\_OPEN\_CLOSED 0**

Definition at line 73 of file fileio.h.

7.35.1.27 **#define COB\_OPEN\_EXTEND 4**

Definition at line 77 of file fileio.h.

7.35.1.28 **#define COB\_OPEN\_I.O 3**

Definition at line 76 of file fileio.h.

7.35.1.29 `#define COB_OPEN_INPUT 1`

Definition at line 74 of file fileio.h.

7.35.1.30 `#define COB_OPEN_LOCKED 5`

Definition at line 78 of file fileio.h.

7.35.1.31 `#define COB_OPEN_OUTPUT 2`

Definition at line 75 of file fileio.h.

7.35.1.32 `#define COB_ORG_INDEXED 3`

Definition at line 46 of file fileio.h.

7.35.1.33 `#define COB_ORG_LINE_SEQUENTIAL 1`

Definition at line 44 of file fileio.h.

7.35.1.34 `#define COB_ORG_MAX 5`

Definition at line 48 of file fileio.h.

7.35.1.35 `#define COB_ORG_RELATIVE 2`

Definition at line 45 of file fileio.h.

7.35.1.36 `#define COB_ORG_SEQUENTIAL 0`

Definition at line 43 of file fileio.h.

7.35.1.37 `#define COB_ORG_SORT 4`

Definition at line 47 of file fileio.h.

7.35.1.38 `#define COB_READ_FIRST 0x04`

Definition at line 102 of file fileio.h.

7.35.1.39 `#define COB_READ_IGNORE_LOCK 0x100`

Definition at line 108 of file fileio.h.

7.35.1.40 `#define COB_READ_KEPT_LOCK 0x40`

Definition at line 106 of file fileio.h.

7.35.1.41 `#define COB_READ_LAST 0x08`

Definition at line 103 of file fileio.h.

7.35.1.42 `#define COB_READ_LOCK 0x10`

Definition at line 104 of file fileio.h.

7.35.1.43 `#define COB_READ_NEXT 0x01`

Definition at line 100 of file fileio.h.

7.35.1.44 `#define COB_READ_NO_LOCK 0x20`

Definition at line 105 of file fileio.h.

7.35.1.45 `#define COB_READ_PREVIOUS 0x02`

Definition at line 101 of file fileio.h.

7.35.1.46 `#define COB_READ_WAIT_LOCK 0x80`

Definition at line 107 of file fileio.h.

7.35.1.47 `#define COB_SELECT_EXTERNAL 0x02`

Definition at line 59 of file fileio.h.

7.35.1.48 `#define COB_SELECT_FILE_STATUS 0x01`

Definition at line 58 of file fileio.h.

7.35.1.49 `#define COB_SELECT_LINAGE 0x04`

Definition at line 60 of file fileio.h.

7.35.1.50 `#define COB_SELECT_SPLITKEY 0x08`

Definition at line 61 of file fileio.h.

7.35.1.51 `#define COB_STATUS_00_SUCCESS 00`

Definition at line 112 of file fileio.h.

7.35.1.52 `#define COB_STATUS_02_SUCCESS_DUPLICATE 02`

Definition at line 113 of file fileio.h.

7.35.1.53 `#define COB_STATUS_04_SUCCESS_INCOMPLETE 04`

Definition at line 114 of file fileio.h.

7.35.1.54 `#define COB_STATUS_05_SUCCESS_OPTIONAL 05`

Definition at line 115 of file fileio.h.

7.35.1.55 `#define COB_STATUS_07_SUCCESS_NO_UNIT 07`

Definition at line 116 of file fileio.h.

7.35.1.56 `#define COB_STATUS_10_END_OF_FILE 10`

Definition at line 117 of file fileio.h.

7.35.1.57 `#define COB_STATUS_14_OUT_OF_KEY_RANGE 14`

Definition at line 118 of file fileio.h.

7.35.1.58 `#define COB_STATUS_21_KEY_INVALID 21`

Definition at line 119 of file fileio.h.

**7.35.1.59 #define COB\_STATUS\_22\_KEY\_EXISTS 22**

Definition at line 120 of file fileio.h.

**7.35.1.60 #define COB\_STATUS\_23\_KEY\_NOT\_EXISTS 23**

Definition at line 121 of file fileio.h.

**7.35.1.61 #define COB\_STATUS\_30\_PERMANENT\_ERROR 30**

Definition at line 122 of file fileio.h.

**7.35.1.62 #define COB\_STATUS\_31\_INCONSISTENT\_FILENAME 31**

Definition at line 123 of file fileio.h.

**7.35.1.63 #define COB\_STATUS\_34\_BOUNDARY\_VIOLATION 34**

Definition at line 124 of file fileio.h.

**7.35.1.64 #define COB\_STATUS\_35\_NOT\_EXISTS 35**

Definition at line 125 of file fileio.h.

**7.35.1.65 #define COB\_STATUS\_37\_PERMISSION\_DENIED 37**

Definition at line 126 of file fileio.h.

**7.35.1.66 #define COB\_STATUS\_38\_CLOSED\_WITH\_LOCK 38**

Definition at line 127 of file fileio.h.

**7.35.1.67 #define COB\_STATUS\_39\_CONFLICT\_ATTRIBUTE 39**

Definition at line 128 of file fileio.h.

**7.35.1.68 #define COB\_STATUS\_41\_ALREADY\_OPEN 41**

Definition at line 129 of file fileio.h.

7.35.1.69 `#define COB_STATUS_42_NOT_OPEN 42`

Definition at line 130 of file fileio.h.

7.35.1.70 `#define COB_STATUS_43_READ_NOT_DONE 43`

Definition at line 131 of file fileio.h.

7.35.1.71 `#define COB_STATUS_44_RECORD_OVERFLOW 44`

Definition at line 132 of file fileio.h.

7.35.1.72 `#define COB_STATUS_46_READ_ERROR 46`

Definition at line 133 of file fileio.h.

7.35.1.73 `#define COB_STATUS_47_INPUT_DENIED 47`

Definition at line 134 of file fileio.h.

7.35.1.74 `#define COB_STATUS_48_OUTPUT_DENIED 48`

Definition at line 135 of file fileio.h.

7.35.1.75 `#define COB_STATUS_49_I_O_DENIED 49`

Definition at line 136 of file fileio.h.

7.35.1.76 `#define COB_STATUS_51_RECORD_LOCKED 51`

Definition at line 137 of file fileio.h.

7.35.1.77 `#define COB_STATUS_52_EOP 52`

Definition at line 138 of file fileio.h.

7.35.1.78 `#define COB_STATUS_57_I_O_LINAGE 57`

Definition at line 139 of file fileio.h.



7.35.1.79 `#define COB_STATUS_61_FILE_SHARING 61`

Definition at line 140 of file fileio.h.

7.35.1.80 `#define COB_STATUS_91_NOT_AVAILABLE 91`

Definition at line 141 of file fileio.h.

7.35.1.81 `#define COB_WRITE_AFTER 0x00100000`

Definition at line 94 of file fileio.h.

7.35.1.82 `#define COB_WRITE_BEFORE 0x00200000`

Definition at line 95 of file fileio.h.

7.35.1.83 `#define COB_WRITE_CHANNEL 0x00040000`

Definition at line 93 of file fileio.h.

7.35.1.84 `#define COB_WRITE_EOP 0x00400000`

Definition at line 96 of file fileio.h.

7.35.1.85 `#define COB_WRITE_LINES 0x00010000`

Definition at line 91 of file fileio.h.

7.35.1.86 `#define COB_WRITE_LOCK 0x00800000`

Definition at line 97 of file fileio.h.

7.35.1.87 `#define COB_WRITE_MASK 0x0000ffff`

Definition at line 90 of file fileio.h.

7.35.1.88 `#define COB_WRITE_PAGE 0x00020000`

Definition at line 92 of file fileio.h.

## 7.35.2 Function Documentation

### 7.35.2.1 int CBL\_CHANGE\_DIR ( unsigned char \* )

Definition at line 4857 of file fileio.c.

```
{
    char      *fn;
    int       ret;

    COB_CHK_PARAMS (CBL_CHANGE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = chdir (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}
```

### 7.35.2.2 int CBL\_CHECK\_FILE\_EXIST ( unsigned char \*, unsigned char \* )

Definition at line 4722 of file fileio.c.

```
{
    char          *fn;
    struct tm     *tm;
    long long     sz;
    struct stat   st;
    short         y;
    char          d, m, hh, mm, ss;

    COB_CHK_PARAMS (CBL_CHECK_FILE_EXIST, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    if (stat (fn, &st) < 0) {
        free (fn);
        return 35;
    }
    free (fn);
    sz = st.st_size;
    tm = localtime (&st.st_mtime);
    d = (char) tm->tm_mday;
    m = (char) tm->tm_mon + 1;
    y = tm->tm_year + 1900;
    hh = (char) tm->tm_hour;
    mm = (char) tm->tm_min;
    ss = (char) tm->tm_sec;

#ifdef WORDS_BIGENDIAN
```

```

        sz = COB_BSWAP_64 (sz);
        y = COB_BSWAP_16 (y);
#endif
    memcpy (file_info, &sz, 8);
    file_info[8] = d;
    file_info[9] = m;
    memcpy (file_info+10, &y, 2);
    file_info[12] = hh;
    file_info[13] = mm;
    file_info[14] = ss;
    file_info[15] = 0;
    return 0;
}

```

### 7.35.2.3 int CBL\_CLOSE\_FILE ( unsigned char \* )

Definition at line 4631 of file fileio.c.

```

{
    int    fd;

    COB_CHK_PARAMS (CBL_CLOSE_FILE, 1);

    memcpy (&fd, file_handle, 4);
    return close (fd);
}

```

### 7.35.2.4 int CBL\_COPY\_FILE ( unsigned char \*, unsigned char \* )

Definition at line 4670 of file fileio.c.

```

{
    char    *fn1;
    char    *fn2;
#ifdef O_BINARY
    int     flag = O_BINARY;
#else
    int     flag = 0;
#endif
    int     ret;
    int     i;
    int     fd1, fd2;

    COB_CHK_PARAMS (CBL_COPY_FILE, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    if (!cob_current_module->cob_procedure_parameters[1]) {
        return -1;
    }
    fn1 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);

    flag |= O_RDONLY;
    fd1 = open (fn1, flag, 0);

```

```

    if (fd1 < 0) {
        free (fn1);
        return -1;
    }
    free (fn1);
    fn2 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[1]);

    flag &= ~O_RDONLY;
    flag |= O_CREAT | O_TRUNC | O_WRONLY;
    fd2 = open (fn2, flag, 0660);
    if (fd2 < 0) {
        close (fd1);
        free (fn2);
        return -1;
    }
    free (fn2);
    ret = 0;
    while ((i = read (fd1, fn1, sizeof(fn1))) > 0) {
        if (write (fd2, fn1, (size_t)i) < 0) {
            ret = -1;
            break;
        }
    }
    close (fd1);
    close (fd2);
    return ret;
}

```

### 7.35.2.5 int CBL\_CREATE\_DIR ( unsigned char \* )

Definition at line 4833 of file fileio.c.

```

{
    char    *fn;
    int     ret;

    COB_CHK_PARMS (CBL_CREATE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
#ifdef _WIN32
    ret = mkdir (fn);
#else
    ret = mkdir (fn, 0770);
#endif
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}

```

**7.35.2.6 int CBL\_CREATE\_FILE ( unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \* )**

Definition at line 4548 of file fileio.c.

```
{
    COB_CHK_PARMS (CBL_CREATE_FILE, 5);

    return open_cbl_file (file_name, file_access, file_handle, O_CREAT | O_TRUNC);
}
```

**7.35.2.7 int CBL\_DELETE\_DIR ( unsigned char \* )**

Definition at line 4877 of file fileio.c.

```
{
    char    *fn;
    int     ret;

    COB_CHK_PARMS (CBL_DELETE_DIR, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = rmdir (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}
```

**7.35.2.8 int CBL\_DELETE\_FILE ( unsigned char \* )**

Definition at line 4650 of file fileio.c.

```
{
    char    *fn;
    int     ret;

    COB_CHK_PARMS (CBL_DELETE_FILE, 1);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    ret = unlink (fn);
    free (fn);
    if (ret) {
        return 128;
    }
    return 0;
}
```

**7.35.2.9 int CBL\_FLUSH\_FILE ( unsigned char \* )**

Definition at line 4642 of file fileio.c.

```

{
    COB_CHK_PARAMS (CBL_FLUSH_FILE, 1);

    return 0;
}

```

**7.35.2.10 int CBL\_GET\_CURRENT\_DIR ( const int , const int , unsigned char \* )**

Definition at line 4793 of file fileio.c.

```

{
    char    *dirname;
    int     dir_size;
    int     has_space;

    COB_CHK_PARAMS (CBL_GET_CURRENT_DIR, 3);

    if (dir_length < 1) {
        return 128;
    }
    if (flags) {
        return 129;
    }
    memset (dir, ' ', (size_t)dir_length);
    dirname = getcwd (NULL, 0);
    if (dirname == NULL) {
        return 128;
    }
    dir_size = (int) strlen (dirname);
    has_space = 0;
    if (strchr (dirname, ' ')) {
        has_space = 2;
    }
    if (dir_size + has_space > dir_length) {
        free (dirname);
        return 128;
    }
    if (has_space) {
        *dir = ' ';
        memcpy (&dir[1], dirname, (size_t)dir_size);
        dir[dir_size + 1] = ' ';
    } else {
        memcpy (dir, dirname, (size_t)dir_size);
    }
    free (dirname);
    return 0;
}

```

**7.35.2.11 int CBL\_OPEN\_FILE ( unsigned char \* , unsigned char \* , unsigned char \* , unsigned char \* , unsigned char \* )**

Definition at line 4538 of file fileio.c.

```

{
    COB_CHK_PARMS (CBL_OPEN_FILE, 5);

    return open_cbl_file (file_name, file_access, file_handle, 0);
}

```

### 7.35.2.12 int CBL\_READ\_FILE ( unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \* )

Definition at line 4558 of file fileio.c.

```

{
    long long    off;
    int          fd;
    int          len;
    int          rc = 0;
    struct stat  st;

    COB_CHK_PARMS (CBL_READ_FILE, 5);

    memcpy (&fd, file_handle, 4);
    memcpy (&off, file_offset, 8);
    memcpy (&len, file_len, 4);
#ifdef WORDS_BIGENDIAN
    off = COB_BSWAP_64 (off);
    len = COB_BSWAP_32 (len);
#endif
    if (lseek (fd, (off_t)off, SEEK_SET) < 0) {
        return -1;
    }
    if (len > 0) {
        rc = read (fd, buf, (size_t)len);
        if (rc < 0) {
            rc = -1;
        } else if (rc == 0) {
            rc = 10;
        } else {
            rc = 0;
        }
    }
    if ((*flags & 0x80) != 0) {
        if (fstat (fd, &st) < 0) {
            return -1;
        }
        off = st.st_size;
#ifdef WORDS_BIGENDIAN
        off = COB_BSWAP_64 (off);
#endif
        memcpy (file_offset, &off, 8);
    }
    return rc;
}

```

### 7.35.2.13 int CBL\_RENAME\_FILE ( unsigned char \*, unsigned char \* )

Definition at line 4767 of file fileio.c.

```

{
    char    *fn1;
    char    *fn2;
    int     ret;

    COB_CHK_PARMS (CBL_RENAME_FILE, 2);

    if (!cob_current_module->cob_procedure_parameters[0]) {
        return -1;
    }
    if (!cob_current_module->cob_procedure_parameters[1]) {
        return -1;
    }
    fn1 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    fn2 = cob_str_from_fld (cob_current_module->cob_procedure_parameters[1]);

    ret = rename (fn1, fn2);
    free (fn1);
    free (fn2);
    if (ret) {
        return 128;
    }
    return 0;
}

```

#### 7.35.2.14 int CBL\_WRITE\_FILE ( unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \*, unsigned char \* )

Definition at line 4603 of file fileio.c.

```

{
    long long    off;
    int          fd;
    int          len;
    int          rc;

    COB_CHK_PARMS (CBL_WRITE_FILE, 5);

    memcpy (&fd, file_handle, 4);
    memcpy (&off, file_offset, 8);
    memcpy (&len, file_len, 4);
#ifdef WORDS_BIGENDIAN
    off = COB_BSWAP_64 (off);
    len = COB_BSWAP_32 (len);
#endif
    if (lseek (fd, (off_t)off, SEEK_SET) < 0) {
        return -1;
    }
    rc = write (fd, buf, (size_t)len);
    if (rc < 0) {
        return 30;
    }
    return 0;
}

```



**7.35.2.15 int cob\_acuw\_chdir ( unsigned char \*, unsigned char \* )**

Definition at line 4911 of file fileio.c.

```
{
    int          ret;

    COB_CHK_PARMS (C$CHDIR, 2);

    ret = CBL_CHANGE_DIR (dir);
    if (ret < 0) {
        ret = 128;
    }
    cob_set_int (cob_current_module->cob_procedure_parameters[1], ret);
    return ret;
}
```

**7.35.2.16 int cob\_acuw\_copyfile ( unsigned char \*, unsigned char \*, unsigned char \* )**

Definition at line 4926 of file fileio.c.

```
{
    int          ret = 128;

    /* RXW - Type is not yet evaluated */
    COB_CHK_PARMS (C$COPY, 3);

    if (cob_call_params < 3) {
        return 128;
    }
    ret = CBL_COPY_FILE (fname1, fname2);
    if (ret < 0) {
        ret = 128;
    }
    return ret;
}
```

**7.35.2.17 int cob\_acuw\_file\_delete ( unsigned char \*, unsigned char \* )**

Definition at line 4993 of file fileio.c.

```
{
    int          ret;

    /* RXW - Type is not yet evaluated */
    COB_CHK_PARMS (C$DELETE, 2);

    if (cob_call_params < 2 || !cob_current_module->cob_procedure_parameters[
0]) {
        return 128;
    }
    ret = CBL_DELETE_FILE (file_name);
    if (ret < 0) {
```

```

        ret = 128;
    }
    return ret;
}

```

### 7.35.2.18 int cob\_acuw\_file\_info ( unsigned char \*, unsigned char \* )

Definition at line 4945 of file fileio.c.

```

{
    char                *fn;
    struct tm          *tm;
    unsigned long long  sz;
    unsigned int        dt;
    short              y;
    short              d, m, hh, mm, ss;
    struct stat        st;

    COB_CHK_PARAMS (C$FILEINFO, 2);

    if (cob_call_params < 2 || !cob_current_module->cob_procedure_parameters[
0]) {
        return 128;
    }
    fn = cob_str_from_fld (cob_current_module->cob_procedure_parameters[0]);
    if (stat (fn, &st) < 0) {
        free (fn);
        return 35;
    }
    free (fn);
    sz = st.st_size;
    tm = localtime (&st.st_mtime);
    d = tm->tm_mday;
    m = tm->tm_mon + 1;
    y = tm->tm_year + 1900;
    hh = tm->tm_hour;
    mm = tm->tm_min;
    ss = tm->tm_sec;

#ifdef WORDS_BIGENDIAN
    sz = COB_BSWAP_64 (sz);
#endif
    memcpy (file_info, &sz, 8);
    dt = (y * 10000) + (m * 100) + d;
#ifdef WORDS_BIGENDIAN
    dt = COB_BSWAP_32 (dt);
#endif
    memcpy (file_info + 8, &dt, 4);
    dt = (hh * 1000000) + (mm * 10000) + (ss * 100);
#ifdef WORDS_BIGENDIAN
    dt = COB_BSWAP_32 (dt);
#endif
    memcpy (file_info + 12, &dt, 4);
    return 0;
}

```

**7.35.2.19 int cob\_acuw\_mkdir ( unsigned char \* )**

Definition at line 4897 of file fileio.c.

```
{
    int          ret;

    COB_CHK_PARMS (C$MAKEDIR, 1);

    ret = CBL_CREATE_DIR (dir);
    if (ret < 0) {
        ret = 128;
    }
    return ret;
}
```

**7.35.2.20 void cob\_close ( cob\_file \*, const int , cob\_field \* )**

Definition at line 3987 of file fileio.c.

```
{
    int          ret;

    f->flag_read_done = 0;

    if (f->special) {
        f->open_mode = COB_OPEN_CLOSED;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    }
    if (f->open_mode == COB_OPEN_CLOSED) {
        RETURN_STATUS (COB_STATUS_42_NOT_OPEN);
    }

    if (f->flag_nonexistent) {
        ret = COB_STATUS_00_SUCCESS;
    } else {
#ifdef WITH_SEQRA_EXTFH
        if (f->organization != COB_ORG_INDEXED) {
            ret = extfh_cob_file_close (f, opt);
        } else {
#endif
            ret = fileio_funcs[(int)f->organization]->close (f, opt);
#ifdef WITH_SEQRA_EXTFH
        }
#endif
    }

    if (ret == COB_STATUS_00_SUCCESS) {
        switch (opt) {
            case COB_CLOSE_LOCK:
                f->open_mode = COB_OPEN_LOCKED;
                break;
            default:
                f->open_mode = COB_OPEN_CLOSED;
                break;
        }
    }
}
```

```
        RETURN_STATUS (ret);
    }
```

#### 7.35.2.21 void cob\_commit ( void )

Definition at line 4273 of file fileio.c.

```
{
    struct file_list      *l;

    for (l = file_cache; l; l = l->next) {
        cob_file_unlock (l->file);
    }
}
```

#### 7.35.2.22 void cob\_default\_error\_handle ( void )

Definition at line 4293 of file fileio.c.

```
{
    const char      *msg;
    unsigned char   *file_status;
    char            *filename;
    int             status;

    file_status = cob_error_file->file_status;
    status = cob_d2i(file_status[0]) * 10 + cob_d2i(file_status[1]);
    switch (status) {
    case COB_STATUS_10_END_OF_FILE:
        msg = "End of file";
        break;
    case COB_STATUS_14_OUT_OF_KEY_RANGE:
        msg = "Key out of range";
        break;
    case COB_STATUS_21_KEY_INVALID:
        msg = "Key order not ascending";
        break;
    case COB_STATUS_22_KEY_EXISTS:
        msg = "Record key already exists";
        break;
    case COB_STATUS_23_KEY_NOT_EXISTS:
        msg = "Record key does not exist";
        break;
    case COB_STATUS_30_PERMANENT_ERROR:
        msg = "Permanent file error";
        break;
    case COB_STATUS_35_NOT_EXISTS:
        msg = "File does not exist";
        break;
    case COB_STATUS_37_PERMISSION_DENIED:
        msg = "Permission denied";
        break;
    case COB_STATUS_41_ALREADY_OPEN:
        msg = "File already open";
    }
}
```

```

        break;
case COB_STATUS_42_NOT_OPEN:
    msg = "File not open";
    break;
case COB_STATUS_43_READ_NOT_DONE:
    msg = "READ must be executed first";
    break;
case COB_STATUS_44_RECORD_OVERFLOW:
    msg = "Record overflow";
    break;
case COB_STATUS_46_READ_ERROR:
    msg = "Failed to read";
    break;
case COB_STATUS_47_INPUT_DENIED:
    msg = "READ/START not allowed";
    break;
case COB_STATUS_48_OUTPUT_DENIED:
    msg = "WRITE not allowed";
    break;
case COB_STATUS_49_I_O_DENIED:
    msg = "DELETE/REWRITE not allowed";
    break;
case COB_STATUS_51_RECORD_LOCKED:
    msg = "Record locked by another file connector";
    break;
case COB_STATUS_52_EOP:
    msg = "A page overflow condition occurred";
    break;
case COB_STATUS_57_I_O_LINAGE:
    msg = "LINAGE values invalid";
    break;
case COB_STATUS_61_FILE_SHARING:
    msg = "File sharing conflict";
    break;
case COB_STATUS_91_NOT_AVAILABLE:
    msg = "Runtime library is not configured for this operation";
    break;
default:
    msg = "Unknown file error";
    break;
}

filename = cob_malloc (COB_MEDIUM_BUFF);
cob_field_to_string (cob_error_file->assign, filename);
cob_runtime_error ("%s (STATUS = %02d) File : '%s'", msg,
                  status, filename);
free (filename);
}

```

### 7.35.2.23 void cob\_delete ( cob\_file \*, cob\_field \* )

Definition at line 4247 of file fileio.c.

```

{
    int    ret;
    int    read_done = f->flag_read_done;

    f->flag_read_done = 0;
}

```

```

    if (unlikely(f->open_mode == COB_OPEN_CLOSED ||
        f->open_mode != COB_OPEN_I_O)) {
        RETURN_STATUS (COB_STATUS_49_I_O_DENIED);
    }

    if (f->access_mode == COB_ACCESS_SEQUENTIAL && !read_done) {
        RETURN_STATUS (COB_STATUS_43_READ_NOT_DONE);
    }

    ret = fileio_funcs[(int)f->organization]->fdelete (f);

    if (unlikely(cob_do_sync && ret == 0)) {
        cob_sync (f, cob_do_sync);
    }

    RETURN_STATUS (ret);
}

```

#### 7.35.2.24 void cob\_file\_release ( cob\_file \* )

Definition at line 5654 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field *fnstatus = NULL;
    int ret;

    hp = f->file;
    if (likely(hp)) {
        fnstatus = hp->fnstatus;
    }
    ret = cob_file_sort_submit (f, f->record->data);
    switch (ret) {
    case 0:
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
        break;
    default:
        if (likely(hp)) {
            *(int *) (hp->sort_return) = 16;
        }
        RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        break;
    }
}

```

#### 7.35.2.25 void cob\_file\_return ( cob\_file \* )

Definition at line 5679 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field *fnstatus = NULL;
    int ret;

    hp = f->file;

```

```

    if (likely (hp)) {
        fnstatus = hp->fnstatus;
    }
    ret = cob_file_sort_retrieve (f, f->record->data);
    switch (ret) {
    case 0:
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
        break;
    case COBSORTEND:
        RETURN_STATUS (COB_STATUS_10_END_OF_FILE);
        break;
    default:
        if (likely (hp)) {
            *(int *) (hp->sort_return) = 16;
        }
        RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        break;
    }
}

```

#### 7.35.2.26 void cob\_file\_sort\_close ( cob\_file \* )

Definition at line 5631 of file fileio.c.

```

{
    struct cobsort *hp;
    cob_field      *fnstatus = NULL;
    size_t        i;

    hp = f->file;
    if (likely (hp)) {
        fnstatus = hp->fnstatus;
        cob_free_list (hp->empty);
        for (i = 0; i < 4; i++) {
            cob_free_list (hp->queue[i].first);
            if (hp->file[i].fp != NULL) {
                fclose (hp->file[i].fp);
            }
        }
        free (hp);
    }
    f->file = NULL;
    RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

#### 7.35.2.27 void cob\_file\_sort\_giving ( cob\_file \*, const size\_t, ... )

Definition at line 5544 of file fileio.c.

```

{
    cob_file      **fbase;
    struct cobsort *hp;
    size_t        i;
    int           ret;
    int           opt;
}

```

```

va_list      args;

fbase = cob_malloc (varcnt * sizeof(cob_file *));
va_start (args, varcnt);
for (i = 0; i < varcnt; i++) {
    fbase[i] = va_arg (args, cob_file *);
}
va_end (args);
for (i = 0; i < varcnt; i++) {
    cob_open (fbase[i], COB_OPEN_OUTPUT, 0, NULL);
}
while (1) {
    ret = cob_file_sort_retrieve (sort_file, sort_file->record->data)
;
    if (ret) {
        if (ret == COBSORTEND) {
            sort_file->file_status[0] = '1';
            sort_file->file_status[1] = '0';
        } else {
            hp = sort_file->file;
            *(int *) (hp->sort_return) = 16;
            sort_file->file_status[0] = '3';
            sort_file->file_status[1] = '0';
        }
        break;
    }
    for (i = 0; i < varcnt; i++) {
        if (fbase[i]->special ||
            fbase[i]->organization == COB_ORG_LINE_SEQUENTIAL) {
            opt = COB_WRITE_BEFORE | COB_WRITE_LINES | 1;
        } else {
            opt = 0;
        }
        cob_copy_check (fbase[i], sort_file);
        cob_write (fbase[i], fbase[i]->record, opt, NULL);
    }
}
for (i = 0; i < varcnt; i++) {
    cob_close (fbase[i], COB_CLOSE_NORMAL, NULL);
}
free (fbase);
}

```

### 7.35.2.28 void cob\_file\_sort\_init ( cob\_file \*, const int, const unsigned char \*, void \*, cob\_field \* )

Definition at line 5594 of file fileio.c.

```

{
    struct cobsort *p;

    p = cob_malloc (sizeof (struct cobsort));
    p->fnstatus = fnstatus;
    p->size = f->record_max;
    p->r_size = f->record_max + sizeof(size_t);
    p->w_size = f->record_max + sizeof(size_t) + 1;
    p->pointer = f;
    p->sort_return = sort_return;
    *(int *)sort_return = 0;
}

```



```

    p->memory = (size_t)cob_sort_memory / (p->size + sizeof(struct cobitem));

    f->file = p;
    f->keys = cob_malloc (sizeof (struct cob_file_key) * nkeys);
    f->nkeys = 0;
    if (collating_sequence) {
        f->sort_collating = collating_sequence;
    } else {
        f->sort_collating = cob_current_module->collating_sequence;
    }
    RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

### 7.35.2.29 void cob\_file\_sort\_init\_key ( cob\_file \*, const int , cob\_field \*, size\_t )

Definition at line 5621 of file fileio.c.

```

{
    f->keys[f->nkeys].flag = flag;
    f->keys[f->nkeys].field = field;
    f->keys[f->nkeys].offset = offset;
    f->nkeys++;
}

```

### 7.35.2.30 void cob\_file\_sort\_using ( cob\_file \*, cob\_file \* )

Definition at line 5524 of file fileio.c.

```

{
    int          ret;

    cob_open (data_file, COB_OPEN_INPUT, 0, NULL);
    while (1) {
        cob_read (data_file, NULL, NULL, COB_READ_NEXT);
        if (data_file->file_status[0] != '0') {
            break;
        }
        cob_copy_check (sort_file, data_file);
        ret = cob_file_sort_submit (sort_file, sort_file->record->data);
        if (ret) {
            break;
        }
    }
    cob_close (data_file, COB_CLOSE_NORMAL, NULL);
}

```

### 7.35.2.31 void cob\_open ( cob\_file \*, const int , const int , cob\_field \* )

Definition at line 3734 of file fileio.c.

```

{
    char          *p;

```

```

char          *src;
char          *dst;
size_t       i;
size_t       simple;
int          was_not_exist = 0;
struct stat   st;

f->flag_read_done = 0;

/* file was previously closed with lock */
if (f->open_mode == COB_OPEN_LOCKED) {
    RETURN_STATUS (COB_STATUS_38_CLOSED_WITH_LOCK);
}

/* file is already open */
if (f->open_mode != COB_OPEN_CLOSED) {
    RETURN_STATUS (COB_STATUS_41_ALREADY_OPEN);
}

f->last_open_mode = mode;
f->flag_nonexistent = 0;
f->flag_end_of_file = 0;
f->flag_begin_of_file = 0;
f->flag_first_read = 2;

if (f->special) {
    if (f->special == 1) {
        if (mode != COB_OPEN_INPUT) {
            RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        }
        f->file = stdin;
        f->open_mode = mode;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    } else {
        if (mode != COB_OPEN_OUTPUT) {
            RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        }
        f->file = stdout;
        f->open_mode = mode;
        RETURN_STATUS (COB_STATUS_00_SUCCESS);
    }
}

/* obtain the file name */
cob_field_to_string (f->assign, file_open_name);

#ifdef WITH_INDEX_EXTFH
if (f->organization == COB_ORG_INDEXED) {
    int    ret;

    ret = extfh_indexed_locate (f, file_open_name);
    switch (ret) {
    case COB_NOT_CONFIGURED:
        /* EXT FH requires OC to process the filename */
        break;
    case COB_STATUS_00_SUCCESS:
        /* EXT FH recognized the file */
        goto file_available;
    default:
        /* EXT FH detected an error */
        RETURN_STATUS (ret);
    }
}

```

```

    }
#endif /* WITH_INDEX_EXTFH */

#ifdef WITH_SEQRA_EXTFH
    if (f->organization != COB_ORG_INDEXED) {
        int    ret;

        ret = extfh_seqra_locate (f, file_open_name);
        switch (ret) {
        case COB_NOT_CONFIGURED:
            /* EXTFH requires OC to process the filename */
            break;
        case COB_STATUS_00_SUCCESS:
            /* EXTFH recognized the file */
            goto file_available;
        default:
            /* EXTFH detected an error */
            RETURN_STATUS (ret);
        }
    }
#endif /* WITH_SEQRA_EXTFH */

    if (cob_current_module->flag_filename_mapping) {
        src = file_open_name;
        dst = file_open_buff;
        simple = 1;
        /* expand environment variables */
        /* ex. "$TMPDIR/foo" -> "/tmp/foo" */
        while (*src) {
            if (!isalnum (*src) && *src != '_' && *src != '-') {
                simple = 0;
            }
            if (*src == '$') {
                for (i = 1; ; i++) {
                    if (!isalnum (src[i]) && src[i] != '_' &&
*src != '-') {
                        break;
                    }
                }
                memcpy (file_open_env, src + 1, i - 1);
                file_open_env[i - 1] = 0;
                if ((p = getenv (file_open_env)) != NULL) {
                    strcpy (dst, p);
                    dst += strlen (p);
                }
                src += i;
            } else {
                *dst++ = *src++;
            }
        }
        *dst = 0;
        strncpy (file_open_name, file_open_buff, COB_SMALL_MAX);

        /* resolve by environment variables */
        /* ex. "TMPFILE" -> DD_TMPFILE, dd_TMPFILE, or TMPFILE */
        if (simple) {
            for (i = 0; i < NUM_PREFIX; i++) {
                snprintf (file_open_buff, COB_SMALL_MAX, "%s%s",
                    prefix[i], file_open_name);
                if ((p = getenv (file_open_buff)) != NULL) {
                    strncpy (file_open_name, p,
COB_SMALL_MAX);

```

```

                                break;
                                }
                                }
                                if (i == NUM_PREFIX && cob_file_path) {
                                    snprintf (file_open_buff, COB_SMALL_MAX, "%s/%s",
                                                cob_file_path, file_open_name);
                                    strncpy (file_open_name, file_open_buff,
                                                COB_SMALL_MAX);
                                }
                                }
                                }
                                /* check if the file exists */
#ifdef USE_DB41
                                if (f->organization == COB_ORG_INDEXED) {
                                    if ((bdb_env && bdb_nofile (file_open_name)) ||
                                        (!bdb_env && stat (file_open_name, &st) == -1 && errno == EN
OENT)) {
                                        was_not_exist = 1;
                                        if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                        }
                                    }
                                } else if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#else
                                /* USE_DB41 */

#ifdef WITH_CISAM || defined(WITH_DISAM) || defined(WITH_VBISAM)
                                if (f->organization == COB_ORG_INDEXED) {
                                    strncpy (file_open_buff, file_open_name, COB_SMALL_MAX);
                                    strcat (file_open_buff, ".idx");
                                    if (stat (file_open_buff, &st) == -1 && errno == ENOENT) {
                                        was_not_exist = 1;
                                        if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                        }
                                    }
                                }
                                strncpy (file_open_buff, file_open_name, COB_SMALL_MAX);
                                strcat (file_open_buff, ".dat");
                                if (stat (file_open_buff, &st) == -1 && errno == ENOENT) {
                                    was_not_exist = 1;
                                    if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                        RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                    }
                                }
                                } else if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#else
                                /* WITH_CISAM || WITH_DISAM || WITH_VBISAM */
                                if (stat (file_open_name, &st) == -1 && errno == ENOENT) {
#endif
                                /* WITH_CISAM || WITH_DISAM || WITH_VBISAM */

#ifdef USE_DB41
                                was_not_exist = 1;
                                if (mode != COB_OPEN_OUTPUT && f->flag_optional == 0) {
                                    RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
                                }
                                }

#ifdef WITH_INDEX_EXTFH || defined(WITH_SEQRA_EXTFH)
                                file_available:
#endif
                                /* WITH_INDEX_EXTFH || WITH_SEQRA_EXTFH */

                                cob_cache_file (f);

```

```

    /* open the file */
#ifdef WITH_SEQRA_EXTFH
    if (f->organization != COB_ORG_INDEXED) {
        int    ret;

        ret = extfh_cob_file_open (f, file_open_name, mode, sharing);
        switch (ret) {
        case COB_STATUS_00_SUCCESS:
            f->open_mode = mode;
            break;
        case COB_STATUS_35_NOT_EXISTS:
            if (f->flag_optional) {
                f->open_mode = mode;
                f->flag_nonexistent = 1;
                f->flag_end_of_file = 1;
                f->flag_begin_of_file = 1;
                RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
            }
            break;
        }
        RETURN_STATUS (ret);
    }
#endif
#if defined(WITH_CISAM) || defined(WITH_DISAM) || defined(WITH_VBISAM)
    if (f->organization == COB_ORG_INDEXED) {
        /* Do this here to avoid mangling of the status in the 'switch' below */
        RETURN_STATUS (fileio_funcs[(int)f->organization]->open (f, file_open_name, mode, sharing));
    }
#endif
    switch (fileio_funcs[(int)f->organization]->open (f, file_open_name, mode, sharing)) {
    case 0:
        f->open_mode = mode;
        if (f->flag_optional && was_not_exist) {
            RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
        } else {
            RETURN_STATUS (COB_STATUS_00_SUCCESS);
        }
    case ENOENT:
        if (mode == COB_OPEN_EXTEND || mode == COB_OPEN_OUTPUT) {
            RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
        }
        if (f->flag_optional) {
            f->open_mode = mode;
            f->flag_nonexistent = 1;
            f->flag_end_of_file = 1;
            f->flag_begin_of_file = 1;
            RETURN_STATUS (COB_STATUS_05_SUCCESS_OPTIONAL);
        } else {
            RETURN_STATUS (COB_STATUS_35_NOT_EXISTS);
        }
    case EACCES:
    case EISDIR:
    case EROFS:
        RETURN_STATUS (COB_STATUS_37_PERMISSION_DENIED);
    case EAGAIN:
    case COB_STATUS_61_FILE_SHARING:
        RETURN_STATUS (COB_STATUS_61_FILE_SHARING);
    case COB_STATUS_91_NOT_AVAILABLE:

```

```

        RETURN_STATUS (COB_STATUS_91_NOT_AVAILABLE);
case COB_LINAGE_INVALID:
    RETURN_STATUS (COB_STATUS_57_I_O_LINAGE);
default:
    RETURN_STATUS (COB_STATUS_30_PERMANENT_ERROR);
    }
}

```

### 7.35.2.32 void cob\_read ( cob\_file \*, cob\_field \*, cob\_field \*, int )

Definition at line 4081 of file fileio.c.

```

{
    int    ret;

    f->flag_read_done = 0;

    if (unlikely(f->flag_nonexistent)) {
        if (f->flag_first_read == 0) {
            RETURN_STATUS (COB_STATUS_23_KEY_NOT_EXISTS);
        }
        f->flag_first_read = 0;
        RETURN_STATUS (COB_STATUS_10_END_OF_FILE);
    }

    /* sequential read at the end of file is an error */
    if (key == NULL) {
        if (f->flag_end_of_file && !(read_opts & COB_READ_PREVIOUS)) {
            RETURN_STATUS (COB_STATUS_46_READ_ERROR);
        }
        if (f->flag_begin_of_file && (read_opts & COB_READ_PREVIOUS)) {
            RETURN_STATUS (COB_STATUS_46_READ_ERROR);
        }
    }

    if (unlikely(f->open_mode == COB_OPEN_CLOSED
        || f->open_mode == COB_OPEN_OUTPUT
        || f->open_mode == COB_OPEN_EXTEND)) {
        RETURN_STATUS (COB_STATUS_47_INPUT_DENIED);
    }

#ifdef USE_DB41
    if (f->organization == COB_ORG_INDEXED && bdb_env != NULL) {
        if (f->open_mode != COB_OPEN_I_O ||
            (f->lock_mode & COB_LOCK_EXCLUSIVE)) {
            read_opts &= ~COB_READ_LOCK;
        } else if ((f->lock_mode & COB_LOCK_AUTOMATIC) &&
            !(read_opts & COB_READ_NO_LOCK)) {
            read_opts |= COB_READ_LOCK;
        }
    } else {
        read_opts &= ~COB_READ_LOCK;
    }
#endif
    if (key) {
        ret = fileio_funcs[(int)f->organization]->read (f, key, read_opts
    );
    } else {
        ret = fileio_funcs[(int)f->organization]->read_next (f, read_opts

```

```

);
}

switch (ret) {
case COB_STATUS_00_SUCCESS:
    f->flag_first_read = 0;
    f->flag_read_done = 1;
    f->flag_end_of_file = 0;
    f->flag_begin_of_file = 0;
    if (f->record_size && f->organization != COB_ORG_LINE_SEQUENTIAL)
    {
        cob_set_int (f->record_size, (int) f->record->size);
    }
    break;
case COB_STATUS_10_END_OF_FILE:
    if (read_opts & COB_READ_PREVIOUS) {
        f->flag_begin_of_file = 1;
    } else {
        f->flag_end_of_file = 1;
    }
    break;
}

RETURN_STATUS (ret);
}

```

### 7.35.2.33 void cob\_rewrite ( cob\_file \*, cob\_field \*, const int, cob\_field \* )

Definition at line 4201 of file fileio.c.

```

{
    int    ret;
    int    read_done = f->flag_read_done;

    f->flag_read_done = 0;

    if (unlikely(f->open_mode == COB_OPEN_CLOSED ||
        f->open_mode != COB_OPEN_I_O)) {
        RETURN_STATUS (COB_STATUS_49_I_O_DENIED);
    }

    if (f->access_mode == COB_ACCESS_SEQUENTIAL && !read_done) {
        RETURN_STATUS (COB_STATUS_43_READ_NOT_DONE);
    }

    if (f->organization == COB_ORG_SEQUENTIAL) {
        if (f->record->size != rec->size) {
            RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
        }

        if (f->record_size) {
            if (f->record->size != (size_t)cob_get_int (f->record_siz
e)) {
                RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
            }
        }
    }

}

/* RXW

```

```

#ifdef USE_DB41
    if (f->organization != COB_ORG_INDEXED || bdb_env == NULL) {
        opt &= ~COB_WRITE_LOCK;
    }
#endif
*/

    ret = fileio_funcs[(int)f->organization]->rewrite (f, opt);

    if (unlikely(cob_do_sync && ret == 0)) {
        cob_sync (f, cob_do_sync);
    }

    RETURN_STATUS (ret);
}

```

### 7.35.2.34 void cob\_rollback ( void )

Definition at line 4283 of file fileio.c.

```

{
    struct file_list    *l;

    for (l = file_cache; l; l = l->next) {
        cob_file_unlock (l->file);
    }
}

```

### 7.35.2.35 void cob\_start ( cob\_file \*, const int, cob\_field \*, cob\_field \* )

Definition at line 4052 of file fileio.c.

```

{
    int    ret;

    f->flag_read_done = 0;
    f->flag_first_read = 0;

    if (f->flag_nonexistent) {
        RETURN_STATUS (COB_STATUS_23_KEY_NOT_EXISTS);
    }

    if (f->open_mode == COB_OPEN_CLOSED
        || f->open_mode == COB_OPEN_OUTPUT
        || f->open_mode == COB_OPEN_EXTEND
        || f->access_mode == COB_ACCESS_RANDOM) {
        RETURN_STATUS (COB_STATUS_47_INPUT_DENIED);
    }

    ret = fileio_funcs[(int)f->organization]->start (f, cond, key);
    if (ret == COB_STATUS_00_SUCCESS) {
        f->flag_end_of_file = 0;
        f->flag_begin_of_file = 0;
        f->flag_first_read = 1;
    }
}

```



```

    RETURN_STATUS (ret);
}

```

### 7.35.2.36 void cob\_unlock\_file ( cob\_file \*, cob\_field \* )

Definition at line 3727 of file fileio.c.

```

{
    cob_file_unlock (f);
    RETURN_STATUS (COB_STATUS_00_SUCCESS);
}

```

### 7.35.2.37 void cob\_write ( cob\_file \*, cob\_field \*, const int, cob\_field \* )

Definition at line 4153 of file fileio.c.

```

{
    int    ret;

    f->flag_read_done = 0;

    if (f->access_mode == COB_ACCESS_SEQUENTIAL) {
        if (f->open_mode == COB_OPEN_CLOSED
            || f->open_mode == COB_OPEN_INPUT
            || f->open_mode == COB_OPEN_I_O) {
            RETURN_STATUS (COB_STATUS_48_OUTPUT_DENIED);
        }
    } else {
        if (f->open_mode == COB_OPEN_CLOSED
            || f->open_mode == COB_OPEN_INPUT
            || f->open_mode == COB_OPEN_EXTEND) {
            RETURN_STATUS (COB_STATUS_48_OUTPUT_DENIED);
        }
    }

    if (f->record_size) {
        f->record->size = cob_get_int (f->record_size);
    } else {
        f->record->size = rec->size;
    }

    if (f->record->size < f->record_min || f->record_max < f->record->size) {

        RETURN_STATUS (COB_STATUS_44_RECORD_OVERFLOW);
    }

    /* RXW
    #ifdef USE_DB41
        if (f->organization != COB_ORG_INDEXED || bdb_env == NULL) {
            opt &= ~COB_WRITE_LOCK;
        }
    #endif
    */
}

```

```
ret = fileio_funcs[(int)f->organization]->write (f, opt);

if (unlikely(cob_do_sync && ret == 0)) {
    cob_sync (f, cob_do_sync);
}

RETURN_STATUS (ret);
}
```

### 7.35.3 Variable Documentation

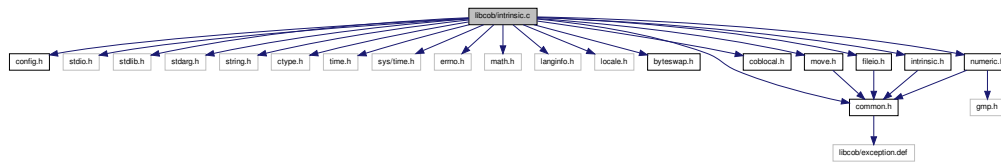
#### 7.35.3.1 DLL\_EXPIMP cob\_file\* cob\_error\_file

Definition at line 197 of file fileio.c.

## 7.36 libcob/intrinsic.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <sys/time.h>
#include <errno.h>
#include <math.h>
#include <langinfo.h>
#include <locale.h>
#include "byteswap.h"
#include "common.h"
#include "coblocal.h"
#include "move.h"
#include "numeric.h"
#include "fileio.h"
#include "intrinsic.h"
```

Include dependency graph for intrinsic.c:



## Defines

- #define [DEPTH\\_LEVEL](#) 8
- #define [COB\\_FIELD\\_INIT](#)(x, y, z)

## Functions

- [cob\\_field](#) \* [cob\\_intr\\_binop](#) ([cob\\_field](#) \*f1, int op, [cob\\_field](#) \*f2)
- [cob\\_field](#) \* [cob\\_intr\\_length](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_integer](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_integer\\_part](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_fraction\\_part](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_sign](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_upper\\_case](#) (const int offset, const int length, [cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_lower\\_case](#) (const int offset, const int length, [cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_reverse](#) (const int offset, const int length, [cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_concatenate](#) (const int offset, const int length, const int params,...)
- [cob\\_field](#) \* [cob\\_intr\\_substitute](#) (const int offset, const int length, const int params,...)
- [cob\\_field](#) \* [cob\\_intr\\_substitute\\_case](#) (const int offset, const int length, const int params,...)
- [cob\\_field](#) \* [cob\\_intr\\_trim](#) (const int offset, const int length, [cob\\_field](#) \*srcfield, const int direction)
- [cob\\_field](#) \* [cob\\_intr\\_exception\\_file](#) (void)
- [cob\\_field](#) \* [cob\\_intr\\_exception\\_location](#) (void)
- [cob\\_field](#) \* [cob\\_intr\\_exception\\_status](#) (void)
- [cob\\_field](#) \* [cob\\_intr\\_exception\\_statement](#) (void)
- [cob\\_field](#) \* [cob\\_intr\\_when\\_compiled](#) (const int offset, const int length, [cob\\_field](#) \*f)
- [cob\\_field](#) \* [cob\\_intr\\_current\\_date](#) (const int offset, const int length)
- [cob\\_field](#) \* [cob\\_intr\\_char](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_ord](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_stored\\_char\\_length](#) ([cob\\_field](#) \*srcfield)
- [cob\\_field](#) \* [cob\\_intr\\_combined\\_datetime](#) ([cob\\_field](#) \*srcdays, [cob\\_field](#) \*srctime)
- [cob\\_field](#) \* [cob\\_intr\\_date\\_of\\_integer](#) ([cob\\_field](#) \*srcdays)
- [cob\\_field](#) \* [cob\\_intr\\_day\\_of\\_integer](#) ([cob\\_field](#) \*srcdays)

- `cob_field * cob_intr_integer_of_date (cob_field *srcfield)`
- `cob_field * cob_intr_integer_of_day (cob_field *srcfield)`
- `cob_field * cob_intr_test_date_yyyymmdd (cob_field *srcfield)`
- `cob_field * cob_intr_test_day_yyyyddd (cob_field *srcfield)`
- `cob_field * cob_intr_factorial (cob_field *srcfield)`
- `cob_field * cob_intr_exp (cob_field *srcfield)`
- `cob_field * cob_intr_exp10 (cob_field *srcfield)`
- `cob_field * cob_intr_abs (cob_field *srcfield)`
- `cob_field * cob_intr_acos (cob_field *srcfield)`
- `cob_field * cob_intr_asin (cob_field *srcfield)`
- `cob_field * cob_intr_atan (cob_field *srcfield)`
- `cob_field * cob_intr_cos (cob_field *srcfield)`
- `cob_field * cob_intr_log (cob_field *srcfield)`
- `cob_field * cob_intr_log10 (cob_field *srcfield)`
- `cob_field * cob_intr_sin (cob_field *srcfield)`
- `cob_field * cob_intr_sqrt (cob_field *srcfield)`
- `cob_field * cob_intr_tan (cob_field *srcfield)`
- `cob_field * cob_intr_numval (cob_field *srcfield)`
- `cob_field * cob_intr_numval_c (cob_field *srcfield, cob_field *currency)`
- `cob_field * cob_intr_annuity (cob_field *srcfield1, cob_field *srcfield2)`
- `cob_field * cob_intr_sum (const int params,...)`
- `cob_field * cob_intr_ord_min (const int params,...)`
- `cob_field * cob_intr_ord_max (const int params,...)`
- `cob_field * cob_intr_min (const int params,...)`
- `cob_field * cob_intr_max (const int params,...)`
- `cob_field * cob_intr_midrange (const int params,...)`
- `cob_field * cob_intr_median (const int params,...)`
- `cob_field * cob_intr_mean (const int params,...)`
- `cob_field * cob_intr_mod (cob_field *srcfield1, cob_field *srcfield2)`
- `cob_field * cob_intr_range (const int params,...)`
- `cob_field * cob_intr_rem (cob_field *srcfield1, cob_field *srcfield2)`
- `cob_field * cob_intr_random (const int params,...)`
- `cob_field * cob_intr_variance (const int params,...)`
- `cob_field * cob_intr_standard_deviation (const int params,...)`
- `cob_field * cob_intr_present_value (const int params,...)`
- `cob_field * cob_intr_year_to_yyyy (const int params,...)`
- `cob_field * cob_intr_date_to_yyyymmdd (const int params,...)`
- `cob_field * cob_intr_day_to_yyyyddd (const int params,...)`
- `cob_field * cob_intr_seconds_past_midnight (void)`
- `cob_field * cob_intr_seconds_from_formatted_time (cob_field *format, cob_field *value)`
- `cob_field * cob_intr_locale_date (const int offset, const int length, cob_field *srcfield, cob_field *locale_field)`
- `cob_field * cob_intr_locale_time (const int offset, const int length, cob_field *srcfield, cob_field *locale_field)`
- `cob_field * cob_intr_lcl_time_from_secs (const int offset, const int length, cob_field *srcfield, cob_field *locale_field)`
- `void cob_init_intrinsic (void)`

## Variables

- static `cob_decimal d2`
- static `cob_decimal d3`
- static `cob_decimal d4`
- static `cob_decimal d5`

### 7.36.1 Define Documentation

#### 7.36.1.1 #define COB\_FIELD\_INIT( x, y, z )

##### Value:

```
do { \
    field.size = x; \
    field.data = y; \
    field.attr = z; \
} while (0)
```

Definition at line 59 of file intrinsic.c.

#### 7.36.1.2 #define DEPTH\_LEVEL 8

Definition at line 57 of file intrinsic.c.

### 7.36.2 Function Documentation

#### 7.36.2.1 void cob\_init\_intrinsic ( void )

Definition at line 3269 of file intrinsic.c.

```
{
    size_t      i;

    cob_decimal_init (&d1);
    cob_decimal_init (&d2);
    cob_decimal_init (&d3);
    cob_decimal_init (&d4);
    cob_decimal_init (&d5);
    /* mpz_init2 (mp, 256); */
    memset ((char *)&calc_field[0], 0, sizeof (calc_field));
    memset ((char *)&calc_attr[0], 0, sizeof (calc_attr));
    for (i = 0; i < DEPTH_LEVEL; ++i) {
        calc_field[i].data = cob_malloc (256);
        calc_field[i].size = 256;
        calc_size[i] = 256;
    }
    locale_buff = cob_malloc (COB_SMALL_BUFF);
}
```

**7.36.2.2 cob\_field\* cob\_intr\_abs ( cob\_field \* srcfield )**

Definition at line 1545 of file intrinsic.c.

```
{
    make_field_entry (srcfield);

    cob_decimal_set_field (&d1, srcfield);
    mpz_abs (d1.value, d1.value);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}
```

**7.36.2.3 cob\_field\* cob\_intr\_acos ( cob\_field \* srcfield )**

Definition at line 1557 of file intrinsic.c.

```
{
    unsigned long long    result;
    double                mathd2;
    int                   i, tempres;
    cob_field_attr        attr;
    cob_field             field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = acos (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    result = (unsigned long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}
```

**7.36.2.4 cob\_field\* cob\_intr\_annuity ( cob\_field \* srcfield1, cob\_field \* srcfield2 )**

Definition at line 1974 of file intrinsic.c.

```

{
    double          mathd1, mathd2;

    make_double_entry ();

    cob_decimal_set_field (&d1, srcfield1);
    cob_decimal_set_field (&d2, srcfield2);

    mathd1 = intr_get_double (&d1);
    mathd2 = intr_get_double (&d2);
    if (mathd1 == 0) {
        mathd1 = 1.0 / mathd2;
        memcpy (curr_field->data, (char *)&mathd1, sizeof (double));
        return curr_field;
    }
    mathd1 /= (1.0 - pow (mathd1 + 1.0, 0.0 - mathd2));
    memcpy (curr_field->data, (char *)&mathd1, sizeof (double));
    return curr_field;
}

```

### 7.36.2.5 cob\_field\* cob\_intr\_asin ( cob\_field \* srcfield )

Definition at line 1591 of file intrinsic.c.

```

{
    long long          result;
    double             mathd2;
    int                i, tempres;
    cob_field_attr     attr;
    cob_field          field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = asin (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

7.36.2.6 `cob_field*cob_intr_atan ( cob_field * srcfield )`

Definition at line 1624 of file intrinsic.c.

```

{
    long long          result;
    double            mathd2;
    int               i, tempres;
    cob_field_attr    attr;
    cob_field         field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = atan (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

7.36.2.7 `cob_field*cob_intr_binop ( cob_field * f1, int op, cob_field * f2 )`

Definition at line 432 of file intrinsic.c.

```

{
/* RXW
    size_t          bitnum;
    size_t          sign;
    size_t          attrsign;
    cob_field_attr  attr;
    cob_field       field;
*/

    cob_decimal_set_field (&d1, f1);
    cob_decimal_set_field (&d2, f2);
    switch (op) {
    case '+':
        cob_decimal_add (&d1, &d2);
        break;
    case '-':
        cob_decimal_sub (&d1, &d2);
        break;

```



```

    case '*':
        cob_decimal_mul (&d1, &d2);
        break;
    case '/':
        cob_decimal_div (&d1, &d2);
        break;
    case '^':
        cob_decimal_pow (&d1, &d2);
        break;
    default:
        break;
}

/* RXW
if (mpz_sgn (dl.value) < 0) {
    attrsign = COB_FLAG_HAVE_SIGN;
    sign = 1;
} else {
    attrsign = 0;
    sign = 0;
}
bitnum = mpz_sizeinbase (dl.value, 2);
if (bitnum < 33 - sign) {
    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, attrsign, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    attr.scale = dl.scale;
    make_field_entry (&field);
} else if (bitnum < 65 - sign) {
    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, attrsign, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    attr.scale = dl.scale;
    make_field_entry (&field);
} else {
    make_double_entry ();
}
/* RXW
*/
cob_decimal_get_field (&d1, curr_field, 0);

return curr_field;
}

```

### 7.36.2.8 cob\_field\* cob\_intr\_char( cob\_field \* srcfield )

Definition at line 1121 of file intrinsic.c.

```

{
    int          i;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (1, NULL, &attr);
    make_field_entry (&field);

    i = cob_get_int (srcfield);
    if (i < 1 || i > 256) {
        *curr_field->data = 0;
    }
}

```

```

    } else {
        *curr_field->data = i - 1;
    }
    return curr_field;
}

```

### 7.36.2.9 **cob\_field\*** **cob\_intr\_combined\_datetime** ( **cob\_field** \* *srcdays*, **cob\_field** \* *srctime* )

Definition at line 1178 of file intrinsic.c.

```

{
    int            srdays;
    int            srctime;
    cob_field_attr attr;
    cob_field      field;
    char           buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 12, 5, 0, NULL);
    COB_FIELD_INIT (12, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    srdays = cob_get_int (srcdays);
    if (srdays < 1 || srdays > 3067671) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 12);
        return curr_field;
    }
    srctime = cob_get_int (srctime);
    if (srctime < 1 || srctime > 86400) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 12);
        return curr_field;
    }
    snprintf (buff, 15, "%7.7d%5.5d", srdays, srctime);
    memcpy (curr_field->data, buff, 12);
    return curr_field;
}

```

### 7.36.2.10 **cob\_field\*** **cob\_intr\_concatenate** ( **const int** *offset*, **const int** *length*, **const int** *params*, ... )

Definition at line 639 of file intrinsic.c.

```

{
    cob_field      **f;
    unsigned char  *p;
    size_t         calcsize;
    int            i;
    cob_field_attr attr;
    cob_field      field;
    va_list        args;

    f = cob_malloc (params * sizeof (cob_field *));

```

```

va_start (args, params);

/* Extract args / calculate size */
calcsize = 0;
for (i = 0; i < params; ++i) {
    f[i] = va_arg (args, cob_field *);
    calcsize += f[i]->size;
}

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (calcsize, NULL, &attr);
make_field_entry (&field);

p = curr_field->data;
for (i = 0; i < params; ++i) {
    memcpy (p, f[i]->data, f[i]->size);
    p += f[i]->size;
}

if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
free (f);
return curr_field;
}

```

### 7.36.2.11 `cob_field* cob_intr_cos ( cob_field * srcfield )`

Definition at line 1657 of file intrinsic.c.

```

{
    long long          result;
    double             mathd2;
    int                i, tempres;
    cob_field_attr     attr;
    cob_field          field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = cos (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
}

```

```

        memcpy (curr_field->data, (char *)&result, 8);
        return curr_field;
    }

```

### 7.36.2.12 cob\_field\* cob\_intr\_current\_date ( const int offset, const int length )

Definition at line 1032 of file intrinsic.c.

```

{
    #if defined(_WIN32) && !defined(__CYGWIN__)
        long          contz;
        struct tm      *tmptr;
        struct _timeb  tmb;
        cob_field_attr attr;
        cob_field      field;
    #else
    #if !defined(__linux__) && !defined(__CYGWIN__) && !defined(COB_STRFTIME) && defi
        ned(HAVE_TIMEZONE)
        struct tm      *tmptr;
        long          contz;
    #endif
        time_t        curtime;
        cob_field_attr attr;
        cob_field      field;
    #if defined(HAVE_SYS_TIME_H) && defined(HAVE_GETTIMEOFDAY)
        struct timeval tmv;
        char          buff2[8];
    #endif
    #endif
    /* _WIN32 */
    char          buff[24];

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (21, NULL, &attr);
    make_field_entry (&field);
    memset (buff, 0, sizeof(buff));

    #if defined(_WIN32) && !defined(__CYGWIN__)
        _ftime (&tmb);
        tmptr = localtime (&(tmb.time));
        if (tmb.timezone <= 0) {
            contz = -tmb.timezone;
            snprintf (buff, 23,
                "%4.4d%2.2d%2.2d%2.2d%2.2d%2.2d%2.2d%2.2d+%2.21d%2.21d",
                tmptr->tm_year + 1900, tmptr->tm_mon + 1, tmptr->tm_mday,

                tmptr->tm_hour, tmptr->tm_min, tmptr->tm_sec,
                tmb.millitm / 100, contz / 60, contz % 60);
        } else {
            contz = tmb.timezone;
            snprintf (buff, 23,
                "%4.4d%2.2d%2.2d%2.2d%2.2d%2.2d%2.2d%2.2d-%2.21d%2.21d",
                tmptr->tm_year + 1900, tmptr->tm_mon + 1, tmptr->tm_mday,

                tmptr->tm_hour, tmptr->tm_min, tmptr->tm_sec,
                tmb.millitm / 100, contz / 60, contz % 60);
        }
    #else
    #if defined(HAVE_SYS_TIME_H) && defined(HAVE_GETTIMEOFDAY)
        gettimeofday (&tmv, NULL);
    #endif

```

```

        curtime = tmv.tv_sec;
#else
        curtime = time (NULL);
#endif

#if defined(__linux__) || defined(__CYGWIN__) || defined(COB_STRFTIME)
    strftime (buff, 22, "%Y%m%d%H%M%S00%z", localtime (&curtime));
#elif defined(HAVE_TIMEZONE)
    tmptr = localtime (&curtime);
    strftime (buff, 17, "%Y%m%d%H%M%S00", tmptr);
    /* RXW - Hack for DST - Need something better */
    if (tmptr->tm_isdst > 0) {
        timezone -= 3600;
    }
    if (timezone <= 0) {
        contz = -timezone;
        buff[16] = '+';
    } else {
        contz = timezone;
        buff[16] = '-';
    }
    sprintf(&buff[17], "%2.2ld%2.2ld", contz / 3600, (contz % 3600) / 60);
#else
    strftime (buff, 22, "%Y%m%d%H%M%S0000000", localtime (&curtime));
#endif

#if defined(HAVE_SYS_TIME_H) && defined(HAVE_GETTIMEOFDAY)
    snprintf(buff2, 7, "%2.2ld", tmv.tv_usec / 10000);
    memcpy (&buff[14], buff2, 2);
#endif
#endif /* _WIN32 */

    memcpy (curr_field->data, buff, 21);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

### 7.36.2.13 cob\_field\* cob\_intr\_date\_of\_integer ( cob\_field \* srcdays )

Definition at line 1209 of file intrinsic.c.

```

{
    int            i;
    int            days;
    int            baseyear = 1601;
    int            leapyear = 365;
    cob_field_attr attr;
    cob_field      field;
    char           buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 8, 0, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    /* Base 1601-01-01 */
    days = cob_get_int (srcdays);
}

```

```

if (days < 1 || days > 3067671) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    memset (curr_field->data, '0', 8);
    return curr_field;
}
while (days > leapyear) {
    days -= leapyear;
    ++baseyear;
    if (leap_year (baseyear)) {
        leapyear = 366;
    } else {
        leapyear = 365;
    }
}
for (i = 0; i < 13; ++i) {
    if (leap_year (baseyear)) {
        if (days <= leap_days[i]) {
            days -= leap_days[i-1];
            break;
        }
    } else {
        if (days <= normal_days[i]) {
            days -= normal_days[i-1];
            break;
        }
    }
}
snprintf (buff, 15, "%4.4d%2.2d%2.2d", baseyear, i, days);
memcpy (curr_field->data, buff, 8);
return curr_field;
}

```

#### 7.36.2.14 cob\_field\* cob\_intr\_date\_to\_yyyymmdd ( const int *params*, ... )

Definition at line 2646 of file intrinsic.c.

```

{
    cob_field      *f;
    struct tm      *timeptr;
    va_list        args;
    time_t         t;
    int            year;
    int            mmdd;
    int            interval;
    int            xqtyear;
    int            maxyear;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    va_start (args, params);
    f = va_arg (args, cob_field *);
    year = cob_get_int (f);
    mmdd = year % 10000;
    year /= 10000;
}

```

```

if (params > 1) {
    f = va_arg (args, cob_field *);
    interval = cob_get_int (f);
} else {
    interval = 50;
}
if (params > 2) {
    f = va_arg (args, cob_field *);
    xqtyear = cob_get_int (f);
} else {
    t = time (NULL);
    timeptr = localtime (&t);
    xqtyear = 1900 + timeptr->tm_year;
}
va_end (args);

if (year < 0 || year > 999999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (xqtyear < 1601 || xqtyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
maxyear = xqtyear + interval;
if (maxyear < 1700 || maxyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (maxyear % 100 >= year) {
    year += 100 * (maxyear / 100);
} else {
    year += 100 * ((maxyear / 100) - 1);
}
year *= 10000;
year += mmdd;
cob_set_int (curr_field, year);
return curr_field;
}

```

#### 7.36.2.15 `cob_field* cob_intr_day_of_integer ( cob_field * srcdays )`

Definition at line 1259 of file intrinsic.c.

```

{
    int          days;
    int          baseyear = 1601;
    int          leapyear = 365;
    cob_field_attr attr;
    cob_field    field;
    char         buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 7, 0, 0, NULL);
    COB_FIELD_INIT (7, NULL, &attr);
    make_field_entry (&field);
}

```

```

cob_exception_code = 0;
/* Base 1601-01-01 */
days = cob_get_int (srcdays);
if (days < 1 || days > 3067671) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    memset (curr_field->data, '0', 7);
    return curr_field;
}
while (days > leapyear) {
    days -= leapyear;
    ++baseyear;
    if (leap_year (baseyear)) {
        leapyear = 366;
    } else {
        leapyear = 365;
    }
}
snprintf (buff, 15, "%4.4d%3.3d", baseyear, days);
memcpy (curr_field->data, buff, 7);
return curr_field;
}

```

### 7.36.2.16 `cob_field* cob_intr_day_to_yyyyddd ( const int params, ... )`

Definition at line 2714 of file `intrinsic.c`.

```

{
    cob_field      *f;
    struct tm      *timeptr;
    va_list        args;
    time_t         t;
    int            year;
    int            days;
    int            interval;
    int            xqtyear;
    int            maxyear;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    va_start (args, params);
    f = va_arg (args, cob_field *);
    year = cob_get_int (f);
    days = year % 1000;
    year /= 1000;
    if (params > 1) {
        f = va_arg (args, cob_field *);
        interval = cob_get_int (f);
    } else {
        interval = 50;
    }
    if (params > 2) {
        f = va_arg (args, cob_field *);
        xqtyear = cob_get_int (f);
    } else {

```



```

        t = time (NULL);
        timeptr = localtime (&t);
        xqtyear = 1900 + timeptr->tm_year;
    }
    va_end (args);

    if (year < 0 || year > 999999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    if (xqtyear < 1601 || xqtyear > 9999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    maxyear = xqtyear + interval;
    if (maxyear < 1700 || maxyear > 9999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    if (maxyear % 100 >= year) {
        year += 100 * (maxyear / 100);
    } else {
        year += 100 * ((maxyear / 100) - 1);
    }
    year *= 1000;
    year += days;
    cob_set_int (curr_field, year);
    return curr_field;
}

```

#### 7.36.2.17 cob\_field\* cob\_intr\_exception\_file ( void )

Definition at line 916 of file intrinsic.c.

```

{
    size_t      flen;
    cob_field_attr attr;
    cob_field   field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    if (cob_exception_code == 0 || !cob_error_file ||
        (cob_exception_code & 0x0500) != 0x0500) {
        field.size = 2;
        make_field_entry (&field);
        memcpy (curr_field->data, "00", 2);
    } else {
        flen = strlen (cob_error_file->select_name);
        field.size = flen + 2;
        make_field_entry (&field);
        memcpy (curr_field->data, cob_error_file->file_status, 2);
        memcpy (&(curr_field->data[2]), cob_error_file->select_name, flen
    );
    }
    return curr_field;
}

```

7.36.2.18 `cob_field*` `cob_intr_exception.location ( void )`Definition at line 940 of file `intrinsic.c`.

```

{
    cob_field_attr  attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    if (!cob_got_exception || !cob_orig_program_id) {
        field.size = 1;
        make_field_entry (&field);
        *(curr_field->data) = ' ';
        return curr_field;
    }
    memset (locale_buff, 0, COB_SMALL_BUFF);
    if (cob_orig_section && cob_orig_paragraph) {
        sprintf (locale_buff, COB_SMALL_MAX, "%s; %s OF %s; %d",
                cob_orig_program_id, cob_orig_paragraph,
                cob_orig_section, cob_orig_line);
    } else if (cob_orig_section) {
        sprintf (locale_buff, COB_SMALL_MAX, "%s; %s; %d",
                cob_orig_program_id, cob_orig_section, cob_orig_line);
    } else if (cob_orig_paragraph) {
        sprintf (locale_buff, COB_SMALL_MAX, "%s; %s; %d",
                cob_orig_program_id, cob_orig_paragraph, cob_orig_line);
    } else {
        sprintf (locale_buff, COB_SMALL_MAX, "%s; ; %d",
                cob_orig_program_id, cob_orig_line);
    }
    field.size = strlen (locale_buff);
    make_field_entry (&field);
    memcpy (curr_field->data, locale_buff, field.size);
    return curr_field;
}

```

7.36.2.19 `cob_field*` `cob_intr_exception.statement ( void )`Definition at line 997 of file `intrinsic.c`.

```

{
    size_t      flen;
    cob_field_attr  attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (31, NULL, &attr);
    make_field_entry (&field);

    memset (curr_field->data, ' ', 31);
    if (cob_exception_code && cob_orig_statement) {
        flen = strlen (cob_orig_statement);
        if (flen > 31) {
            memcpy (curr_field->data, cob_orig_statement, 31);
        } else {
            memcpy (curr_field->data, cob_orig_statement, flen);
        }
    }
}

```

```

    }
  }
  return curr_field;
}

```

### 7.36.2.20 `cob_field* cob_intr_exception_status ( void )`

Definition at line 975 of file intrinsic.c.

```

{
  const char      *except_name;
  cob_field_attr  attr;
  cob_field       field;

  COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
  COB_FIELD_INIT (31, NULL, &attr);
  make_field_entry (&field);

  memset (curr_field->data, ' ', 31);
  if (cob_exception_code) {
    except_name = cob_get_exception_name (cob_exception_code);
    if (except_name == NULL) {
      except_name = "EXCEPTION-OBJECT";
    }
    memcpy (curr_field->data, except_name, strlen (except_name));
  }
  return curr_field;
}

```

### 7.36.2.21 `cob_field* cob_intr_exp ( cob_field * srcfield )`

Definition at line 1509 of file intrinsic.c.

```

{
  double          mathd2;

  cob_decimal_set_field (&d1, srcfield);
  make_double_entry ();

  errno = 0;
  mathd2 = pow (2.7182818284590452354, intr_get_double (&d1));
  if (errno) {
    cob_set_int (curr_field, 0);
    return curr_field;
  }
  memcpy (curr_field->data, (char *)&mathd2, 8);
  return curr_field;
}

```

### 7.36.2.22 `cob_field* cob_intr_exp10 ( cob_field * srcfield )`

Definition at line 1527 of file intrinsic.c.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = pow (10.0, intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

### 7.36.2.23 `cob_field* cob_intr_factorial ( cob_field * srcfield )`

Definition at line 1485 of file intrinsic.c.

```

{
    int             srcval;
    cob_field_attr attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    srcval = cob_get_int (srcfield);
    if (srcval < 0) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    dl.scale = 0;
    mpz_fac_ui (dl.value, (unsigned int)srcval);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

### 7.36.2.24 `cob_field* cob_intr_fraction_part ( cob_field * srcfield )`

Definition at line 552 of file intrinsic.c.

```

{
    cob_field_attr attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 18, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);
}

```

```

    cob_move (srcfield, curr_field);
    return curr_field;
}

```

### 7.36.2.25 cob\_field\* cob\_intr\_integer ( cob\_field \* srcfield )

Definition at line 511 of file intrinsic.c.

```

{
    int          i, scale;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_decimal_set_field (&dl, srcfield);
    if (mpz_sgn (dl.value) >= 0) {
        cob_decimal_get_field (&dl, curr_field, 0);
        return curr_field;
    }
    scale = 1;
    for (i = 0; i < dl.scale; ++i) {
        scale *= 10;
    }
    if (mpz_fdiv_ui (dl.value, (unsigned int)scale)) {
        mpz_sub_ui (dl.value, dl.value, (unsigned int)scale);
    }
    cob_decimal_get_field (&dl, curr_field, 0);
    return curr_field;
}

```

### 7.36.2.26 cob\_field\* cob\_intr\_integer\_of\_date ( cob\_field \* srcfield )

Definition at line 1295 of file intrinsic.c.

```

{
    int          indate;
    int          days;
    int          totaldays;
    int          month;
    int          year;
    int          baseyear = 1601;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    /* Base 1601-01-01 */
    indate = cob_get_int (srcfield);
}

```

```

year = indate / 10000;
if (year < 1601 || year > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
indate %= 10000;
month = indate / 100;
if (month < 1 || month > 12) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
days = indate % 100;
if (days < 1 || days > 31) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (leap_year (year)) {
    if (days > leap_month_days[month]) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
} else {
    if (days > normal_month_days[month]) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
}
totaldays = 0;
while (baseyear != year) {
    if (leap_year (baseyear)) {
        totaldays += 366;
    } else {
        totaldays += 365;
    }
    ++baseyear;
}
if (leap_year (baseyear)) {
    totaldays += leap_days[month - 1];
} else {
    totaldays += normal_days[month - 1];
}
totaldays += days;
cob_set_int (curr_field, totaldays);
return curr_field;
}

```

### 7.36.2.27 cob\_field\* cob\_intr\_integer\_of\_day ( cob\_field \* srcfield )

Definition at line 1365 of file intrinsic.c.

```

{
    int            indate;
    int            days;
    int            totaldays;
}

```

```

int          year;
int          baseyear = 1601;
cob_field_attr attr;
cob_field    field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
COB_FIELD_INIT (4, NULL, &attr);
make_field_entry (&field);

cob_exception_code = 0;
/* Base 1601-01-01 */
indate = cob_get_int (srcfield);
year = indate / 1000;
if (year < 1601 || year > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
days = indate % 1000;
if (days < 1 || days > 365 + leap_year (year)) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
totaldays = 0;
while (baseyear != year) {
    if (leap_year (baseyear)) {
        totaldays += 366;
    } else {
        totaldays += 365;
    }
    ++baseyear;
}
totaldays += days;
cob_set_int (curr_field, totaldays);
return curr_field;
}

```

### 7.36.2.28 `cob_field* cob_intr_integer_part ( cob_field * srcfield )`

Definition at line 538 of file intrinsic.c.

```

{
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_move (srcfield, curr_field);
    return curr_field;
}

```

### 7.36.2.29 `cob_field* cob_intr_lcl_time_from_secs ( const int offset, const int length, cob_field * srcfield, cob_field * locale_field )`

Definition at line 3149 of file intrinsic.c.

```

{
    cob_field_attr attr;
    cob_field field;
#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    size_t len;
    int indate;
    int hours;
    int minutes;
    int seconds;
#elseif defined HAVE_LANGINFO_CODESET
    char *deflocale = NULL;
    char *localep = NULL;
    char *localep2;
    struct tm tstruct;
    char buff2[128];
#else
    char *p;
    LCID localeid = LOCALE_USER_DEFAULT;
    SYSTEMTIME syst;
#endif
    char buff[128];
#endif

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    cob_exception_code = 0;

#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    if (COB_FIELD_IS_NUMERIC (srcfield)) {
        indate = cob_get_int (srcfield);
    } else {
        goto derror;
    }
    if (indate > 86400) {
        goto derror;
    }
    hours = indate / 3600;
    indate %= 3600;
    minutes = indate / 60;
    seconds = indate % 60;
#elseif defined HAVE_LANGINFO_CODESET
    memset ((void *)&tstruct, 0, sizeof(struct tm));
    tstruct.tm_hour = hours;
    tstruct.tm_min = minutes;
    tstruct.tm_sec = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFF) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        deflocale = locale_buff;
        localep2 = setlocale (LC_TIME, NULL);
        if (localep2) {
            localep = strdup (localep2);
        }
    }
#endif

```



```

        (void) setlocale (LC_TIME, deflocale);
    }
    memset (buff2, 0, sizeof(buff2));
    snprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(T_FMT));
    if (deflocale) {
        if (localep) {
            (void) setlocale (LC_TIME, localep);
        }
    }
    strftime (buff, sizeof(buff), buff2, &tstruct);
#else
    memset ((void *)&syst, 0, sizeof(syst));
    syst.wHour = hours;
    syst.wMinute = minutes;
    syst.wSecond = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFF) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        for (p = locale_buff; *p; ++p) {
            if (isalnum(*p) || *p == '_') {
                continue;
            }
            break;
        }
        *p = 0;
        for (len = 0; len < WINLOCSIZE; ++len) {
            if (!strcmp(locale_buff, wintable[len].winlocalename)) {
                localeid = wintable[len].winlocaleid;
                break;
            }
        }
        if (len == WINLOCSIZE) {
            goto derror;
        }
    }
    if (!GetTimeFormat (localeid, LOCALE_NOUSEROVERRIDE, &syst, NULL, buff, sizeof(buff))) {
        goto derror;
    }
#endif
    len = strlen (buff);
    field.size = len;
    make_field_entry (&field);
    memcpy (curr_field->data, buff, len);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
derror:
#endif
    field.size = 10;
    make_field_entry (&field);
    memset (curr_field->data, ' ', 10);
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    return curr_field;
}

```

**7.36.2.30** `cob_field* cob_intr_length ( cob_field * srcfield )`

Definition at line 497 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, (int)srcfield->size);
    return curr_field;
}

```

**7.36.2.31** `cob_field* cob_intr_locale_date ( const int offset, const int length, cob_field * srcfield, cob_field * locale.field )`

Definition at line 2866 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;
#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    size_t        len;
    int           indate;
    int           days;
    int           month;
    int           year;
#else
    HAVE_LANGINFO_CODESET
    unsigned char *p;
    char          *deflocale = NULL;
    char          *localep = NULL;
    char          *localep2;
    struct tm     tstruct;
    char          buff2[128];
#endif
    char          *p;
    LCID          localeid = LOCALE_USER_DEFAULT;
    SYSTEMTIME    syst;
#endif
    char          buff[128];
#endif

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    cob_exception_code = 0;

#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    if (COB_FIELD_IS_NUMERIC (srcfield)) {
        indate = cob_get_int (srcfield);
    } else {
        if (srcfield->size < 8) {
            goto derror;
        }
        p = srcfield->data;
        indate = 0;

```

```

        for (len = 0; len < 8; ++len, ++p) {
            if (isdigit (*p)) {
                indate *= 10;
                indate += (*p - '0');
            } else {
                goto derror;
            }
        }
    }
    year = indate / 10000;
    if (year < 1601 || year > 9999) {
        goto derror;
    }
    indate %= 10000;
    month = indate / 100;
    if (month < 1 || month > 12) {
        goto derror;
    }
    days = indate % 100;
    if (days < 1 || days > 31) {
        goto derror;
    }
    if (leap_year (year)) {
        if (days > leap_month_days[month]) {
            goto derror;
        }
    } else {
        if (days > normal_month_days[month]) {
            goto derror;
        }
    }
}
#ifdef HAVE_LANGINFO_CODESET
month--;

memset ((void *)&tstruct, 0, sizeof(struct tm));
tstruct.tm_year = year - 1900;
tstruct.tm_mon = month;
tstruct.tm_mday = days;
if (locale_field) {
    if (locale_field->size >= COB_SMALL_BUFF) {
        goto derror;
    }
    cob_field_to_string (locale_field, locale_buff);
    deflocale = locale_buff;
    localep2 = setlocale (LC_TIME, NULL);
    if (localep2) {
        localep = strdup (localep2);
    }
    (void) setlocale (LC_TIME, deflocale);
}
memset (buff2, 0, sizeof(buff2));
snprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(D_FMT));
if (deflocale) {
    if (localep) {
        (void) setlocale (LC_TIME, localep);
    }
}
strftime (buff, sizeof(buff), buff2, &tstruct);
#else
memset ((void *)&syst, 0, sizeof(syst));
syst.wYear = year;
syst.wMonth = month;

```

```

syst.wDay = days;
if (locale_field) {
    if (locale_field->size >= COB_SMALL_BUFF) {
        goto derror;
    }
    cob_field_to_string (locale_field, locale_buff);
    for (p = locale_buff; *p; ++p) {
        if (isalnum(*p) || *p == '_' ) {
            continue;
        }
        break;
    }
    *p = 0;
    for (len = 0; len < WINLOCSIZE; ++len) {
        if (!strcmp(locale_buff, wintable[len].winlocalename)) {
            localeid = wintable[len].winlocaleid;
            break;
        }
    }
    if (len == WINLOCSIZE) {
        goto derror;
    }
}
if (!GetDateFormat (localeid, DATE_SHORTDATE, &syst, NULL, buff, sizeof(buff))) {
    goto derror;
}
#endif
len = strlen (buff);
field.size = len;
make_field_entry (&field);
memcpy (curr_field->data, buff, len);
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
return curr_field;
derror:
#endif
field.size = 10;
make_field_entry (&field);
memset (curr_field->data, ' ', 10);
cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
return curr_field;
}

```

### 7.36.2.32 **cob\_field\*** **cob\_intr\_locale\_time** ( **const int** *offset*, **const int** *length*, **cob\_field** \**srcfield*, **cob\_field** \**locale.field* )

Definition at line 3012 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;
#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    size_t        len;
    int           indate;
    int           hours;
    int           minutes;
    int           seconds;

```

```

#ifdef HAVE_LANGINFO_CODESET
    unsigned char *p;
    char *deflocale = NULL;
    char *localep = NULL;
    char *localep2;
    struct tm tstruct;
    char buff2[128];
#else
    char *p;
    LCID localeid = LOCALE_USER_DEFAULT;
    SYSTEMTIME syst;
#endif
char buff[128];
#endif

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
cob_exception_code = 0;

#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    if (COB_FIELD_IS_NUMERIC (srcfield)) {
        indate = cob_get_int (srcfield);
    } else {
        if (srcfield->size < 6) {
            goto derror;
        }
        p = srcfield->data;
        indate = 0;
        for (len = 0; len < 6; ++len, ++p) {
            if (isdigit (*p)) {
                indate *= 10;
                indate += (*p - '0');
            } else {
                goto derror;
            }
        }
        hours = indate / 10000;
        if (hours < 0 || hours > 24) {
            goto derror;
        }
        indate %= 10000;
        minutes = indate / 100;
        if (minutes < 0 || minutes > 59) {
            goto derror;
        }
        seconds = indate % 100;
        if (seconds < 0 || seconds > 59) {
            goto derror;
        }
    }
#endif

#ifdef HAVE_LANGINFO_CODESET
    memset ((void *)&tstruct, 0, sizeof(struct tm));
    tstruct.tm_hour = hours;
    tstruct.tm_min = minutes;
    tstruct.tm_sec = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFF) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        deflocale = locale_buff;
    }

```

```

        localep2 = setlocale (LC_TIME, NULL);
        if (localep2) {
            localep = strdup (localep2);
        }
        (void) setlocale (LC_TIME, deflocale);
    }
    memset (buff2, 0, sizeof(buff2));
    snprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(T_FMT));
    if (deflocale) {
        if (localep) {
            (void) setlocale (LC_TIME, localep);
        }
    }
    strftime (buff, sizeof(buff), buff2, &tstruct);
#else
    memset ((void *)&syst, 0, sizeof(syst));
    syst.wHour = hours;
    syst.wMinute = minutes;
    syst.wSecond = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFF) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        for (p = locale_buff; *p; ++p) {
            if (isalnum(*p) || *p == '_') {
                continue;
            }
            break;
        }
        *p = 0;
        for (len = 0; len < WINLOCSIZE; ++len) {
            if (!strcmp(locale_buff, wintable[len].winlocalename)) {
                localeid = wintable[len].winlocaleid;
                break;
            }
        }
        if (len == WINLOCSIZE) {
            goto derror;
        }
    }
    if (!GetTimeFormat (localeid, LOCALE_NOUSEROVERRIDE, &syst, NULL, buff, sizeof(buff))) {
        goto derror;
    }
#endif
    len = strlen (buff);
    field.size = len;
    make_field_entry (&field);
    memcpy (curr_field->data, buff, len);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
derror:
#endif
    field.size = 10;
    make_field_entry (&field);
    memset (curr_field->data, ' ', 10);
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    return curr_field;

```

```
}

```

### 7.36.2.33 `cob_field* cob_intr_log ( cob_field * srcfield )`

Definition at line 1690 of file intrinsic.c.

```
{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = log (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}
```

### 7.36.2.34 `cob_field* cob_intr_log10 ( cob_field * srcfield )`

Definition at line 1708 of file intrinsic.c.

```
{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = log10 (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}
```

### 7.36.2.35 `cob_field* cob_intr_lower_case ( const int offset, const int length, cob_field * srcfield )`

Definition at line 605 of file intrinsic.c.

```
{
    size_t          i, size;

    make_field_entry (srcfield);

```

```

size = srcfield->size;
for (i = 0; i < size; ++i) {
    curr_field->data[i] = tolower (srcfield->data[i]);
}
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
return curr_field;
}

```

### 7.36.2.36 `cob_field*` `cob_intr_max ( const int params, ... )`

Definition at line 2125 of file `intrinsic.c`.

```

{
    cob_field      *f, *basef;
    va_list        args;
    int            i;

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) > 0) {
            basef = f;
        }
    }
    va_end (args);

    return basef;
}

```

### 7.36.2.37 `cob_field*` `cob_intr_mean ( const int params, ... )`

Definition at line 2224 of file `intrinsic.c`.

```

{
    cob_field      *f;
    va_list        args;
    long long      n;
    union {
        unsigned char  data[8];
        long long      datall;
    } datun;
    int            i;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    mpz_set_ui (dl.value, 0);
}

```



```

d1.scale = 0;

va_start (args, params);
for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_add (&d1, &d2);
}
va_end (args);

mpz_set_ui (d2.value, (unsigned int)params);
d2.scale = 0;
cob_decimal_div (&d1, &d2);
field.data = datun.data;
cob_decimal_get_field (&d1, &field, 0);
n = datun.datall;
for (i = 0; n; n /= 10, ++i) ;
field.data = NULL;
if (i <= 18) {
    attr.scale = 18 - i;
}
make_field_entry (&field);
cob_decimal_get_field (&d1, curr_field, 0);
return curr_field;
}

```

### 7.36.2.38 cob\_field\* cob\_intr\_median ( const int *params*, ... )

Definition at line 2179 of file intrinsic.c.

```

{
    cob_field    *f;
    cob_field    **field_alloc;
    va_list      args;
    int          i;

    va_start (args, params);

    f = va_arg (args, cob_field *);
    if (params == 1) {
        va_end (args);
        return f;
    }

    field_alloc = cob_malloc (params * sizeof (cob_field *));
    field_alloc[0] = f;

    for (i = 1; i < params; ++i) {
        field_alloc[i] = va_arg (args, cob_field *);
    }
    va_end (args);

    qsort (field_alloc, (size_t)params, (size_t)sizeof (cob_field *), comp_fi
eld);

    i = params / 2;
    if (params % 2) {
        f = field_alloc[i];
    } else {

```

```

        make_double_entry ();
        cob_decimal_set_field (&d1, field_alloc[i-1]);
        cob_decimal_set_field (&d2, field_alloc[i]);
        cob_decimal_add (&d1, &d2);
        mpz_set_ui (d2.value, 2);
        d2.scale = 0;
        cob_decimal_div (&d1, &d2);
        cob_decimal_get_field (&d1, curr_field, 0);
        f = curr_field;
    }

    free (field_alloc);
    return f;
}

```

### 7.36.2.39 `cob_field* cob_intr_midrange ( const int params, ... )`

Definition at line 2146 of file `intrinsic.c`.

```

{
    cob_field      *f, *basemin, *basemax;
    va_list        args;
    int            i;

    make_double_entry ();
    va_start (args, params);

    basemin = va_arg (args, cob_field *);
    basemax = basemin;
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basemin) < 0) {
            basemin = f;
        }
        if (cob_cmp (f, basemax) > 0) {
            basemax = f;
        }
    }
    va_end (args);

    cob_decimal_set_field (&d1, basemin);
    cob_decimal_set_field (&d2, basemax);
    cob_decimal_add (&d1, &d2);
    mpz_set_ui (d2.value, 2);
    d2.scale = 0;
    cob_decimal_div (&d1, &d2);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

### 7.36.2.40 `cob_field* cob_intr_min ( const int params, ... )`

Definition at line 2104 of file `intrinsic.c`.

```

{

```

```

cob_field      *f, *basef;
va_list       args;
int           i;

va_start (args, params);

basef = va_arg (args, cob_field *);
for (i = 1; i < params; ++i) {
    f = va_arg (args, cob_field *);
    if (cob_cmp (f, basef) < 0) {
        basef = f;
    }
}
va_end (args);

return basef;
}

```

#### 7.36.2.41 `cob_field* cob_intr_mod ( cob_field * srcfield1, cob_field * srcfield2 )`

Definition at line 2268 of file intrinsic.c.

```

{
    cob_field      *f1;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    f1 = cob_intr_integer (cob_intr_binop (srcfield1, '/', srcfield2));
    cob_decimal_set_field (&d1, srcfield2);
    cob_decimal_set_field (&d2, f1);
    cob_decimal_mul (&d2, &d1);
    cob_decimal_set_field (&d1, srcfield1);
    cob_decimal_sub (&d1, &d2);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

#### 7.36.2.42 `cob_field* cob_intr_numval ( cob_field * srcfield )`

Definition at line 1795 of file intrinsic.c.

```

{
    long long      llval = 0;
    double         val;
    size_t         i;
    int            integer_digits = 0;
    int            decimal_digits = 0;
    int            sign = 0;
    int            decimal_seen = 0;
    cob_field_attr attr;
}

```

```

cob_field      field;
unsigned char  integer_buff[64];
unsigned char  decimal_buff[64];
unsigned char  final_buff[64];

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

COB_FIELD_INIT (8, NULL, &attr);
memset (integer_buff, 0, sizeof (integer_buff));
memset (decimal_buff, 0, sizeof (decimal_buff));
memset (final_buff, 0, sizeof (final_buff));

for (i = 0; i < srcfield->size; ++i) {
    if (i < (srcfield->size - 2)) {
        if (strcasecmp ((char *)&srcfield->data[i], "CR") == 0
            || strcasecmp ((char *)&srcfield->data[i], "DB") ==
0) {
                sign = 1;
                break;
            }
        }
    if (srcfield->data[i] == ' ') {
        continue;
    }
    if (srcfield->data[i] == '+') {
        continue;
    }
    if (srcfield->data[i] == '-') {
        sign = 1;
        continue;
    }
    if (srcfield->data[i] == cob_current_module->decimal_point) {
        decimal_seen = 1;
        continue;
    }
    if (srcfield->data[i] >= '0' && srcfield->data[i] <= '9') {
        llval *= 10;
        llval += srcfield->data[i] - '0';
        if (decimal_seen) {
            decimal_buff[decimal_digits++] = srcfield->data[i]
];
        } else {
            integer_buff[integer_digits++] = srcfield->data[i]
];
        }
    }
    if ((integer_digits + decimal_digits) > 30) {
        break;
    }
}
if (!integer_digits) {
    integer_buff[0] = '0';
}
if (!decimal_digits) {
    decimal_buff[0] = '0';
}
if (sign) {
    llval = -llval;
}
if ((integer_digits + decimal_digits) <= 18) {
    attr.scale = decimal_digits;
    make_field_entry (&field);
}

```

```

        memcpy (curr_field->data, (char *)&llval, 8);
    } else {
        snprintf ((char *)final_buff, 63, "%s%.%s", sign ? "-" : "",
                integer_buff, decimal_buff);
        sscanf ((char *)final_buff, "%lf", &val);
        make_double_entry ();
        memcpy (curr_field->data, (char *)&val, sizeof (double));
    }
    return curr_field;
}

```

### 7.36.2.43 cob\_field\* cob\_intr\_numval.c ( cob\_field \* srcfield, cob\_field \* currency )

Definition at line 1875 of file intrinsic.c.

```

{
    unsigned char    *currency_data;
    long long        llval = 0;
    double           val;
    size_t           i;
    int              integer_digits = 0;
    int              decimal_digits = 0;
    int              sign = 0;
    int              decimal_seen = 0;
    cob_field_attr   attr;
    cob_field        field;
    unsigned char    integer_buff[64];
    unsigned char    decimal_buff[64];
    unsigned char    final_buff[64];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    memset (integer_buff, 0, sizeof (integer_buff));
    memset (decimal_buff, 0, sizeof (decimal_buff));
    memset (final_buff, 0, sizeof (final_buff));

    currency_data = NULL;
    if (currency) {
        if (currency->size < srcfield->size) {
            currency_data = currency->data;
        }
    }
    for (i = 0; i < srcfield->size; ++i) {
        if (i < (srcfield->size - 2)) {
            if (strcasecmp ((char *)&srcfield->data[i], "CR") == 0
                || strcasecmp ((char *)&srcfield->data[i], "DB") ==
0) {
                sign = 1;
                break;
            }
        }
        if (currency_data) {
            if (i < (srcfield->size - currency->size)) {
                if (memcmp ((char *)&srcfield->data[i], currency_
data,
                            currency->size) == 0) {
                    i += (currency->size - 1);
                    continue;

```

```

        }
    }
    if (srcfield->data[i] == ' ') {
        continue;
    }
    if (srcfield->data[i] == '+') {
        continue;
    }
    if (srcfield->data[i] == '-') {
        sign = 1;
        continue;
    }
    if (srcfield->data[i] == cob_current_module->decimal_point) {
        decimal_seen = 1;
        continue;
    }
    if (srcfield->data[i] == cob_current_module->currency_symbol) {
        continue;
    }
    if (srcfield->data[i] >= '0' && srcfield->data[i] <= '9') {
        llval *= 10;
        llval += srcfield->data[i] - '0';
        if (decimal_seen) {
            decimal_buff[decimal_digits++] = srcfield->data[i];
];
        } else {
            integer_buff[integer_digits++] = srcfield->data[i];
];
        }
    }
    if ((integer_digits + decimal_digits) > 30) {
        break;
    }
}
if (!integer_digits) {
    integer_buff[0] = '0';
}
if (!decimal_digits) {
    decimal_buff[0] = '0';
}
if (sign) {
    llval = -llval;
}
if ((integer_digits + decimal_digits) <= 18) {
    attr.scale = decimal_digits;
    make_field_entry (&field);
    memcpy (curr_field->data, (char *)&llval, 8);
} else {
    snprintf ((char *)final_buff, 63, "%s%s.%s", sign ? "-" : "",
             integer_buff, decimal_buff);
    sscanf ((char *)final_buff, "%lf", &val);
    make_double_entry ();
    memcpy (curr_field->data, (char *)&val, sizeof (double));
}
return curr_field;
}

```

**7.36.2.44** `cob_field* cob_intr_ord ( cob_field * srcfield )`

Definition at line 1141 of file intrinsic.c.

```
{
    cob_field_attr  attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, (int)(*srcfield->data + 1));
    return curr_field;
}
```

**7.36.2.45** `cob_field* cob_intr_ord_max ( const int params, ... )`

Definition at line 2069 of file intrinsic.c.

```
{
    cob_field       *f, *basef;
    int             ordmin = 0;
    int             i;
    va_list         args;
    cob_field_attr  attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    if (params <= 1) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) > 0) {
            basef = f;
            ordmin = i;
        }
    }
    va_end (args);

    cob_set_int (curr_field, ordmin + 1);
    return curr_field;
}
```

**7.36.2.46** `cob_field* cob_intr_ord_min ( const int params, ... )`

Definition at line 2034 of file intrinsic.c.

```

{
    cob_field      *f, *basef;
    int            i;
    int            ordmin = 0;
    va_list        args;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    if (params <= 1) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) < 0) {
            basef = f;
            ordmin = i;
        }
    }
    va_end (args);

    cob_set_int (curr_field, ordmin + 1);
    return curr_field;
}

```

### 7.36.2.47 `cob_field*` `cob_intr_present_value` ( `const int params`, ... )

Definition at line 2538 of file `intrinsic.c`.

```

{
    cob_field      *f;
    va_list        args;
    int            i;

    va_start (args, params);
    make_double_entry ();

    if (params < 2) {
        va_end (args);
        fprintf (stderr, "Wrong number of parameters for FUNCTION PRESENT
-VALUE\n");
        fflush (stderr);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d1, f);
    mpz_set_ui (d2.value, 1);
    d2.scale = 0;
    cob_decimal_add (&d1, &d2);
}

```



```

mpz_set_ui (d4.value, 0);
d4.scale = 0;

for (i = 1; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    mpz_set (d3.value, d1.value);
    d3.scale = d1.scale;
    if (i > 1) {
        mpz_set_ui (d5.value, (unsigned int)i);
        d5.scale = 0;
        cob_decimal_pow (&d3, &d5);
    }
    cob_decimal_div (&d2, &d3);
    cob_decimal_add (&d4, &d2);
}
va_end (args);

cob_decimal_get_field (&d4, curr_field, 0);
return curr_field;
}

```

#### 7.36.2.48 cob\_field\* cob\_intr\_random ( const int *params*, ... )

Definition at line 2352 of file intrinsic.c.

```

{
    cob_field      *f;
    va_list        args;
    int            seed = 1;
    int            randnum;
    int            i;
    int            exp10;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 9, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    va_start (args, params);

    if (params) {
        f = va_arg (args, cob_field *);
        seed = cob_get_int (f);
        if (seed < 0) {
            seed = 0;
        }
#ifdef __CYGWIN__
        srandom ((unsigned int)seed);
#else
        srand ((unsigned int)seed);
#endif
    }
    va_end (args);

#ifdef __CYGWIN__
    randnum = (int)random ();
#else
    randnum = rand ();

```

```

#endif
    exp10 = 1;
    for (i = 0; i < 10; ++i) {
        if ((randnum / exp10) == 0) {
            break;
        }
        exp10 *= 10;
    }
    if (i == 0) {
        i = 1;
    }
    attr.scale = i;
    make_field_entry (&field);
    *(long long *)curr_field->data = (long long)randnum;
    return curr_field;
}

```

### 7.36.2.49 cob\_field\* cob\_intr\_range ( const int params, ... )

Definition at line 2289 of file intrinsic.c.

```

{
    cob_field      *f, *basemin, *basemax;
    va_list        args;
    int            i;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    va_start (args, params);

    basemin = va_arg (args, cob_field *);
    basemax = basemin;
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basemin) < 0) {
            basemin = f;
        }
        if (cob_cmp (f, basemax) > 0) {
            basemax = f;
        }
    }
    va_end (args);

    attr.scale = COB_FIELD_SCALE(basemin);
    if (COB_FIELD_SCALE(basemax) > attr.scale) {
        attr.scale = COB_FIELD_SCALE(basemax);
    }
    make_field_entry (&field);
    cob_decimal_set_field (&d1, basemax);
    cob_decimal_set_field (&d2, basemin);
    cob_decimal_sub (&d1, &d2);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

7.36.2.50 **cob\_field\* cob\_intr\_rem ( cob\_field \* srcfield1, cob\_field \* srcfield2 )**

Definition at line 2327 of file intrinsic.c.

```

{
    cob_field      *f1;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    f1 = cob_intr_integer_part (cob_intr_binop (srcfield1, '/', srcfield2));
    cob_decimal_set_field (&d1, srcfield2);
    cob_decimal_set_field (&d2, f1);
    cob_decimal_mul (&d2, &d1);
    cob_decimal_set_field (&d1, srcfield1);
    cob_decimal_sub (&d1, &d2);

    attr.scale = COB_FIELD_SCALE(srcfield1);
    if (COB_FIELD_SCALE(srcfield2) > attr.scale) {
        attr.scale = COB_FIELD_SCALE(srcfield2);
    }
    make_field_entry (&field);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

7.36.2.51 **cob\_field\* cob\_intr\_reverse ( const int offset, const int length, cob\_field \* srcfield )**

Definition at line 622 of file intrinsic.c.

```

{
    size_t      i, size;

    make_field_entry (srcfield);

    size = srcfield->size;
    for (i = 0; i < size; ++i) {
        curr_field->data[i] = srcfield->data[srcfield->size - i - 1];
    }
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

7.36.2.52 **cob\_field\* cob\_intr\_seconds\_from\_formatted\_time ( cob\_field \* format, cob\_field \* value )**

Definition at line 2803 of file intrinsic.c.

```

{

```

```

unsigned char *p1;
unsigned char *p2;
size_t n;
int seconds = 0;
int minutes = 0;
int hours = 0;
int seconds_seen = 0;
int minutes_seen = 0;
int hours_seen = 0;
cob_field_attr attr;
cob_field field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
COB_FIELD_INIT (4, NULL, &attr);
make_field_entry (&field);

cob_exception_code = 0;
if (value->size < format->size) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
p1 = format->data;
p2 = value->data;
for (n = 0; n < format->size - 1; ++n, ++p1, ++p2) {
    if (!memcmp (p1, "hh", 2) && !hours_seen) {
        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            hours = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');
            hours_seen = 1;
            continue;
        }
    }
    if (!memcmp (p1, "mm", 2) && !minutes_seen) {
        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            minutes = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');
            minutes_seen = 1;
            continue;
        }
    }
    if (!memcmp (p1, "ss", 2) && !seconds_seen) {
        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            seconds = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');
            seconds_seen = 1;
            continue;
        }
    }
}
if (hours_seen && minutes_seen && seconds_seen) {
    seconds += (hours * 3600) + (minutes * 60);
} else {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    seconds = 0;
}
cob_set_int (curr_field, seconds);
return curr_field;
}

```

**7.36.2.53 cob\_field\* cob\_intr\_seconds\_past\_midnight ( void )**

Definition at line 2782 of file intrinsic.c.

```

{
    struct tm      *timeptr;
    time_t        t;
    int           seconds;
    cob_field_attr attr;
    cob_field     field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    t = time (NULL);
    timeptr = localtime (&t);
    seconds = (timeptr->tm_hour * 3600) + (timeptr->tm_min * 60) +
              timeptr->tm_sec;
    cob_set_int (curr_field, seconds);
    return curr_field;
}

```

**7.36.2.54 cob\_field\* cob\_intr\_sign ( cob\_field \* srcfield )**

Definition at line 566 of file intrinsic.c.

```

{
    int           n;
    cob_field_attr attr;
    cob_field     field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, COB_FLAG_HAVE_SIGN, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, 0);
    n = cob_cmp (srcfield, curr_field);
    if (n < 0) {
        cob_set_int (curr_field, -1);
    } else if (n > 0) {
        cob_set_int (curr_field, 1);
    }

    return curr_field;
}

```

**7.36.2.55 cob\_field\* cob\_intr\_sin ( cob\_field \* srcfield )**

Definition at line 1726 of file intrinsic.c.

```

{
    long long      result;
    double         mathd2;

```

```

int                i, tempres;
cob_field_attr     attr;
cob_field          field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
COB_FIELD_INIT (8, NULL, &attr);
cob_decimal_set_field (&d1, srcfield);
make_field_entry (&field);

errno = 0;
mathd2 = sin (intr_get_double (&d1));
if (errno) {
    cob_set_int (curr_field, 0);
    return curr_field;
}
result = (long long) mathd2;
mathd2 -= result;
for (i = 0; i < 17; ++i) {
    mathd2 *= 10;
    tempres = (int) mathd2;
    result *= 10;
    result += tempres;
    mathd2 -= tempres;
}
memcpy (curr_field->data, (char *)&result, 8);
return curr_field;
}

```

### 7.36.2.56 `cob_field*` `cob_intr_sqrt` ( `cob_field` \* *srcfield* )

Definition at line 1759 of file `intrinsic.c`.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = sqrt (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

### 7.36.2.57 `cob_field*` `cob_intr_standard_deviation` ( `const int` *params*, ... )

Definition at line 2472 of file `intrinsic.c`.

```

{
    cob_field      *f;

```

```
va_list      args;
int          i;

va_start (args, params);
make_double_entry ();

if (params == 1) {
    va_end (args);
    cob_set_int (curr_field, 0);
    return curr_field;
}

/* MEAN for all params */
mpz_set_ui (d1.value, 0);
d1.scale = 0;

for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_add (&d1, &d2);
}
va_end (args);
mpz_set_ui (d2.value, (unsigned int)params);
d2.scale = 0;
cob_decimal_div (&d1, &d2);

/* Got the MEAN in d1, iterate again */

mpz_set_ui (d4.value, 0);
d4.scale = 0;

va_start (args, params);

for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_sub (&d2, &d1);
    cob_decimal_mul (&d2, &d2);
    cob_decimal_add (&d4, &d2);
}
va_end (args);

mpz_set_ui (d3.value, (unsigned int)params);
d3.scale = 0;
cob_decimal_div (&d4, &d3);
/* We have the VARIANCE in d4, sqrt = STANDARD-DEVIATION */

/* Do not know why this does not work
d5.scale = d4.scale;
mpz_mul_ui (d5.value, d4.value, 1000000000);
mpz_mul_ui (d4.value, d5.value, 1000000000);
mpz_sqrt (d5.value, d4.value);
mpz_div_ui (d4.value, d5.value, 1000000000);
cob_decimal_get_field (&d4, curr_field, 0);
return curr_field;
*/

cob_decimal_get_field (&d4, curr_field, 0);
f = cob_intr_sqrt (curr_field);
return f;
}
```

7.36.2.58 `cob_field* cob_intr_stored_char_length ( cob_field * srcfield )`

Definition at line 1155 of file intrinsic.c.

```

{
    unsigned char    *p;
    int              count;
    cob_field_attr  attr;
    cob_field        field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    count = srcfield->size;
    p = srcfield->data + srcfield->size - 1;
    for (; count > 0; count--, p--) {
        if (*p != ' ') {
            break;
        }
    }
    cob_set_int (curr_field, count);
    return curr_field;
}

```

7.36.2.59 `cob_field* cob_intr_substitute ( const int offset, const int length, const int params, ... )`

Definition at line 678 of file intrinsic.c.

```

{
    cob_field        *var;
    cob_field        **f1;
    cob_field        **f2;
    unsigned char    *p1;
    unsigned char    *p2;
    size_t           varsize;
    size_t           calcsize;
    size_t           n;
    size_t           found;
    int              numreps;
    int              i;
    cob_field_attr  attr;
    cob_field        field;
    va_list          args;

    numreps = params / 2;
    f1 = cob_malloc (numreps * sizeof (cob_field *));
    f2 = cob_malloc (numreps * sizeof (cob_field *));

    va_start (args, params);

    var = va_arg (args, cob_field *);
    varsize = var->size;

    /* Extract args */
    for (i = 0; i < params - 1; ++i) {
        if ((i % 2) == 0) {

```



```

        f1[i / 2] = va_arg (args, cob_field *);
    } else {
        f2[i / 2] = va_arg (args, cob_field *);
    }
}

/* Calculate required size */
calcsize = 0;
found = 0;
p1 = var->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!memcmp (p1, f1[i]->data, f1[i]->size)) {
                p1 += f1[i]->size;
                n += f1[i]->size;
                calcsize += f2[i]->size;
                found = 1;
                break;
            }
        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    ++p1;
    ++calcsize;
}

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
field.size = calcsize;
make_field_entry (&field);

found = 0;
p1 = var->data;
p2 = curr_field->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!memcmp (p1, f1[i]->data, f1[i]->size)) {
                memcpy (p2, f2[i]->data, f2[i]->size);
                p1 += f1[i]->size;
                p2 += f2[i]->size;
                n += f1[i]->size;
                found = 1;
                break;
            }
        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    *p2++ = *p1++;
}
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}

```

```

    free (f1);
    free (f2);
    return curr_field;
}

```

### 7.36.2.60 `cob_field*` `cob_intr_substitute_case` ( `const int offset`, `const int length`, `const int params`, ... )

Definition at line 775 of file `intrinsic.c`.

```

{
    cob_field      *var;
    cob_field      **f1;
    cob_field      **f2;
    unsigned char  *p1;
    unsigned char  *p2;
    size_t         varsize;
    size_t         calcsize;
    size_t         n;
    size_t         found;
    int            numreps;
    int            i;
    cob_field_attr attr;
    cob_field      field;
    va_list        args;

    numreps = params / 2;
    f1 = cob_malloc (numreps * sizeof (cob_field *));
    f2 = cob_malloc (numreps * sizeof (cob_field *));

    va_start (args, params);

    var = va_arg (args, cob_field *);
    varsize = var->size;

    /* Extract args */
    for (i = 0; i < params - 1; ++i) {
        if ((i % 2) == 0) {
            f1[i / 2] = va_arg (args, cob_field *);
        } else {
            f2[i / 2] = va_arg (args, cob_field *);
        }
    }

    /* Calculate required size */
    calcsize = 0;
    found = 0;
    p1 = var->data;
    for (n = 0; n < varsize; ) {
        for (i = 0; i < numreps; ++i) {
            if (n + f1[i]->size <= varsize) {
                if (!strncasecmp ((const char *)p1,
                                   (const char *) (f1[i]->data),
                                   f1[i]->size)) {
                    p1 += f1[i]->size;
                    n += f1[i]->size;
                    calcsize += f2[i]->size;
                    found = 1;
                    break;
                }
            }
        }
    }
}

```

```

        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    ++p1;
    ++calcsize;
}

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
field.size = calcsize;
make_field_entry (&field);

found = 0;
p1 = var->data;
p2 = curr_field->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!strncasecmp ((const char *)p1,
                              (const char *) (f1[i]->data),
                              f1[i]->size)) {
                memcpy (p2, f2[i]->data, f2[i]->size);
                p1 += f1[i]->size;
                p2 += f2[i]->size;
                n += f1[i]->size;
                found = 1;
                break;
            }
        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    *p2++ = *p1++;
}
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
free (f1);
free (f2);
return curr_field;
}

```

### 7.36.2.61 `cob_field* cob_intr_sum ( const int params, ... )`

Definition at line 1996 of file intrinsic.c.

```

{
    cob_field    *f;
    va_list      args;
    int          i;
    int          digits = 0;
}

```

```

int          scale = 0;
cob_field_attr attr;
cob_field    field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

COB_FIELD_INIT (8, NULL, &attr);
mpz_set_ui (d1.value, 0);
d1.scale = 0;

va_start (args, params);

for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    if ((COB_FIELD_DIGITS(f) - COB_FIELD_SCALE(f)) > digits) {
        digits = COB_FIELD_DIGITS(f) - COB_FIELD_SCALE(f);
    }
    if (COB_FIELD_SCALE(f) > scale) {
        scale = COB_FIELD_SCALE(f);
    }
    cob_decimal_set_field (&d2, f);
    cob_decimal_add (&d1, &d2);
}
va_end (args);

attr.scale = scale;
make_field_entry (&field);
cob_decimal_get_field (&d1, curr_field, 0);
return curr_field;
}

```

### 7.36.2.62 `cob_field* cob_intr_tan ( cob_field * srcfield )`

Definition at line 1777 of file `intrinsic.c`.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = tan (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

### 7.36.2.63 `cob_field* cob_intr_test_date_yyyymmdd ( cob_field * srcfield )`

Definition at line 1409 of file `intrinsic.c`.

```

{
    int            indate;
    int            days;
    int            month;
    int            year;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    /* Base 1601-01-01 */
    indate = cob_get_int (srcfield);
    year = indate / 10000;
    if (year < 1601 || year > 9999) {
        cob_set_int (curr_field, 1);
        return curr_field;
    }
    indate %= 10000;
    month = indate / 100;
    if (month < 1 || month > 12) {
        cob_set_int (curr_field, 2);
        return curr_field;
    }
    days = indate % 100;
    if (days < 1 || days > 31) {
        cob_set_int (curr_field, 3);
        return curr_field;
    }
    if (leap_year (year)) {
        if (days > leap_month_days[month]) {
            cob_set_int (curr_field, 3);
            return curr_field;
        }
    }
    else {
        if (days > normal_month_days[month]) {
            cob_set_int (curr_field, 3);
            return curr_field;
        }
    }
    cob_set_int (curr_field, 0);
    return curr_field;
}

```

### 7.36.2.64 cob\_field\*cob\_intr\_test\_day\_yyyyddd ( cob\_field \* srcfield )

Definition at line 1456 of file intrinsic.c.

```

{
    int            indate;
    int            days;
    int            year;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

```

```

/* Base 1601-01-01 */
indate = cob_get_int (srcfield);
year = indate / 1000;
if (year < 1601 || year > 9999) {
    cob_set_int (curr_field, 1);
    return curr_field;
}
days = indate % 1000;
if (days < 1 || days > 365 + leap_year (year)) {
    cob_set_int (curr_field, 2);
    return curr_field;
}
cob_set_int (curr_field, 0);
return curr_field;
}

```

### 7.36.2.65 `cob_field*cob_intr.trim ( const int offset, const int length, cob_field * srcfield, const int direction )`

Definition at line 876 of file intrinsic.c.

```

{
    unsigned char *begin;
    unsigned char *end;
    size_t i;
    size_t size = 0;

    make_field_entry (srcfield);

    for (i = 0; i < srcfield->size; ++i) {
        if (srcfield->data[i] != ' ') {
            break;
        }
    }
    if (i == srcfield->size) {
        curr_field->size = 1;
        curr_field->data[0] = ' ';
        return curr_field;
    }
    begin = srcfield->data;
    if (direction != 2) {
        for (; *begin == ' '; ++begin) ;
    }
    end = srcfield->data + srcfield->size - 1;
    if (direction != 1) {
        for (; *end == ' '; end--);
    }
    for (i = 0; begin <= end; ++begin, ++i) {
        curr_field->data[i] = *begin;
        ++size;
    }
    curr_field->size = size;
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

**7.36.2.66** `cob_field* cob_intr_upper_case ( const int offset, const int length, cob_field * srcfield )`

Definition at line 588 of file intrinsic.c.

```

{
    size_t          i, size;

    make_field_entry (srcfield);

    size = srcfield->size;
    for (i = 0; i < size; ++i) {
        curr_field->data[i] = toupper (srcfield->data[i]);
    }
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

**7.36.2.67** `cob_field* cob_intr_variance ( const int params, ... )`

Definition at line 2403 of file intrinsic.c.

```

{
    cob_field      *f;
    va_list        args;
    long long      n;
    union {
        unsigned char  data[8];
        long long      datall;
    } datun;
    int            i;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    if (params == 1) {
        make_field_entry (&field);
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    /* MEAN for all params */
    mpz_set_ui (d1.value, 0);
    d1.scale = 0;

    va_start (args, params);
    for (i = 0; i < params; ++i) {
        f = va_arg (args, cob_field *);
        cob_decimal_set_field (&d2, f);
        cob_decimal_add (&d1, &d2);
    }
    va_end (args);
    mpz_set_ui (d2.value, (unsigned int)params);
    d2.scale = 0;
}

```

```

cob_decimal_div (&d1, &d2);

/* Got the MEAN in d1, iterate again */

mpz_set_ui (d4.value, 0);
d4.scale = 0;

va_start (args, params);

for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_sub (&d2, &d1);
    cob_decimal_mul (&d2, &d2);
    cob_decimal_add (&d4, &d2);
}
va_end (args);

mpz_set_ui (d3.value, (unsigned int)params);
d3.scale = 0;
cob_decimal_div (&d4, &d3);
field.data = datun.data;
cob_decimal_get_field (&d4, &field, 0);
n = datun.datall;
for (i = 0; n; n /= 10, ++i) ;
field.data = NULL;
if (i <= 18) {
    attr.scale = 18 - i;
}
make_field_entry (&field);
cob_decimal_get_field (&d4, curr_field, 0);
return curr_field;
}

```

### 7.36.2.68 `cob_field*` `cob_intr_when_compiled` ( `const int offset`, `const int length`, `cob_field *f` )

Definition at line 1020 of file `intrinsic.c`.

```

{
    make_field_entry (f);

    memcpy (curr_field->data, f->data, f->size);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

### 7.36.2.69 `cob_field*` `cob_intr_year_to_yyyy` ( `const int params`, ... )

Definition at line 2583 of file `intrinsic.c`.

```

{
    cob_field      *f;

```



```

struct tm      *timeptr;
va_list       args;
time_t        t;
int           year;
int           interval;
int           xqtyear;
int           maxyear;
cob_field_attr attr;
cob_field     field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
COB_FIELD_INIT (4, NULL, &attr);
make_field_entry (&field);

cob_exception_code = 0;
va_start (args, params);
f = va_arg (args, cob_field *);
year = cob_get_int (f);
if (params > 1) {
    f = va_arg (args, cob_field *);
    interval = cob_get_int (f);
} else {
    interval = 50;
}
if (params > 2) {
    f = va_arg (args, cob_field *);
    xqtyear = cob_get_int (f);
} else {
    t = time (NULL);
    timeptr = localtime (&t);
    xqtyear = 1900 + timeptr->tm_year;
}
va_end (args);

if (year < 0 || year > 99) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (xqtyear < 1601 || xqtyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
maxyear = xqtyear + interval;
if (maxyear < 1700 || maxyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (maxyear % 100 >= year) {
    year += 100 * (maxyear / 100);
} else {
    year += 100 * ((maxyear / 100) - 1);
}
cob_set_int (curr_field, year);
return curr_field;
}

```

### 7.36.3 Variable Documentation

### 7.36.3.1 `cob_decimal d2`

Definition at line 68 of file `intrinsic.c`.

### 7.36.3.2 `cob_decimal d3`

Definition at line 68 of file `intrinsic.c`.

### 7.36.3.3 `cob_decimal d4`

Definition at line 68 of file `intrinsic.c`.

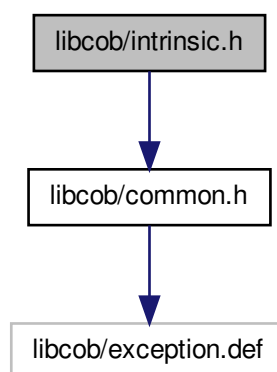
### 7.36.3.4 `cob_decimal d5`

Definition at line 68 of file `intrinsic.c`.

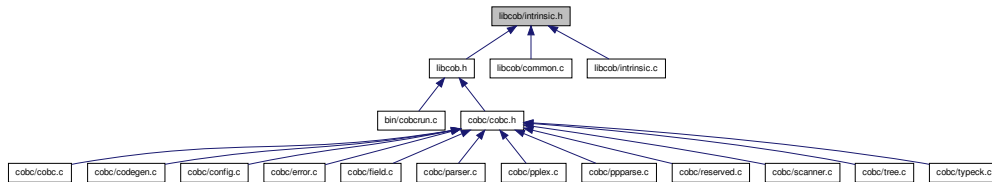
## 7.37 `libcob/intrinsic.h` File Reference

```
#include <libcob/common.h>
```

Include dependency graph for `intrinsic.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- `cob_field * cob_intr_binop (cob_field *, int, cob_field *)`
- `cob_field * cob_intr_current_date (const int, const int)`
- `cob_field * cob_intr_when_compiled (const int, const int, cob_field *)`
- `cob_field * cob_intr_exception_file (void)`
- `cob_field * cob_intr_exception_location (void)`
- `cob_field * cob_intr_exception_status (void)`
- `cob_field * cob_intr_exception_statement (void)`
- `cob_field * cob_intr_char (cob_field *)`
- `cob_field * cob_intr_ord (cob_field *)`
- `cob_field * cob_intr_stored_char_length (cob_field *)`
- `cob_field * cob_intr_combined_datetime (cob_field *, cob_field *)`
- `cob_field * cob_intr_date_of_integer (cob_field *)`
- `cob_field * cob_intr_day_of_integer (cob_field *)`
- `cob_field * cob_intr_integer_of_date (cob_field *)`
- `cob_field * cob_intr_integer_of_day (cob_field *)`
- `cob_field * cob_intr_test_date_yyyymmdd (cob_field *)`
- `cob_field * cob_intr_test_day_yyyyddd (cob_field *)`
- `cob_field * cob_intr_factorial (cob_field *)`
- `cob_field * cob_intr_exp (cob_field *)`
- `cob_field * cob_intr_exp10 (cob_field *)`
- `cob_field * cob_intr_abs (cob_field *)`
- `cob_field * cob_intr_acos (cob_field *)`
- `cob_field * cob_intr_asin (cob_field *)`
- `cob_field * cob_intr_atan (cob_field *)`
- `cob_field * cob_intr_cos (cob_field *)`
- `cob_field * cob_intr_log (cob_field *)`
- `cob_field * cob_intr_log10 (cob_field *)`
- `cob_field * cob_intr_sin (cob_field *)`
- `cob_field * cob_intr_sqrt (cob_field *)`
- `cob_field * cob_intr_tan (cob_field *)`
- `cob_field * cob_intr_upper_case (const int, const int, cob_field *)`
- `cob_field * cob_intr_lower_case (const int, const int, cob_field *)`
- `cob_field * cob_intr_reverse (const int, const int, cob_field *)`

- `cob_field * cob_intr_concatenate` (const int, const int, const int,...)
- `cob_field * cob_intr_substitute` (const int, const int, const int,...)
- `cob_field * cob_intr_substitute_case` (const int, const int, const int,...)
- `cob_field * cob_intr_trim` (const int, const int, `cob_field *`, const int)
- `cob_field * cob_intr_length` (`cob_field *`)
- `cob_field * cob_intr_integer` (`cob_field *`)
- `cob_field * cob_intr_integer_part` (`cob_field *`)
- `cob_field * cob_intr_fraction_part` (`cob_field *`)
- `cob_field * cob_intr_sign` (`cob_field *`)
- `cob_field * cob_intr_numval` (`cob_field *`)
- `cob_field * cob_intr_numval_c` (`cob_field *`, `cob_field *`)
- `cob_field * cob_intr_annuity` (`cob_field *`, `cob_field *`)
- `cob_field * cob_intr_mod` (`cob_field *`, `cob_field *`)
- `cob_field * cob_intr_rem` (`cob_field *`, `cob_field *`)
- `cob_field * cob_intr_sum` (const int,...)
- `cob_field * cob_intr_ord_min` (const int,...)
- `cob_field * cob_intr_ord_max` (const int,...)
- `cob_field * cob_intr_min` (const int,...)
- `cob_field * cob_intr_max` (const int,...)
- `cob_field * cob_intr_midrange` (const int,...)
- `cob_field * cob_intr_median` (const int,...)
- `cob_field * cob_intr_mean` (const int,...)
- `cob_field * cob_intr_range` (const int,...)
- `cob_field * cob_intr_random` (const int,...)
- `cob_field * cob_intr_variance` (const int,...)
- `cob_field * cob_intr_standard_deviation` (const int,...)
- `cob_field * cob_intr_present_value` (const int,...)
- `cob_field * cob_intr_year_to_yyyy` (const int,...)
- `cob_field * cob_intr_date_to_yyyymmdd` (const int,...)
- `cob_field * cob_intr_day_to_yyyddd` (const int,...)
- `cob_field * cob_intr_locale_date` (const int, const int, `cob_field *`, `cob_field *`)
- `cob_field * cob_intr_locale_time` (const int, const int, `cob_field *`, `cob_field *`)
- `cob_field * cob_intr_seconds_past_midnight` (void)
- `cob_field * cob_intr_lcl_time_from_secs` (const int, const int, `cob_field *`, `cob_field *`)
- `cob_field * cob_intr_seconds_from_formatted_time` (`cob_field *`, `cob_field *`)

### 7.37.1 Function Documentation

#### 7.37.1.1 `cob_field* cob_intr_abs ( cob_field * )`

Definition at line 1545 of file `intrinsic.c`.

```

{
    make_field_entry (srcfield);

    cob_decimal_set_field (&d1, srcfield);
    mpz_abs (d1.value, d1.value);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

### 7.37.1.2 cob\_field\* cob\_intr\_acos ( cob\_field \* )

Definition at line 1557 of file intrinsic.c.

```

{
    unsigned long long    result;
    double                mathd2;
    int                   i, tempres;
    cob_field_attr        attr;
    cob_field              field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = acos (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    result = (unsigned long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

### 7.37.1.3 cob\_field\* cob\_intr\_annuity ( cob\_field \*, cob\_field \* )

Definition at line 1974 of file intrinsic.c.

```

{
    double                mathd1, mathd2;

    make_double_entry ();
}

```

```

cob_decimal_set_field (&d1, srcfield1);
cob_decimal_set_field (&d2, srcfield2);

mathd1 = intr_get_double (&d1);
mathd2 = intr_get_double (&d2);
if (mathd1 == 0) {
    mathd1 = 1.0 / mathd2;
    memcpy (curr_field->data, (char *)&mathd1, sizeof (double));
    return curr_field;
}
mathd1 /= (1.0 - pow (mathd1 + 1.0, 0.0 - mathd2));
memcpy (curr_field->data, (char *)&mathd1, sizeof (double));
return curr_field;
}

```

#### 7.37.1.4 `cob_field* cob_intr.asin ( cob_field * )`

Definition at line 1591 of file intrinsic.c.

```

{
    long long          result;
    double             mathd2;
    int                i, tempres;
    cob_field_attr     attr;
    cob_field          field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = asin (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

#### 7.37.1.5 `cob_field* cob_intr.atan ( cob_field * )`

Definition at line 1624 of file intrinsic.c.

```

{

```

```

long long          result;
double            mathd2;
int               i, tempres;
cob_field_attr    attr;
cob_field         field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
COB_FIELD_INIT (8, NULL, &attr);
cob_decimal_set_field (&d1, srcfield);
make_field_entry (&field);

errno = 0;
mathd2 = atan (intr_get_double (&d1));
if (errno) {
    cob_set_int (curr_field, 0);
    return curr_field;
}
result = (long long) mathd2;
mathd2 -= result;
for (i = 0; i < 17; ++i) {
    mathd2 *= 10;
    tempres = (int) mathd2;
    result *= 10;
    result += tempres;
    mathd2 -= tempres;
}
memcpy (curr_field->data, (char *)&result, 8);
return curr_field;
}

```

#### 7.37.1.6 cob\_field\* cob\_intr\_binop( cob\_field \*, int, cob\_field \* )

Definition at line 432 of file intrinsic.c.

```

{
/* RXW
size_t          bitnum;
size_t          sign;
size_t          attrsign;
cob_field_attr  attr;
cob_field       field;
*/

cob_decimal_set_field (&d1, f1);
cob_decimal_set_field (&d2, f2);
switch (op) {
case '+':
    cob_decimal_add (&d1, &d2);
    break;
case '-':
    cob_decimal_sub (&d1, &d2);
    break;
case '*':
    cob_decimal_mul (&d1, &d2);
    break;
case '/':
    cob_decimal_div (&d1, &d2);
    break;
}

```

```

    case '^':
        cob_decimal_pow (&d1, &d2);
        break;
    default:
        break;
}

/* RXW
if (mpz_sgn (dl.value) < 0) {
    attrsign = COB_FLAG_HAVE_SIGN;
    sign = 1;
} else {
    attrsign = 0;
    sign = 0;
}
bitnum = mpz_sizeinbase (dl.value, 2);
if (bitnum < 33 - sign) {
    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, attrsign, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    attr.scale = dl.scale;
    make_field_entry (&field);
} else if (bitnum < 65 - sign) {
    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, attrsign, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    attr.scale = dl.scale;
    make_field_entry (&field);
} else {
*/
    make_double_entry ();
/* RXW
}
*/
cob_decimal_get_field (&d1, curr_field, 0);

return curr_field;
}

```

### 7.37.1.7 cob\_field\*cob\_intr\_char( cob\_field\* )

Definition at line 1121 of file intrinsic.c.

```

{
    int          i;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (1, NULL, &attr);
    make_field_entry (&field);

    i = cob_get_int (srcfield);
    if (i < 1 || i > 256) {
        *curr_field->data = 0;
    } else {
        *curr_field->data = i - 1;
    }
    return curr_field;
}

```



**7.37.1.8 cob\_field\* cob\_intr\_combined\_datetime ( cob\_field \*, cob\_field \* )**

Definition at line 1178 of file intrinsic.c.

```

{
    int          srdays;
    int          srtime;
    cob_field_attr attr;
    cob_field    field;
    char         buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 12, 5, 0, NULL);
    COB_FIELD_INIT (12, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    srdays = cob_get_int (srcdays);
    if (srdays < 1 || srdays > 3067671) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 12);
        return curr_field;
    }
    srtime = cob_get_int (srctime);
    if (srtime < 1 || srtime > 86400) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 12);
        return curr_field;
    }
    snprintf (buff, 15, "%7.7d%5.5d", srdays, srtime);
    memcpy (curr_field->data, buff, 12);
    return curr_field;
}

```

**7.37.1.9 cob\_field\* cob\_intr\_concatenate ( const int , const int , const int , ... )**

Definition at line 639 of file intrinsic.c.

```

{
    cob_field    **f;
    unsigned char *p;
    size_t      calcsize;
    int         i;
    cob_field_attr attr;
    cob_field    field;
    va_list     args;

    f = cob_malloc (params * sizeof (cob_field *));

    va_start (args, params);

    /* Extract args / calculate size */
    calcsize = 0;
    for (i = 0; i < params; ++i) {
        f[i] = va_arg (args, cob_field *);
        calcsize += f[i]->size;
    }

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);

```

```

COB_FIELD_INIT (calcsize, NULL, &attr);
make_field_entry (&field);

p = curr_field->data;
for (i = 0; i < params; ++i) {
    memcpy (p, f[i]->data, f[i]->size);
    p += f[i]->size;
}

if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
free (f);
return curr_field;
}

```

### 7.37.1.10 `cob_field* cob_intr_cos ( cob_field * )`

Definition at line 1657 of file intrinsic.c.

```

{
    long long          result;
    double             mathd2;
    int                i, tempres;
    cob_field_attr     attr;
    cob_field          field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = cos (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

### 7.37.1.11 `cob_field* cob_intr_current_date ( const int , const int )`

Definition at line 1032 of file intrinsic.c.



```

    if (tmptr->tm_isdst > 0) {
        timezone -= 3600;
    }
    if (timezone <= 0) {
        contz = -timezone;
        buff[16] = '+';
    } else {
        contz = timezone;
        buff[16] = '-';
    }
    sprintf(&buff[17], "%2.2ld%2.2ld", contz / 3600, (contz % 3600) / 60);
#else
    strftime (buff, 22, "%Y%m%d%H%M%S000000", localtime (&curtime));
#endif

#if defined(HAVE_SYS_TIME_H) && defined(HAVE_GETTIMEOFDAY)
    snprintf(buff2, 7, "%2.2ld", tmv.tv_usec / 10000);
    memcpy (&buff[14], buff2, 2);
#endif
#endif /* _WIN32 */

    memcpy (curr_field->data, buff, 21);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

### 7.37.1.12 cob\_field\* cob\_intr\_date\_of\_integer ( cob\_field \* )

Definition at line 1209 of file intrinsic.c.

```

{
    int          i;
    int          days;
    int          baseyear = 1601;
    int          leapyear = 365;
    cob_field_attr attr;
    cob_field    field;
    char         buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 8, 0, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    /* Base 1601-01-01 */
    days = cob_get_int (srcdays);
    if (days < 1 || days > 3067671) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 8);
        return curr_field;
    }
    while (days > leapyear) {
        days -= leapyear;
        ++baseyear;
        if (leap_year (baseyear)) {
            leapyear = 366;
        } else {

```

```

        leapyear = 365;
    }
}
for (i = 0; i < 13; ++i) {
    if (leap_year (baseyear)) {
        if (days <= leap_days[i]) {
            days -= leap_days[i-1];
            break;
        }
    } else {
        if (days <= normal_days[i]) {
            days -= normal_days[i-1];
            break;
        }
    }
}
snprintf (buff, 15, "%4.4d%2.2d%2.2d", baseyear, i, days);
memcpy (curr_field->data, buff, 8);
return curr_field;
}

```

#### 7.37.1.13 cob\_field\* cob\_intr\_date\_to\_yyyymmdd ( const int, ... )

Definition at line 2646 of file intrinsic.c.

```

{
    cob_field      *f;
    struct tm      *timeptr;
    va_list        args;
    time_t         t;
    int            year;
    int            mmdd;
    int            interval;
    int            xqtyear;
    int            maxyear;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    va_start (args, params);
    f = va_arg (args, cob_field *);
    year = cob_get_int (f);
    mmdd = year % 10000;
    year /= 10000;
    if (params > 1) {
        f = va_arg (args, cob_field *);
        interval = cob_get_int (f);
    } else {
        interval = 50;
    }
    if (params > 2) {
        f = va_arg (args, cob_field *);
        xqtyear = cob_get_int (f);
    } else {
        t = time (NULL);
    }
}

```

```

        timeptr = localtime (&t);
        xqtyear = 1900 + timeptr->tm_year;
    }
    va_end (args);

    if (year < 0 || year > 999999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    if (xqtyear < 1601 || xqtyear > 9999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    maxyear = xqtyear + interval;
    if (maxyear < 1700 || maxyear > 9999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    if (maxyear % 100 >= year) {
        year += 100 * (maxyear / 100);
    } else {
        year += 100 * ((maxyear / 100) - 1);
    }
    year *= 10000;
    year += mmdd;
    cob_set_int (curr_field, year);
    return curr_field;
}

```

#### 7.37.1.14 cob\_field\* cob\_intr\_day\_of\_integer ( cob\_field \* )

Definition at line 1259 of file intrinsic.c.

```

{
    int            days;
    int            baseyear = 1601;
    int            leapyear = 365;
    cob_field_attr attr;
    cob_field      field;
    char           buff[16];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, 7, 0, 0, NULL);
    COB_FIELD_INIT (7, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    /* Base 1601-01-01 */
    days = cob_get_int (srcdays);
    if (days < 1 || days > 3067671) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        memset (curr_field->data, '0', 7);
        return curr_field;
    }
    while (days > leapyear) {
        days -= leapyear;
        ++baseyear;
    }
}

```

```

        if (leap_year (baseyear)) {
            leapyear = 366;
        } else {
            leapyear = 365;
        }
    }
    snprintf (buff, 15, "%4.4d%3.3d", baseyear, days);
    memcpy (curr_field->data, buff, 7);
    return curr_field;
}

```

#### 7.37.1.15 cob\_field\* cob\_intr\_day\_to\_yyyyddd ( const int, ... )

Definition at line 2714 of file intrinsic.c.

```

{
    cob_field      *f;
    struct tm      *timeptr;
    va_list        args;
    time_t         t;
    int            year;
    int            days;
    int            interval;
    int            xqtyear;
    int            maxyear;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    va_start (args, params);
    f = va_arg (args, cob_field *);
    year = cob_get_int (f);
    days = year % 1000;
    year /= 1000;
    if (params > 1) {
        f = va_arg (args, cob_field *);
        interval = cob_get_int (f);
    } else {
        interval = 50;
    }
    if (params > 2) {
        f = va_arg (args, cob_field *);
        xqtyear = cob_get_int (f);
    } else {
        t = time (NULL);
        timeptr = localtime (&t);
        xqtyear = 1900 + timeptr->tm_year;
    }
    va_end (args);

    if (year < 0 || year > 999999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
}

```

```

if (xqtyear < 1601 || xqtyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
maxyear = xqtyear + interval;
if (maxyear < 1700 || maxyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (maxyear % 100 >= year) {
    year += 100 * (maxyear / 100);
} else {
    year += 100 * ((maxyear / 100) - 1);
}
year *= 1000;
year += days;
cob_set_int (curr_field, year);
return curr_field;
}

```

#### 7.37.1.16 cob\_field\* cob\_intr\_exception.file ( void )

Definition at line 916 of file intrinsic.c.

```

{
    size_t      flen;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    if (cob_exception_code == 0 || !cob_error_file ||
        (cob_exception_code & 0x0500) != 0x0500) {
        field.size = 2;
        make_field_entry (&field);
        memcpy (curr_field->data, "00", 2);
    } else {
        flen = strlen (cob_error_file->select_name);
        field.size = flen + 2;
        make_field_entry (&field);
        memcpy (curr_field->data, cob_error_file->file_status, 2);
        memcpy (&(curr_field->data[2]), cob_error_file->select_name, flen
    );
    }
    return curr_field;
}

```

#### 7.37.1.17 cob\_field\* cob\_intr\_exception.location ( void )

Definition at line 940 of file intrinsic.c.

```

{
    cob_field_attr attr;

```



```

cob_field      field;

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
if (!cob_got_exception || !cob_orig_program_id) {
    field.size = 1;
    make_field_entry (&field);
    *(curr_field->data) = ' ';
    return curr_field;
}
memset (locale_buff, 0, COB_SMALL_BUFF);
if (cob_orig_section && cob_orig_paragraph) {
    snprintf (locale_buff, COB_SMALL_MAX, "%s; %s OF %s; %d",
             cob_orig_program_id, cob_orig_paragraph,
             cob_orig_section, cob_orig_line);
} else if (cob_orig_section) {
    snprintf (locale_buff, COB_SMALL_MAX, "%s; %s; %d",
             cob_orig_program_id, cob_orig_section, cob_orig_line);
} else if (cob_orig_paragraph) {
    snprintf (locale_buff, COB_SMALL_MAX, "%s; %s; %d",
             cob_orig_program_id, cob_orig_paragraph, cob_orig_line)
;
} else {
    snprintf (locale_buff, COB_SMALL_MAX, "%s; ; %d",
             cob_orig_program_id, cob_orig_line);
}
field.size = strlen (locale_buff);
make_field_entry (&field);
memcpy (curr_field->data, locale_buff, field.size);
return curr_field;
}

```

#### 7.37.1.18 cob\_field\* cob\_intr\_exception\_statement ( void )

Definition at line 997 of file intrinsic.c.

```

{
    size_t      flen;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (31, NULL, &attr);
    make_field_entry (&field);

    memset (curr_field->data, ' ', 31);
    if (cob_exception_code && cob_orig_statement) {
        flen = strlen (cob_orig_statement);
        if (flen > 31) {
            memcpy (curr_field->data, cob_orig_statement, 31);
        } else {
            memcpy (curr_field->data, cob_orig_statement, flen);
        }
    }
    return curr_field;
}

```

**7.37.1.19 cob\_field\* cob\_intr\_exception\_status ( void )**

Definition at line 975 of file intrinsic.c.

```

{
    const char      *except_name;
    cob_field_attr  attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (31, NULL, &attr);
    make_field_entry (&field);

    memset (curr_field->data, ' ', 31);
    if (cob_exception_code) {
        except_name = cob_get_exception_name (cob_exception_code);
        if (except_name == NULL) {
            except_name = "EXCEPTION-OBJECT";
        }
        memcpy (curr_field->data, except_name, strlen (except_name));
    }
    return curr_field;
}

```

**7.37.1.20 cob\_field\* cob\_intr\_exp ( cob\_field \* )**

Definition at line 1509 of file intrinsic.c.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = pow (2.7182818284590452354, intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

**7.37.1.21 cob\_field\* cob\_intr\_exp10 ( cob\_field \* )**

Definition at line 1527 of file intrinsic.c.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

```

```
    errno = 0;
    mathd2 = pow (10.0, intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}
```

#### 7.37.1.22 cob\_field\* cob\_intr\_factorial ( cob\_field \* )

Definition at line 1485 of file intrinsic.c.

```
{
    int          srcval;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, 0, NULL);
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    srcval = cob_get_int (srcfield);
    if (srcval < 0) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    dl.scale = 0;
    mpz_fac_ui (dl.value, (unsigned int)srcval);
    cob_decimal_get_field (&dl, curr_field, 0);
    return curr_field;
}
```

#### 7.37.1.23 cob\_field\* cob\_intr\_fraction\_part ( cob\_field \* )

Definition at line 552 of file intrinsic.c.

```
{
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 18, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_move (srcfield, curr_field);
    return curr_field;
}
```

7.37.1.24 `cob_field* cob_intr_integer ( cob_field * )`

Definition at line 511 of file intrinsic.c.

```

{
    int            i, scale;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_decimal_set_field (&d1, srcfield);
    if (mpz_sgn (d1.value) >= 0) {
        cob_decimal_get_field (&d1, curr_field, 0);
        return curr_field;
    }
    scale = 1;
    for (i = 0; i < d1.scale; ++i) {
        scale *= 10;
    }
    if (mpz_fdiv_ui (d1.value, (unsigned int)scale)) {
        mpz_sub_ui (d1.value, d1.value, (unsigned int)scale);
    }
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

7.37.1.25 `cob_field* cob_intr_integer_of_date ( cob_field * )`

Definition at line 1295 of file intrinsic.c.

```

{
    int            indate;
    int            days;
    int            totaldays;
    int            month;
    int            year;
    int            baseyear = 1601;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    /* Base 1601-01-01 */
    indate = cob_get_int (srcfield);
    year = indate / 10000;
    if (year < 1601 || year > 9999) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    indate %= 10000;

```

```

month = indate / 100;
if (month < 1 || month > 12) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
days = indate % 100;
if (days < 1 || days > 31) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (leap_year (year)) {
    if (days > leap_month_days[month]) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
} else {
    if (days > normal_month_days[month]) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
}
totaldays = 0;
while (baseyear != year) {
    if (leap_year (baseyear)) {
        totaldays += 366;
    } else {
        totaldays += 365;
    }
    ++baseyear;
}
if (leap_year (baseyear)) {
    totaldays += leap_days[month - 1];
} else {
    totaldays += normal_days[month - 1];
}
totaldays += days;
cob_set_int (curr_field, totaldays);
return curr_field;
}

```

#### 7.37.1.26 cob\_field\* cob\_intr\_integer\_of\_day ( cob\_field \* )

Definition at line 1365 of file intrinsic.c.

```

{
    int          indate;
    int          days;
    int          totaldays;
    int          year;
    int          baseyear = 1601;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
}

```

```

make_field_entry (&field);

cob_exception_code = 0;
/* Base 1601-01-01 */
indate = cob_get_int (srcfield);
year = indate / 1000;
if (year < 1601 || year > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
days = indate % 1000;
if (days < 1 || days > 365 + leap_year (year)) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
totaldays = 0;
while (baseyear != year) {
    if (leap_year (baseyear)) {
        totaldays += 366;
    } else {
        totaldays += 365;
    }
    ++baseyear;
}
totaldays += days;
cob_set_int (curr_field, totaldays);
return curr_field;
}

```

#### 7.37.1.27 cob\_field\* cob\_intr\_integer\_part ( cob\_field \* )

Definition at line 538 of file intrinsic.c.

```

{
    cob_field_attr attr;
    cob_field field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    cob_move (srcfield, curr_field);
    return curr_field;
}

```

#### 7.37.1.28 cob\_field\* cob\_intr\_lcl\_time\_from\_secs ( const int , const int , cob\_field \* , cob\_field \* )

Definition at line 3149 of file intrinsic.c.

```

{
    cob_field_attr attr;

```

```

        cob_field      field;
#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
        size_t         len;
        int            indate;
        int            hours;
        int            minutes;
        int            seconds;
#endif
#ifdef HAVE_LANGINFO_CODESET
        char            *deflocale = NULL;
        char            *localep = NULL;
        char            *localep2;
        struct tm       tstruct;
        char            buff2[128];
#else
        char            *p;
        LCID            localeid = LOCALE_USER_DEFAULT;
        SYSTEMTIME      syst;
#endif
        char            buff[128];
#endif

        COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
        COB_FIELD_INIT (0, NULL, &attr);
        cob_exception_code = 0;

#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
        if (COB_FIELD_IS_NUMERIC (srcfield)) {
                indate = cob_get_int (srcfield);
        } else {
                goto derror;
        }
        if (indate > 86400) {
                goto derror;
        }
        hours = indate / 3600;
        indate %= 3600;
        minutes = indate / 60;
        seconds = indate % 60;

#ifdef HAVE_LANGINFO_CODESET
        memset ((void *)&tstruct, 0, sizeof(struct tm));
        tstruct.tm_hour = hours;
        tstruct.tm_min = minutes;
        tstruct.tm_sec = seconds;
        if (locale_field) {
                if (locale_field->size >= COB_SMALL_BUFFER) {
                        goto derror;
                }
                cob_field_to_string (locale_field, locale_buff);
                deflocale = locale_buff;
                localep2 = setlocale (LC_TIME, NULL);
                if (localep2) {
                        localep = strdup (localep2);
                }
                (void) setlocale (LC_TIME, deflocale);
        }
        memset (buff2, 0, sizeof(buff2));
        sprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(T_FMT));
        if (deflocale) {
                if (localep) {
                        (void) setlocale (LC_TIME, localep);
                }
        }
#endif
#endif

```

```

    }
    strftime (buff, sizeof(buff), buff2, &tstruct);
#else
    memset ((void *)&syst, 0, sizeof(syst));
    syst.wHour = hours;
    syst.wMinute = minutes;
    syst.wSecond = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFF) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        for (p = locale_buff; *p; ++p) {
            if (isalnum(*p) || *p == '_') {
                continue;
            }
            break;
        }
        *p = 0;
        for (len = 0; len < WINLOCSIZE; ++len) {
            if (!strcmp(locale_buff, wintable[len].winlocalename)) {
                localeid = wintable[len].winlocaleid;
                break;
            }
        }
        if (len == WINLOCSIZE) {
            goto derror;
        }
    }
    if (!GetTimeFormat (localeid, LOCALE_NOUSEROVERRIDE, &syst, NULL, buff, s
sizeof(buff))) {
        goto derror;
    }
#endif
    len = strlen (buff);
    field.size = len;
    make_field_entry (&field);
    memcpy (curr_field->data, buff, len);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
derror:
#endif
    field.size = 10;
    make_field_entry (&field);
    memset (curr_field->data, ' ', 10);
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    return curr_field;
}

```

### 7.37.1.29 cob\_field\* cob\_intr\_length( cob\_field \* )

Definition at line 497 of file intrinsic.c.

```

{
    cob_field_attr attr;
    cob_field field;

```



```

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, (int)srcfield->size);
    return curr_field;
}

```

### 7.37.1.30 `cob_field* cob_intr_locale_date ( const int, const int, cob_field *, cob_field * )`

Definition at line 2866 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;
#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    size_t        len;
    int            indate;
    int            days;
    int            month;
    int            year;
#elseif defined HAVE_LANGINFO_CODESET
    unsigned char  *p;
    char           *deflocale = NULL;
    char           *localep = NULL;
    char           *localep2;
    struct tm      tstruct;
    char           buff2[128];
#else
    char           *p;
    LCID           localeid = LOCALE_USER_DEFAULT;
    SYSTEMTIME     syst;
#endif
    char           buff[128];
#endif

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    cob_exception_code = 0;

#if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    if (COB_FIELD_IS_NUMERIC (srcfield)) {
        indate = cob_get_int (srcfield);
    } else {
        if (srcfield->size < 8) {
            goto derror;
        }
        p = srcfield->data;
        indate = 0;
        for (len = 0; len < 8; ++len, ++p) {
            if (isdigit (*p)) {
                indate *= 10;
                indate += (*p - '0');
            } else {
                goto derror;
            }
        }
    }
}

```

```

    }
    year = indate / 10000;
    if (year < 1601 || year > 9999) {
        goto derror;
    }
    indate %= 10000;
    month = indate / 100;
    if (month < 1 || month > 12) {
        goto derror;
    }
    days = indate % 100;
    if (days < 1 || days > 31) {
        goto derror;
    }
    if (leap_year (year)) {
        if (days > leap_month_days[month]) {
            goto derror;
        }
    } else {
        if (days > normal_month_days[month]) {
            goto derror;
        }
    }
}
#ifdef HAVE_LANGINFO_CODESET
month--;

memset ((void *)&tstruct, 0, sizeof(struct tm));
tstruct.tm_year = year - 1900;
tstruct.tm_mon = month;
tstruct.tm_mday = days;
if (locale_field) {
    if (locale_field->size >= COB_SMALL_BUFF) {
        goto derror;
    }
    cob_field_to_string (locale_field, locale_buff);
    deflocale = locale_buff;
    localep2 = setlocale (LC_TIME, NULL);
    if (localep2) {
        localep = strdup (localep2);
    }
    (void) setlocale (LC_TIME, deflocale);
}
memset (buff2, 0, sizeof(buff2));
snprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(D_FMT));
if (deflocale) {
    if (localep) {
        (void) setlocale (LC_TIME, localep);
    }
}
strftime (buff, sizeof(buff), buff2, &tstruct);
#else
memset ((void *)&syst, 0, sizeof(syst));
syst.wYear = year;
syst.wMonth = month;
syst.wDay = days;
if (locale_field) {
    if (locale_field->size >= COB_SMALL_BUFF) {
        goto derror;
    }
}
cob_field_to_string (locale_field, locale_buff);
for (p = locale_buff; *p; ++p) {
    if (isalnum(*p) || *p == '_' ) {

```

```

        continue;
    }
    break;
}
*p = 0;
for (len = 0; len < WINLOCSIZE; ++len) {
    if (!strcmp(locale_buff, wintable[len].winlocalename)) {
        localeid = wintable[len].winlocaleid;
        break;
    }
}
if (len == WINLOCSIZE) {
    goto derror;
}
}
if (!GetDateFormat (localeid, DATE_SHORTDATE, &syst, NULL, buff, sizeof(b
uff))) {
    goto derror;
}
}
#endif
len = strlen (buff);
field.size = len;
make_field_entry (&field);
memcpy (curr_field->data, buff, len);
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
return curr_field;
derror:
#endif
field.size = 10;
make_field_entry (&field);
memset (curr_field->data, ' ', 10);
cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
return curr_field;
}

```

### 7.37.1.31 `cob_field* cob_intr_locale_time ( const int, const int, cob_field *, cob_field * )`

Definition at line 3012 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;
#ifdef _WIN32 || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
    size_t        len;
    int           indate;
    int           hours;
    int           minutes;
    int           seconds;
#endif
#ifdef HAVE_LANGINFO_CODESET
    unsigned char *p;
    char          *deflocale = NULL;
    char          *localep = NULL;
    char          *localep2;
    struct tm     tstruct;
    char          buff2[128];
#else

```

```

        char          *p;
        LCID          localeid = LOCALE_USER_DEFAULT;
        SYSTEMTIME    syst;
    #endif
    char          buff[128];
    #endif

    COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
    COB_FIELD_INIT (0, NULL, &attr);
    cob_exception_code = 0;

    #if defined(_WIN32) || defined(__CYGWIN__) || defined(HAVE_LANGINFO_CODESET)
        if (COB_FIELD_IS_NUMERIC (srcfield)) {
            indate = cob_get_int (srcfield);
        } else {
            if (srcfield->size < 6) {
                goto derror;
            }
            p = srcfield->data;
            indate = 0;
            for (len = 0; len < 6; ++len, ++p) {
                if (isdigit (*p)) {
                    indate *= 10;
                    indate += (*p - '0');
                } else {
                    goto derror;
                }
            }
        }
        hours = indate / 10000;
        if (hours < 0 || hours > 24) {
            goto derror;
        }
        indate %= 10000;
        minutes = indate / 100;
        if (minutes < 0 || minutes > 59) {
            goto derror;
        }
        seconds = indate % 100;
        if (seconds < 0 || seconds > 59) {
            goto derror;
        }
    #endif

    #ifndef HAVE_LANGINFO_CODESET
        memset ((void *)&tstruct, 0, sizeof(struct tm));
        tstruct.tm_hour = hours;
        tstruct.tm_min = minutes;
        tstruct.tm_sec = seconds;
        if (locale_field) {
            if (locale_field->size >= COB_SMALL_BUFF) {
                goto derror;
            }
            cob_field_to_string (locale_field, locale_buff);
            deflocale = locale_buff;
            localep2 = setlocale (LC_TIME, NULL);
            if (localep2) {
                localep = strdup (localep2);
            }
            (void) setlocale (LC_TIME, deflocale);
        }
        memset (buff2, 0, sizeof(buff2));
        snprintf(buff2, sizeof(buff2) - 1, "%s", nl_langinfo(T_FMT));
    #endif

```

```

        if (deflocale) {
            if (localep) {
                (void) setlocale (LC_TIME, localep);
            }
        }
        strftime (buff, sizeof(buff), buff2, &tstruct);
#else
    memset ((void *)&syst, 0, sizeof(syst));
    syst.wHour = hours;
    syst.wMinute = minutes;
    syst.wSecond = seconds;
    if (locale_field) {
        if (locale_field->size >= COB_SMALL_BUFFER) {
            goto derror;
        }
        cob_field_to_string (locale_field, locale_buff);
        for (p = locale_buff; *p; ++p) {
            if (isalnum(*p) || *p == '_') {
                continue;
            }
            break;
        }
        *p = 0;
        for (len = 0; len < WINLOCSIZE; ++len) {
            if (!strcmp(locale_buff, wintable[len].winlocalename)) {
                localeid = wintable[len].winlocaleid;
                break;
            }
        }
        if (len == WINLOCSIZE) {
            goto derror;
        }
    }
    if (!GetTimeFormat (localeid, LOCALE_NOUSEROVERRIDE, &syst, NULL, buff, sizeof(buff))) {
        goto derror;
    }
#endif
    len = strlen (buff);
    field.size = len;
    make_field_entry (&field);
    memcpy (curr_field->data, buff, len);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
derror:
#endif
    field.size = 10;
    make_field_entry (&field);
    memset (curr_field->data, ' ', 10);
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    return curr_field;
}

```

### 7.37.1.32 cob\_field\* cob\_intr\_log ( cob\_field \* )

Definition at line 1690 of file intrinsic.c.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = log (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

### 7.37.1.33 cob\_field\* cob\_intr\_log10 ( cob\_field \* )

Definition at line 1708 of file intrinsic.c.

```

{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = log10 (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}

```

### 7.37.1.34 cob\_field\* cob\_intr\_lower\_case ( const int , const int , cob\_field \* )

Definition at line 605 of file intrinsic.c.

```

{
    size_t          i, size;

    make_field_entry (srcfield);

    size = srcfield->size;
    for (i = 0; i < size; ++i) {
        curr_field->data[i] = tolower (srcfield->data[i]);
    }
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

7.37.1.35 `cob_field* cob_intr_max ( const int, ... )`

Definition at line 2125 of file intrinsic.c.

```

{
    cob_field      *f, *basef;
    va_list        args;
    int            i;

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) > 0) {
            basef = f;
        }
    }
    va_end (args);

    return basef;
}

```

7.37.1.36 `cob_field* cob_intr_mean ( const int, ... )`

Definition at line 2224 of file intrinsic.c.

```

{
    cob_field      *f;
    va_list        args;
    long long      n;
    union {
        unsigned char  data[8];
        long long      datall;
    } datun;
    int            i;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    mpz_set_ui (d1.value, 0);
    d1.scale = 0;

    va_start (args, params);
    for (i = 0; i < params; ++i) {
        f = va_arg (args, cob_field *);
        cob_decimal_set_field (&d2, f);
        cob_decimal_add (&d1, &d2);
    }
    va_end (args);

    mpz_set_ui (d2.value, (unsigned int)params);
    d2.scale = 0;
    cob_decimal_div (&d1, &d2);
    field.data = datun.data;
}

```

```

cob_decimal_get_field (&d1, &field, 0);
n = datun.datall;
for (i = 0; n; n /= 10, ++i) ;
field.data = NULL;
if (i <= 18) {
    attr.scale = 18 - i;
}
make_field_entry (&field);
cob_decimal_get_field (&d1, curr_field, 0);
return curr_field;
}

```

### 7.37.1.37 cob\_field\* cob\_intr\_median ( const int, ... )

Definition at line 2179 of file intrinsic.c.

```

{
    cob_field      *f;
    cob_field      **field_alloc;
    va_list        args;
    int            i;

    va_start (args, params);

    f = va_arg (args, cob_field *);
    if (params == 1) {
        va_end (args);
        return f;
    }

    field_alloc = cob_malloc (params * sizeof (cob_field *));
    field_alloc[0] = f;

    for (i = 1; i < params; ++i) {
        field_alloc[i] = va_arg (args, cob_field *);
    }
    va_end (args);

    qsort (field_alloc, (size_t)params, (size_t)sizeof (cob_field *), comp_fi
eld);

    i = params / 2;
    if (params % 2) {
        f = field_alloc[i];
    } else {
        make_double_entry ();
        cob_decimal_set_field (&d1, field_alloc[i-1]);
        cob_decimal_set_field (&d2, field_alloc[i]);
        cob_decimal_add (&d1, &d2);
        mpz_set_ui (d2.value, 2);
        d2.scale = 0;
        cob_decimal_div (&d1, &d2);
        cob_decimal_get_field (&d1, curr_field, 0);
        f = curr_field;
    }

    free (field_alloc);
    return f;
}

```



**7.37.1.38 cob\_field\* cob\_intr\_midrange ( const int, ... )**

Definition at line 2146 of file intrinsic.c.

```
{
    cob_field      *f, *basemin, *basemax;
    va_list        args;
    int            i;

    make_double_entry ();
    va_start (args, params);

    basemin = va_arg (args, cob_field *);
    basemax = basemin;
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basemin) < 0) {
            basemin = f;
        }
        if (cob_cmp (f, basemax) > 0) {
            basemax = f;
        }
    }
    va_end (args);

    cob_decimal_set_field (&d1, basemin);
    cob_decimal_set_field (&d2, basemax);
    cob_decimal_add (&d1, &d2);
    mpz_set_ui (d2.value, 2);
    d2.scale = 0;
    cob_decimal_div (&d1, &d2);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}
```

**7.37.1.39 cob\_field\* cob\_intr\_min ( const int, ... )**

Definition at line 2104 of file intrinsic.c.

```
{
    cob_field      *f, *basef;
    va_list        args;
    int            i;

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) < 0) {
            basef = f;
        }
    }
    va_end (args);

    return basef;
}
```

7.37.1.40 **cob\_field\*** **cob\_intr\_mod** ( **cob\_field\***, **cob\_field\*** )

Definition at line 2268 of file intrinsic.c.

```

{
    cob_field      *f1;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    make_field_entry (&field);

    f1 = cob_intr_integer (cob_intr_binop (srcfield1, '/', srcfield2));
    cob_decimal_set_field (&d1, srcfield2);
    cob_decimal_set_field (&d2, f1);
    cob_decimal_mul (&d2, &d1);
    cob_decimal_set_field (&d1, srcfield1);
    cob_decimal_sub (&d1, &d2);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

7.37.1.41 **cob\_field\*** **cob\_intr\_numval** ( **cob\_field\*** )

Definition at line 1795 of file intrinsic.c.

```

{
    long long      llval = 0;
    double         val;
    size_t         i;
    int            integer_digits = 0;
    int            decimal_digits = 0;
    int            sign = 0;
    int            decimal_seen = 0;
    cob_field_attr attr;
    cob_field      field;
    unsigned char  integer_buff[64];
    unsigned char  decimal_buff[64];
    unsigned char  final_buff[64];

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    memset (integer_buff, 0, sizeof (integer_buff));
    memset (decimal_buff, 0, sizeof (decimal_buff));
    memset (final_buff, 0, sizeof (final_buff));

    for (i = 0; i < srcfield->size; ++i) {
        if (i < (srcfield->size - 2)) {
            if (strcasecmp ((char *)&srcfield->data[i], "CR") == 0
                || strcasecmp ((char *)&srcfield->data[i], "DB") ==
0) {
                sign = 1;
                break;
            }
        }
    }
}

```

```

        if (srcfield->data[i] == ' ') {
            continue;
        }
        if (srcfield->data[i] == '+') {
            continue;
        }
        if (srcfield->data[i] == '-') {
            sign = 1;
            continue;
        }
        if (srcfield->data[i] == cob_current_module->decimal_point) {
            decimal_seen = 1;
            continue;
        }
        if (srcfield->data[i] >= '0' && srcfield->data[i] <= '9') {
            llval *= 10;
            llval += srcfield->data[i] - '0';
            if (decimal_seen) {
                decimal_buff[decimal_digits++] = srcfield->data[i];
            } else {
                integer_buff[integer_digits++] = srcfield->data[i];
            }
        }
        if ((integer_digits + decimal_digits) > 30) {
            break;
        }
    }
    if (!integer_digits) {
        integer_buff[0] = '0';
    }
    if (!decimal_digits) {
        decimal_buff[0] = '0';
    }
    if (sign) {
        llval = -llval;
    }
    if ((integer_digits + decimal_digits) <= 18) {
        attr.scale = decimal_digits;
        make_field_entry (&field);
        memcpy (curr_field->data, (char *)&llval, 8);
    } else {
        snprintf ((char *)final_buff, 63, "%s%s.%s", sign ? "-" : "",
                integer_buff, decimal_buff);
        sscanf ((char *)final_buff, "%lf", &val);
        make_double_entry ();
        memcpy (curr_field->data, (char *)&val, sizeof (double));
    }
    return curr_field;
}

```

#### 7.37.1.42 cob\_field\* cob\_intr\_numval.c ( cob\_field \*, cob\_field \* )

Definition at line 1875 of file intrinsic.c.

```

{
    unsigned char    *currency_data;
    long long        llval = 0;
}

```

```

double          val;
size_t          i;
int             integer_digits = 0;
int             decimal_digits = 0;
int             sign = 0;
int             decimal_seen = 0;
cob_field_attr attr;
cob_field       field;
unsigned char   integer_buff[64];
unsigned char   decimal_buff[64];
unsigned char   final_buff[64];

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

COB_FIELD_INIT (8, NULL, &attr);
memset (integer_buff, 0, sizeof (integer_buff));
memset (decimal_buff, 0, sizeof (decimal_buff));
memset (final_buff, 0, sizeof (final_buff));

currency_data = NULL;
if (currency) {
    if (currency->size < srcfield->size) {
        currency_data = currency->data;
    }
}
for (i = 0; i < srcfield->size; ++i) {
    if (i < (srcfield->size - 2)) {
        if (strcascmp ((char *)&srcfield->data[i], "CR") == 0
            || strcascmp ((char *)&srcfield->data[i], "DB") ==
0) {
                sign = 1;
                break;
            }
        }
    if (currency_data) {
        if (i < (srcfield->size - currency->size)) {
            if (memcmp ((char *)&srcfield->data[i], currency_
data,
                                currency->size) == 0) {
                i += (currency->size - 1);
                continue;
            }
        }
    }
    if (srcfield->data[i] == ' ') {
        continue;
    }
    if (srcfield->data[i] == '+') {
        continue;
    }
    if (srcfield->data[i] == '-') {
        sign = 1;
        continue;
    }
    if (srcfield->data[i] == cob_current_module->decimal_point) {
        decimal_seen = 1;
        continue;
    }
    if (srcfield->data[i] == cob_current_module->currency_symbol) {
        continue;
    }
    if (srcfield->data[i] >= '0' && srcfield->data[i] <= '9') {

```

```

        llval *= 10;
        llval += srcfield->data[i] - '0';
        if (decimal_seen) {
            decimal_buff[decimal_digits++] = srcfield->data[i];
};
        } else {
            integer_buff[integer_digits++] = srcfield->data[i];
};
        }
    }
    if ((integer_digits + decimal_digits) > 30) {
        break;
    }
}
if (!integer_digits) {
    integer_buff[0] = '0';
}
if (!decimal_digits) {
    decimal_buff[0] = '0';
}
if (sign) {
    llval = -llval;
}
if ((integer_digits + decimal_digits) <= 18) {
    attr.scale = decimal_digits;
    make_field_entry (&field);
    memcpy (curr_field->data, (char *)&llval, 8);
} else {
    sprintf ((char *)final_buff, 63, "%s%s.%s", sign ? "-" : "",
            integer_buff, decimal_buff);
    sscanf ((char *)final_buff, "%lf", &val);
    make_double_entry ();
    memcpy (curr_field->data, (char *)&val, sizeof (double));
}
return curr_field;
}

```

#### 7.37.1.43 `cob_field* cob_intr_ord ( cob_field * )`

Definition at line 1141 of file intrinsic.c.

```

{
    cob_field_attr  attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, (int)(*srcfield->data + 1));
    return curr_field;
}

```

#### 7.37.1.44 `cob_field* cob_intr_ord_max ( const int, ... )`

Definition at line 2069 of file intrinsic.c.

```

{
    cob_field      *f, *basef;
    int            ordmin = 0;
    int            i;
    va_list        args;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    if (params <= 1) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if (cob_cmp (f, basef) > 0) {
            basef = f;
            ordmin = i;
        }
    }
    va_end (args);

    cob_set_int (curr_field, ordmin + 1);
    return curr_field;
}

```

#### 7.37.1.45 `cob_field* cob_intr_ord_min ( const int, ... )`

Definition at line 2034 of file `intrinsic.c`.

```

{
    cob_field      *f, *basef;
    int            i;
    int            ordmin = 0;
    va_list        args;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    if (params <= 1) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    va_start (args, params);

    basef = va_arg (args, cob_field *);
    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);

```

```

        if (cob_cmp (f, basef) < 0) {
            basef = f;
            ordmin = i;
        }
    }
    va_end (args);

    cob_set_int (curr_field, ordmin + 1);
    return curr_field;
}

```

#### 7.37.1.46 cob\_field\* cob\_intr\_present\_value ( const int, ... )

Definition at line 2538 of file intrinsic.c.

```

{
    cob_field      *f;
    va_list        args;
    int            i;

    va_start (args, params);
    make_double_entry ();

    if (params < 2) {
        va_end (args);
        fprintf (stderr, "Wrong number of parameters for FUNCTION PRESENT
-VALUE\n");
        fflush (stderr);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d1, f);
    mpz_set_ui (d2.value, 1);
    d2.scale = 0;
    cob_decimal_add (&d1, &d2);

    mpz_set_ui (d4.value, 0);
    d4.scale = 0;

    for (i = 1; i < params; ++i) {
        f = va_arg (args, cob_field *);
        cob_decimal_set_field (&d2, f);
        mpz_set (d3.value, d1.value);
        d3.scale = d1.scale;
        if (i > 1) {
            mpz_set_ui (d5.value, (unsigned int)i);
            d5.scale = 0;
            cob_decimal_pow (&d3, &d5);
        }
        cob_decimal_div (&d2, &d3);
        cob_decimal_add (&d4, &d2);
    }
    va_end (args);

    cob_decimal_get_field (&d4, curr_field, 0);
    return curr_field;
}

```

7.37.1.47 `cob_field*` `cob_intr_random ( const int, ... )`Definition at line 2352 of file `intrinsic.c`.

```

{
    cob_field      *f;
    va_list        args;
    int            seed = 1;
    int            randnum;
    int            i;
    int            exp10;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 9, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    va_start (args, params);

    if (params) {
        f = va_arg (args, cob_field *);
        seed = cob_get_int (f);
        if (seed < 0) {
            seed = 0;
        }
#ifdef __CYGWIN__
        srandom ((unsigned int)seed);
#else
        srand ((unsigned int)seed);
#endif
    }
    va_end (args);

#ifdef __CYGWIN__
    randnum = (int)random ();
#else
    randnum = rand ();
#endif

    exp10 = 1;
    for (i = 0; i < 10; ++i) {
        if ((randnum / exp10) == 0) {
            break;
        }
        exp10 *= 10;
    }
    if (i == 0) {
        i = 1;
    }
    attr.scale = i;
    make_field_entry (&field);
    *(long long *)curr_field->data = (long long)randnum;
    return curr_field;
}

```

7.37.1.48 `cob_field*` `cob_intr_range ( const int, ... )`Definition at line 2289 of file `intrinsic.c`.

```

{

```



```

cob_field      *f, *basemin, *basemax;
va_list       args;
int           i;
cob_field_attr attr;
cob_field      field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

COB_FIELD_INIT (8, NULL, &attr);
va_start (args, params);

basemin = va_arg (args, cob_field *);
basemax = basemin;
for (i = 1; i < params; ++i) {
    f = va_arg (args, cob_field *);
    if (cob_cmp (f, basemin) < 0) {
        basemin = f;
    }
    if (cob_cmp (f, basemax) > 0) {
        basemax = f;
    }
}
va_end (args);

attr.scale = COB_FIELD_SCALE(basemin);
if (COB_FIELD_SCALE(basemax) > attr.scale) {
    attr.scale = COB_FIELD_SCALE(basemax);
}
make_field_entry (&field);
cob_decimal_set_field (&d1, basemax);
cob_decimal_set_field (&d2, basemin);
cob_decimal_sub (&d1, &d2);
cob_decimal_get_field (&d1, curr_field, 0);
return curr_field;
}

```

#### 7.37.1.49 cob\_field\*cob\_intr\_rem( cob\_field \*, cob\_field \* )

Definition at line 2327 of file intrinsic.c.

```

{
    cob_field      *f1;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    f1 = cob_intr_integer_part (cob_intr_binop (srcfield1, '/', srcfield2));
    cob_decimal_set_field (&d1, srcfield2);
    cob_decimal_set_field (&d2, f1);
    cob_decimal_mul (&d2, &d1);
    cob_decimal_set_field (&d1, srcfield1);
    cob_decimal_sub (&d1, &d2);

    attr.scale = COB_FIELD_SCALE(srcfield1);
    if (COB_FIELD_SCALE(srcfield2) > attr.scale) {
        attr.scale = COB_FIELD_SCALE(srcfield2);
    }
}

```

```

    make_field_entry (&field);
    cob_decimal_get_field (&dl, curr_field, 0);
    return curr_field;
}

```

### 7.37.1.50 `cob_field* cob_intr_reverse ( const int , const int , cob_field * )`

Definition at line 622 of file `intrinsic.c`.

```

{
    size_t          i, size;

    make_field_entry (srcfield);

    size = srcfield->size;
    for (i = 0; i < size; ++i) {
        curr_field->data[i] = srcfield->data[srcfield->size - i - 1];
    }
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

### 7.37.1.51 `cob_field* cob_intr_seconds_from_formatted_time ( cob_field * , cob_field * )`

Definition at line 2803 of file `intrinsic.c`.

```

{
    unsigned char  *p1;
    unsigned char  *p2;
    size_t        n;
    int           seconds = 0;
    int           minutes = 0;
    int           hours = 0;
    int           seconds_seen = 0;
    int           minutes_seen = 0;
    int           hours_seen = 0;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    if (value->size < format->size) {
        cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    p1 = format->data;
    p2 = value->data;
    for (n = 0; n < format->size - 1; ++n, ++p1, ++p2) {
        if (!memcmp (p1, "hh", 2) && !hours_seen) {

```

```

        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            hours = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');
            hours_seen = 1;
            continue;
        }
    }
    if (!memcmp (p1, "mm", 2) && !minutes_seen) {
        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            minutes = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');

            minutes_seen = 1;
            continue;
        }
    }
    if (!memcmp (p1, "ss", 2) && !seconds_seen) {
        if (*p2 >= '0' && *p2 <= '9' &&
            *(p2 + 1) >= '0' && *(p2 + 1) <= '9') {
            seconds = ((*p2 - '0') * 10) + (*(p2 + 1) - '0');

            seconds_seen = 1;
            continue;
        }
    }
}
if (hours_seen && minutes_seen && seconds_seen) {
    seconds += (hours * 3600) + (minutes * 60);
} else {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    seconds = 0;
}
cob_set_int (curr_field, seconds);
return curr_field;
}

```

#### 7.37.1.52 cob\_field\* cob\_intr.seconds\_past\_midnight ( void )

Definition at line 2782 of file intrinsic.c.

```

{
    struct tm      *timeptr;
    time_t        t;
    int           seconds;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    t = time (NULL);
    timeptr = localtime (&t);
    seconds = (timeptr->tm_hour * 3600) + (timeptr->tm_min * 60) +
              timeptr->tm_sec;
    cob_set_int (curr_field, seconds);
    return curr_field;
}

```

**7.37.1.53 cob\_field\* cob\_intr\_sign ( cob\_field \* )**

Definition at line 566 of file intrinsic.c.

```

{
    int          n;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, COB_FLAG_HAVE_SIGN, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_set_int (curr_field, 0);
    n = cob_cmp (srcfield, curr_field);
    if (n < 0) {
        cob_set_int (curr_field, -1);
    } else if (n > 0) {
        cob_set_int (curr_field, 1);
    }

    return curr_field;
}

```

**7.37.1.54 cob\_field\* cob\_intr\_sin ( cob\_field \* )**

Definition at line 1726 of file intrinsic.c.

```

{
    long long    result;
    double       mathd2;
    int          i, tempres;
    cob_field_attr attr;
    cob_field    field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 17, COB_FLAG_HAVE_SIGN, NULL)
;
    COB_FIELD_INIT (8, NULL, &attr);
    cob_decimal_set_field (&d1, srcfield);
    make_field_entry (&field);

    errno = 0;
    mathd2 = sin (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    result = (long long) mathd2;
    mathd2 -= result;
    for (i = 0; i < 17; ++i) {
        mathd2 *= 10;
        tempres = (int) mathd2;
        result *= 10;
        result += tempres;
        mathd2 -= tempres;
    }
    memcpy (curr_field->data, (char *)&result, 8);
    return curr_field;
}

```

**7.37.1.55 cob\_field\* cob\_intr\_sqrt ( cob\_field \* )**

Definition at line 1759 of file intrinsic.c.

```
{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = sqrt (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}
```

**7.37.1.56 cob\_field\* cob\_intr\_standard\_deviation ( const int, ... )**

Definition at line 2472 of file intrinsic.c.

```
{
    cob_field      *f;
    va_list        args;
    int            i;

    va_start (args, params);
    make_double_entry ();

    if (params == 1) {
        va_end (args);
        cob_set_int (curr_field, 0);
        return curr_field;
    }

    /* MEAN for all params */
    mpz_set_ui (d1.value, 0);
    d1.scale = 0;

    for (i = 0; i < params; ++i) {
        f = va_arg (args, cob_field *);
        cob_decimal_set_field (&d2, f);
        cob_decimal_add (&d1, &d2);
    }
    va_end (args);
    mpz_set_ui (d2.value, (unsigned int)params);
    d2.scale = 0;
    cob_decimal_div (&d1, &d2);

    /* Got the MEAN in d1, iterate again */

    mpz_set_ui (d4.value, 0);
    d4.scale = 0;

    va_start (args, params);
```

```

    for (i = 0; i < params; ++i) {
        f = va_arg (args, cob_field *);
        cob_decimal_set_field (&d2, f);
        cob_decimal_sub (&d2, &d1);
        cob_decimal_mul (&d2, &d2);
        cob_decimal_add (&d4, &d2);
    }
    va_end (args);

    mpz_set_ui (d3.value, (unsigned int)params);
    d3.scale = 0;
    cob_decimal_div (&d4, &d3);
    /* We have the VARIANCE in d4, sqrt = STANDARD-DEVIATION */

/* Do not know why this does not work
    d5.scale = d4.scale;
    mpz_mul_ui (d5.value, d4.value, 1000000000);
    mpz_mul_ui (d4.value, d5.value, 1000000000);
    mpz_sqrt (d5.value, d4.value);
    mpz_div_ui (d4.value, d5.value, 1000000000);
    cob_decimal_get_field (&d4, curr_field, 0);
    return curr_field;
*/

    cob_decimal_get_field (&d4, curr_field, 0);
    f = cob_intr_sqrt (curr_field);
    return f;
}

```

### 7.37.1.57 cob\_field\*cob\_intr\_stored\_char.length ( cob\_field \* )

Definition at line 1155 of file intrinsic.c.

```

{
    unsigned char *p;
    int count;
    cob_field_attr attr;
    cob_field field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    count = srcfield->size;
    p = srcfield->data + srcfield->size - 1;
    for (; count > 0; count--, p--) {
        if (*p != ' ') {
            break;
        }
    }
    cob_set_int (curr_field, count);
    return curr_field;
}

```

7.37.1.58 `cob_field* cob_intr_substitute ( const int , const int , const int , ... )`

Definition at line 678 of file intrinsic.c.

```

{
    cob_field      *var;
    cob_field      **f1;
    cob_field      **f2;
    unsigned char  *p1;
    unsigned char  *p2;
    size_t         varsize;
    size_t         calcsize;
    size_t         n;
    size_t         found;
    int            numreps;
    int            i;
    cob_field_attr attr;
    cob_field      field;
    va_list        args;

    numreps = params / 2;
    f1 = cob_malloc (numreps * sizeof (cob_field *));
    f2 = cob_malloc (numreps * sizeof (cob_field *));

    va_start (args, params);

    var = va_arg (args, cob_field *);
    varsize = var->size;

    /* Extract args */
    for (i = 0; i < params - 1; ++i) {
        if ((i % 2) == 0) {
            f1[i / 2] = va_arg (args, cob_field *);
        } else {
            f2[i / 2] = va_arg (args, cob_field *);
        }
    }

    /* Calculate required size */
    calcsize = 0;
    found = 0;
    p1 = var->data;
    for (n = 0; n < varsize; ) {
        for (i = 0; i < numreps; ++i) {
            if (n + f1[i]->size <= varsize) {
                if (!memcmp (p1, f1[i]->data, f1[i]->size)) {
                    p1 += f1[i]->size;
                    n += f1[i]->size;
                    calcsize += f2[i]->size;
                    found = 1;
                    break;
                }
            }
        }
        if (found) {
            found = 0;
            continue;
        }
        ++n;
        ++p1;
        ++calcsize;
    }
}

```

```

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
field.size = calcsize;
make_field_entry (&field);

found = 0;
p1 = var->data;
p2 = curr_field->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!memcmp (p1, f1[i]->data, f1[i]->size)) {
                memcpy (p2, f2[i]->data, f2[i]->size);
                p1 += f1[i]->size;
                p2 += f2[i]->size;
                n += f1[i]->size;
                found = 1;
                break;
            }
        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    *p2++ = *p1++;
}
if (unlikely(offset > 0)) {
    calc_ref_mod (curr_field, offset, length);
}
free (f1);
free (f2);
return curr_field;
}

```

### 7.37.1.59 `cob_field*` `cob_intr_substitute_case` ( `const int`, `const int`, `const int`, ... )

Definition at line 775 of file `intrinsic.c`.

```

{
    cob_field      *var;
    cob_field      **f1;
    cob_field      **f2;
    unsigned char  *p1;
    unsigned char  *p2;
    size_t         varsize;
    size_t         calcsize;
    size_t         n;
    size_t         found;
    int            numreps;
    int            i;
    cob_field_attr attr;
    cob_field      field;
    va_list        args;

    numreps = params / 2;
    f1 = cob_malloc (numreps * sizeof (cob_field *));

```



```

f2 = cob_malloc (numreps * sizeof (cob_field *));

va_start (args, params);

var = va_arg (args, cob_field *);
varsize = var->size;

/* Extract args */
for (i = 0; i < params - 1; ++i) {
    if ((i % 2) == 0) {
        f1[i / 2] = va_arg (args, cob_field *);
    } else {
        f2[i / 2] = va_arg (args, cob_field *);
    }
}

/* Calculate required size */
calcsz = 0;
found = 0;
p1 = var->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!strncasecmp ((const char *)p1,
                (const char *) (f1[i]->data),
                f1[i]->size)) {
                p1 += f1[i]->size;
                n += f1[i]->size;
                calcsz += f2[i]->size;
                found = 1;
                break;
            }
        }
    }
    if (found) {
        found = 0;
        continue;
    }
    ++n;
    ++p1;
    ++calcsz;
}

COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
COB_FIELD_INIT (0, NULL, &attr);
field.size = calcsz;
make_field_entry (&field);

found = 0;
p1 = var->data;
p2 = curr_field->data;
for (n = 0; n < varsize; ) {
    for (i = 0; i < numreps; ++i) {
        if (n + f1[i]->size <= varsize) {
            if (!strncasecmp ((const char *)p1,
                (const char *) (f1[i]->data),
                f1[i]->size)) {
                memcpy (p2, f2[i]->data, f2[i]->size);
                p1 += f1[i]->size;
                p2 += f2[i]->size;
                n += f1[i]->size;
                found = 1;
            }
        }
    }
}

```

```

                                break;
                                }
                                }
                                }
                                if (found) {
                                    found = 0;
                                    continue;
                                }
                                ++n;
                                *p2++ = *p1++;
                            }
                            if (unlikely(offset > 0)) {
                                calc_ref_mod (curr_field, offset, length);
                            }
                            free (f1);
                            free (f2);
                            return curr_field;
                        }
                    }

```

### 7.37.1.60 `cob_field* cob_intr_sum ( const int, ... )`

Definition at line 1996 of file `intrinsic.c`.

```

{
    cob_field      *f;
    va_list        args;
    int            i;
    int            digits = 0;
    int            scale = 0;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

    COB_FIELD_INIT (8, NULL, &attr);
    mpz_set_ui (dl.value, 0);
    dl.scale = 0;

    va_start (args, params);

    for (i = 0; i < params; ++i) {
        f = va_arg (args, cob_field *);
        if ((COB_FIELD_DIGITS(f) - COB_FIELD_SCALE(f)) > digits) {
            digits = COB_FIELD_DIGITS(f) - COB_FIELD_SCALE(f);
        }
        if (COB_FIELD_SCALE(f) > scale) {
            scale = COB_FIELD_SCALE(f);
        }
        cob_decimal_set_field (&d2, f);
        cob_decimal_add (&d1, &d2);
    }
    va_end (args);

    attr.scale = scale;
    make_field_entry (&field);
    cob_decimal_get_field (&d1, curr_field, 0);
    return curr_field;
}

```

**7.37.1.61 cob\_field\* cob\_intr\_tan ( cob\_field \* )**

Definition at line 1777 of file intrinsic.c.

```
{
    double          mathd2;

    cob_decimal_set_field (&d1, srcfield);
    make_double_entry ();

    errno = 0;
    mathd2 = tan (intr_get_double (&d1));
    if (errno) {
        cob_set_int (curr_field, 0);
        return curr_field;
    }
    memcpy (curr_field->data, (char *)&mathd2, 8);
    return curr_field;
}
```

**7.37.1.62 cob\_field\* cob\_intr\_test\_date\_yyyymmdd ( cob\_field \* )**

Definition at line 1409 of file intrinsic.c.

```
{
    int             indate;
    int             days;
    int             month;
    int             year;
    cob_field_attr  attr;
    cob_field       field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    /* Base 1601-01-01 */
    indate = cob_get_int (srcfield);
    year = indate / 10000;
    if (year < 1601 || year > 9999) {
        cob_set_int (curr_field, 1);
        return curr_field;
    }
    indate %= 10000;
    month = indate / 100;
    if (month < 1 || month > 12) {
        cob_set_int (curr_field, 2);
        return curr_field;
    }
    days = indate % 100;
    if (days < 1 || days > 31) {
        cob_set_int (curr_field, 3);
        return curr_field;
    }
    if (leap_year (year)) {
        if (days > leap_month_days[month]) {
            cob_set_int (curr_field, 3);
            return curr_field;
        }
    }
}
```

```

    }
  } else {
    if (days > normal_month_days[month]) {
      cob_set_int (curr_field, 3);
      return curr_field;
    }
  }
  cob_set_int (curr_field, 0);
  return curr_field;
}

```

### 7.37.1.63 cob\_field\*cob\_intr\_test\_day\_yyyyddd ( cob\_field \* )

Definition at line 1456 of file intrinsic.c.

```

{
  int          indate;
  int          days;
  int          year;
  cob_field_attr attr;
  cob_field    field;

  COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
  COB_FIELD_INIT (4, NULL, &attr);
  make_field_entry (&field);

  /* Base 1601-01-01 */
  indate = cob_get_int (srcfield);
  year = indate / 1000;
  if (year < 1601 || year > 9999) {
    cob_set_int (curr_field, 1);
    return curr_field;
  }
  days = indate % 1000;
  if (days < 1 || days > 365 + leap_year (year)) {
    cob_set_int (curr_field, 2);
    return curr_field;
  }
  cob_set_int (curr_field, 0);
  return curr_field;
}

```

### 7.37.1.64 cob\_field\*cob\_intr\_trim ( const int , const int , cob\_field \* , const int )

Definition at line 876 of file intrinsic.c.

```

{
  unsigned char *begin;
  unsigned char *end;
  size_t i;
  size_t size = 0;

  make_field_entry (srcfield);

  for (i = 0; i < srcfield->size; ++i) {

```

```

        if (srcfield->data[i] != ' ') {
            break;
        }
    }
    if (i == srcfield->size) {
        curr_field->size = 1;
        curr_field->data[0] = ' ';
        return curr_field;
    }
    begin = srcfield->data;
    if (direction != 2) {
        for (; *begin == ' '; ++begin) ;
    }
    end = srcfield->data + srcfield->size - 1;
    if (direction != 1) {
        for (; *end == ' '; end--) ;
    }
    for (i = 0; begin <= end; ++begin, ++i) {
        curr_field->data[i] = *begin;
        ++size;
    }
    curr_field->size = size;
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

#### 7.37.1.65 cob\_field\* cob\_intr\_upper\_case ( const int, const int, cob\_field \* )

Definition at line 588 of file intrinsic.c.

```

{
    size_t      i, size;

    make_field_entry (srcfield);

    size = srcfield->size;
    for (i = 0; i < size; ++i) {
        curr_field->data[i] = toupper (srcfield->data[i]);
    }
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}

```

#### 7.37.1.66 cob\_field\* cob\_intr\_variance ( const int, ... )

Definition at line 2403 of file intrinsic.c.

```

{
    cob_field    *f;
    va_list      args;
    long long    n;
}

```

```

union {
    unsigned char    data[8];
    long long        datall;
} datun;
int                i;
cob_field_attr    attr;
cob_field         field;

COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0, COB_FLAG_HAVE_SIGN, NULL);

COB_FIELD_INIT (8, NULL, &attr);
if (params == 1) {
    make_field_entry (&field);
    cob_set_int (curr_field, 0);
    return curr_field;
}

/* MEAN for all params */
mpz_set_ui (d1.value, 0);
d1.scale = 0;

va_start (args, params);
for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_add (&d1, &d2);
}
va_end (args);
mpz_set_ui (d2.value, (unsigned int)params);
d2.scale = 0;
cob_decimal_div (&d1, &d2);

/* Got the MEAN in d1, iterate again */

mpz_set_ui (d4.value, 0);
d4.scale = 0;

va_start (args, params);
for (i = 0; i < params; ++i) {
    f = va_arg (args, cob_field *);
    cob_decimal_set_field (&d2, f);
    cob_decimal_sub (&d2, &d1);
    cob_decimal_mul (&d2, &d2);
    cob_decimal_add (&d4, &d2);
}
va_end (args);

mpz_set_ui (d3.value, (unsigned int)params);
d3.scale = 0;
cob_decimal_div (&d4, &d3);
field.data = datun.data;
cob_decimal_get_field (&d4, &field, 0);
n = datun.datall;
for (i = 0; n; n /= 10, ++i) ;
field.data = NULL;
if (i <= 18) {
    attr.scale = 18 - i;
}
make_field_entry (&field);
cob_decimal_get_field (&d4, curr_field, 0);
return curr_field;

```

```
}

```

### 7.37.1.67 `cob_field* cob_intr_when_compiled ( const int, const int, cob_field * )`

Definition at line 1020 of file intrinsic.c.

```
{
    make_field_entry (f);

    memcpy (curr_field->data, f->data, f->size);
    if (unlikely(offset > 0)) {
        calc_ref_mod (curr_field, offset, length);
    }
    return curr_field;
}
```

### 7.37.1.68 `cob_field* cob_intr_year_to_yyyy ( const int, ... )`

Definition at line 2583 of file intrinsic.c.

```
{
    cob_field      *f;
    struct tm      *timeptr;
    va_list        args;
    time_t         t;
    int            year;
    int            interval;
    int            xqtyear;
    int            maxyear;
    cob_field_attr attr;
    cob_field      field;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 8, 0, 0, NULL);
    COB_FIELD_INIT (4, NULL, &attr);
    make_field_entry (&field);

    cob_exception_code = 0;
    va_start (args, params);
    f = va_arg (args, cob_field *);
    year = cob_get_int (f);
    if (params > 1) {
        f = va_arg (args, cob_field *);
        interval = cob_get_int (f);
    } else {
        interval = 50;
    }
    if (params > 2) {
        f = va_arg (args, cob_field *);
        xqtyear = cob_get_int (f);
    } else {
        t = time (NULL);
        timeptr = localtime (&t);
        xqtyear = 1900 + timeptr->tm_year;
    }
    va_end (args);
}
```

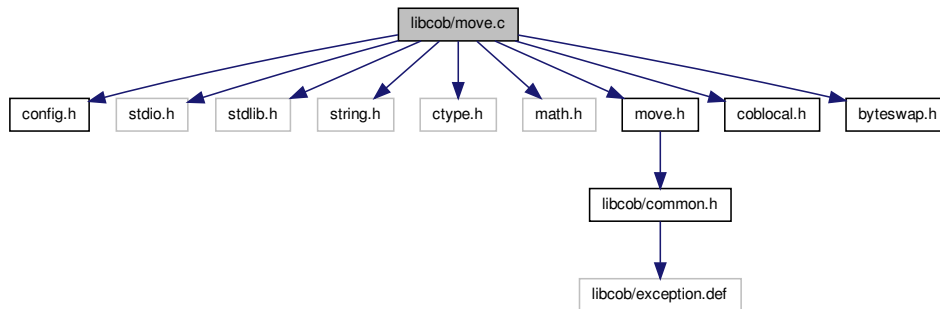
```
if (year < 0 || year > 99) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (xqtyear < 1601 || xqtyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
maxyear = xqtyear + interval;
if (maxyear < 1700 || maxyear > 9999) {
    cob_set_exception (COB_EC_ARGUMENT_FUNCTION);
    cob_set_int (curr_field, 0);
    return curr_field;
}
if (maxyear % 100 >= year) {
    year += 100 * (maxyear / 100);
} else {
    year += 100 * ((maxyear / 100) - 1);
}
cob_set_int (curr_field, year);
return curr_field;
}
```

### 7.38 libcob/move.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include "move.h"
#include "coblocal.h"
#include "byteswap.h"
```



Include dependency graph for move.c:



## Functions

- void `cob_move` (`cob_field *src`, `cob_field *dst`)
- void `cob_set_int` (`cob_field *f`, int `n`)
- int `cob_get_int` (`cob_field *f`)
- long long `cob_get_long_long` (`cob_field *f`)
- void `cob_init_move` (void)

### 7.38.1 Function Documentation

#### 7.38.1.1 int `cob_get_int` (`cob_field * f`)

Definition at line 1304 of file `move.c`.

```

{
    int          n;
    cob_field    temp;
    cob_field_attr attr;

    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_display_get_int (f);
    case COB_TYPE_NUMERIC_BINARY:
        return (int)cob_binary_mget_int64 (f);
    case COB_TYPE_NUMERIC_PACKED:
        return cob_packed_get_int (f);
    default:
        COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 9, 0,
                      COB_FLAG_HAVE_SIGN, NULL);
        temp.size = 4;
        temp.data = (unsigned char *)&n;
        temp.attr = &attr;
        cob_move (f, &temp);
        return n;
    }
}

```

```
    }  
}
```

### 7.38.1.2 long long cob\_get.long\_long ( cob\_field \* f )

Definition at line 1329 of file move.c.

```
{  
    long long      n;  
    cob_field      temp;  
    cob_field_attr attr;  
  
    switch (COB_FIELD_TYPE (f)) {  
    case COB_TYPE_NUMERIC_DISPLAY:  
        return cob_display_get_long_long (f);  
    case COB_TYPE_NUMERIC_BINARY:  
        return cob_binary_mget_int64 (f);  
    case COB_TYPE_NUMERIC_PACKED:  
        return cob_packed_get_long_long (f);  
    default:  
        COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 18, 0,  
                       COB_FLAG_HAVE_SIGN, NULL);  
        temp.size = 8;  
        temp.data = (unsigned char *)&n;  
        temp.attr = &attr;  
        cob_move (f, &temp);  
        return n;  
    }  
}
```

### 7.38.1.3 void cob\_init.move ( void )

Definition at line 1354 of file move.c.

```
{  
    lastdata = cob_malloc (COB_SMALL_BUFF);  
    lastsize = COB_SMALL_BUFF;  
}
```

### 7.38.1.4 void cob\_move ( cob\_field \* src, cob\_field \* dst )

Definition at line 1023 of file move.c.

```
{  
    if (COB_FIELD_TYPE (src) == COB_TYPE_ALPHANUMERIC_ALL) {  
        cob_move_all (src, dst);  
        return;  
    }  
    if (dst->size == 0) {  
        return;  
    }  
    if (src->size == 0) {
```

```

        src = &cob_space;
    }

    /* non-elementary move */
    if (COB_FIELD_TYPE (src) == COB_TYPE_GROUP || COB_FIELD_TYPE (dst) ==
COB_TYPE_GROUP) {
        cob_move_alphanum_to_alphanum (src, dst);
        return;
    }

    /* elementary move */
    switch (COB_FIELD_TYPE (src)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        switch (COB_FIELD_TYPE (dst)) {
        case COB_TYPE_NUMERIC_FLOAT:
        case COB_TYPE_NUMERIC_DOUBLE:
            cob_move_display_to_fp (src, dst);
            return;
        case COB_TYPE_NUMERIC_DISPLAY:
            cob_move_display_to_display (src, dst);
            return;
        case COB_TYPE_NUMERIC_PACKED:
            cob_move_display_to_packed (src, dst);
            return;
        case COB_TYPE_NUMERIC_BINARY:
            cob_move_display_to_binary (src, dst);
            return;
        case COB_TYPE_NUMERIC_EDITED:
            cob_move_display_to_edited (src, dst);
            return;
        case COB_TYPE_ALPHANUMERIC_EDITED:
            if (COB_FIELD_SCALE(src) < 0 ||
                COB_FIELD_SCALE(src) > COB_FIELD_DIGITS(src)) {
                /* expand P's */
                indirect_move (cob_move_display_to_display, src,
dst,
                                (size_t)cob_max_int ((int)
COB_FIELD_DIGITS(src), (int)COB_FIELD_SCALE(src)),
                                cob_max_int (0, (int)
COB_FIELD_SCALE(src)));
                return;
            } else {
                cob_move_alphanum_to_edited (src, dst);
                return;
            }
        default:
            cob_move_display_to_alphanum (src, dst);
            return;
        }

    case COB_TYPE_NUMERIC_PACKED:
        switch (COB_FIELD_TYPE (dst)) {
        case COB_TYPE_NUMERIC_DISPLAY:
            cob_move_packed_to_display (src, dst);
            return;
        default:
            indirect_move (cob_move_packed_to_display, src, dst,
                COB_FIELD_DIGITS(src), COB_FIELD_SCALE(src)
);
            return;
        }
    }
}

```

```

case COB_TYPE_NUMERIC_BINARY:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_binary_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_BINARY:
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_EDITED:
    case COB_TYPE_NUMERIC_FLOAT:
    case COB_TYPE_NUMERIC_DOUBLE:
        indirect_move (cob_move_binary_to_display, src, dst,
            20, COB_FIELD_SCALE(src));
        return;
    default:
        indirect_move (cob_move_binary_to_display, src, dst,
            COB_FIELD_DIGITS(src), COB_FIELD_SCALE(src)
);
    }
    return;
}

case COB_TYPE_NUMERIC_EDITED:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_edited_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_BINARY:
    case COB_TYPE_NUMERIC_EDITED:
    case COB_TYPE_NUMERIC_FLOAT:
    case COB_TYPE_NUMERIC_DOUBLE:
        indirect_move (cob_move_edited_to_display, src, dst, 36,
18);
        return;
    case COB_TYPE_ALPHANUMERIC_EDITED:
        cob_move_alphanum_to_edited (src, dst);
        return;
    default:
        cob_move_alphanum_to_alphanum (src, dst);
        return;
    }

case COB_TYPE_NUMERIC_FLOAT:
case COB_TYPE_NUMERIC_DOUBLE:
    indirect_move (cob_move_fp_to_display, src, dst, 40, 20);
    return;

default:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_alphanum_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_BINARY:
    case COB_TYPE_NUMERIC_EDITED:
    case COB_TYPE_NUMERIC_FLOAT:
    case COB_TYPE_NUMERIC_DOUBLE:
        indirect_move (cob_move_alphanum_to_display, src, dst, 36
, 18);
        return;
    case COB_TYPE_ALPHANUMERIC_EDITED:
        cob_move_alphanum_to_edited (src, dst);
        return;

```

```
        default:
            cob_move_alphanum_to_alphanum (src, dst);
            return;
        }
    }
}
```

#### 7.38.1.5 void cob\_set\_int ( cob\_field \* f, int n )

Definition at line 1291 of file move.c.

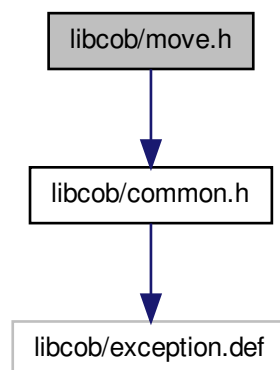
```
{
    cob_field      temp;
    cob_field_attr attr;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 9, 0, COB_FLAG_HAVE_SIGN, NULL);
    temp.size = 4;
    temp.data = (unsigned char *)&n;
    temp.attr = &attr;
    cob_move (&temp, f);
}
```

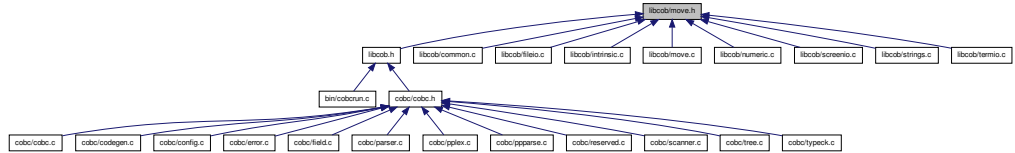
## 7.39 libcob/move.h File Reference

```
#include <libcob/common.h>
```

Include dependency graph for move.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void `cob_move` (`cob_field *`, `cob_field *`)
- void `cob_set_int` (`cob_field *`, int)
- int `cob_get_int` (`cob_field *`)

### 7.39.1 Function Documentation

#### 7.39.1.1 int `cob_get_int` ( `cob_field *` )

Definition at line 1304 of file `move.c`.

```
{
    int          n;
    cob_field    temp;
    cob_field_attr attr;

    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_display_get_int (f);
    case COB_TYPE_NUMERIC_BINARY:
        return (int)cob_binary_mget_int64 (f);
    case COB_TYPE_NUMERIC_PACKED:
        return cob_packed_get_int (f);
    default:
        COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 9, 0,
                      COB_FLAG_HAVE_SIGN, NULL);
        temp.size = 4;
        temp.data = (unsigned char *)&n;
        temp.attr = &attr;
        cob_move (f, &temp);
        return n;
    }
}
```

#### 7.39.1.2 void `cob_move` ( `cob_field *`, `cob_field *` )

Definition at line 1023 of file `move.c`.

```

{
    if (COB_FIELD_TYPE (src) == COB_TYPE_ALPHANUMERIC_ALL) {
        cob_move_all (src, dst);
        return;
    }
    if (dst->size == 0) {
        return;
    }
    if (src->size == 0) {
        src = &cob_space;
    }

    /* non-elementary move */
    if (COB_FIELD_TYPE (src) == COB_TYPE_GROUP || COB_FIELD_TYPE (dst) ==
COB_TYPE_GROUP) {
        cob_move_alphanum_to_alphanum (src, dst);
        return;
    }

    /* elementary move */
    switch (COB_FIELD_TYPE (src)) {
case COB_TYPE_NUMERIC_DISPLAY:
        switch (COB_FIELD_TYPE (dst)) {
case COB_TYPE_NUMERIC_FLOAT:
case COB_TYPE_NUMERIC_DOUBLE:
            cob_move_display_to_fp (src, dst);
            return;
case COB_TYPE_NUMERIC_DISPLAY:
            cob_move_display_to_display (src, dst);
            return;
case COB_TYPE_NUMERIC_PACKED:
            cob_move_display_to_packed (src, dst);
            return;
case COB_TYPE_NUMERIC_BINARY:
            cob_move_display_to_binary (src, dst);
            return;
case COB_TYPE_NUMERIC_EDITED:
            cob_move_display_to_edited (src, dst);
            return;
case COB_TYPE_ALPHANUMERIC_EDITED:
            if (COB_FIELD_SCALE(src) < 0 ||
                COB_FIELD_SCALE(src) > COB_FIELD_DIGITS(src)) {
                /* expand P's */
                indirect_move (cob_move_display_to_display, src,
dst,
                                (size_t)cob_max_int ((int)
COB_FIELD_DIGITS(src), (int)COB_FIELD_SCALE(src)),
                                cob_max_int (0, (int)
COB_FIELD_SCALE(src)));
                return;
            } else {
                cob_move_alphanum_to_edited (src, dst);
                return;
            }
default:
            cob_move_display_to_alphanum (src, dst);
            return;
        }

case COB_TYPE_NUMERIC_PACKED:
        switch (COB_FIELD_TYPE (dst)) {
case COB_TYPE_NUMERIC_DISPLAY:

```

```

        cob_move_packed_to_display (src, dst);
        return;
    default:
        indirect_move (cob_move_packed_to_display, src, dst,
                      COB_FIELD_DIGITS(src), COB_FIELD_SCALE(src)
);
        return;
    }

case COB_TYPE_NUMERIC_BINARY:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_binary_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_BINARY:
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_EDITED:
    case COB_TYPE_NUMERIC_FLOAT:
    case COB_TYPE_NUMERIC_DOUBLE:
        indirect_move (cob_move_binary_to_display, src, dst,
                      20, COB_FIELD_SCALE(src));
        return;
    default:
        indirect_move (cob_move_binary_to_display, src, dst,
                      COB_FIELD_DIGITS(src), COB_FIELD_SCALE(src)
);
        return;
    }

case COB_TYPE_NUMERIC_EDITED:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_edited_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_BINARY:
    case COB_TYPE_NUMERIC_EDITED:
    case COB_TYPE_NUMERIC_FLOAT:
    case COB_TYPE_NUMERIC_DOUBLE:
        indirect_move (cob_move_edited_to_display, src, dst, 36,
);
        return;
    case COB_TYPE_ALPHANUMERIC_EDITED:
        cob_move_alphanum_to_edited (src, dst);
        return;
    default:
        cob_move_alphanum_to_alphanum (src, dst);
        return;
    }

case COB_TYPE_NUMERIC_FLOAT:
case COB_TYPE_NUMERIC_DOUBLE:
    indirect_move (cob_move_fp_to_display, src, dst, 40, 20);
    return;

default:
    switch (COB_FIELD_TYPE (dst)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        cob_move_alphanum_to_display (src, dst);
        return;
    case COB_TYPE_NUMERIC_PACKED:
    case COB_TYPE_NUMERIC_BINARY:

```



```

        case COB_TYPE_NUMERIC_EDITED:
        case COB_TYPE_NUMERIC_FLOAT:
        case COB_TYPE_NUMERIC_DOUBLE:
            indirect_move (cob_move_alphanum_to_display, src, dst, 36
, 18);
            return;
        case COB_TYPE_ALPHANUMERIC_EDITED:
            cob_move_alphanum_to_edited (src, dst);
            return;
        default:
            cob_move_alphanum_to_alphanum (src, dst);
            return;
    }
}

```

### 7.39.1.3 void cob\_set\_int ( cob\_field \*, int )

Definition at line 1291 of file move.c.

```

{
    cob_field      temp;
    cob_field_attr attr;

    COB_ATTR_INIT (COB_TYPE_NUMERIC_BINARY, 9, 0, COB_FLAG_HAVE_SIGN, NULL);
    temp.size = 4;
    temp.data = (unsigned char *)&n;
    temp.attr = &attr;
    cob_move (&temp, f);
}

```

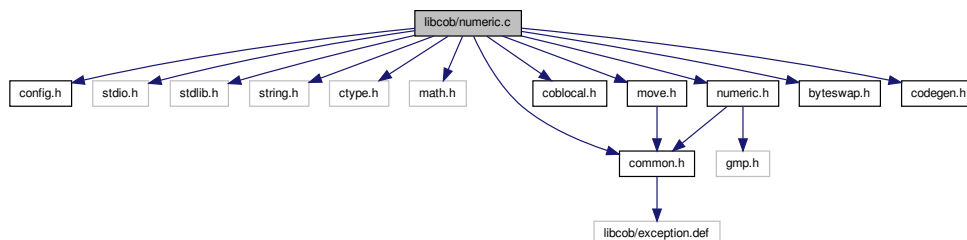
## 7.40 libcob/numeric.c File Reference

```

#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include "common.h"
#include "coblocal.h"
#include "move.h"
#include "numeric.h"
#include "byteswap.h"
#include "codegen.h"

```

Include dependency graph for numeric.c:



## Defines

- #define `COB_LIB_INCLUDE`
- #define `DECIMAL_NAN` -128
- #define `DECIMAL_CHECK(d1, d2)`
- #define `COB_MAX_BINARY` 36

## Functions

- void `cob_decimal_init` (`cob_decimal *d`)
- void `cob_set_packed_zero` (`cob_field *f`)
- void `cob_set_packed_int` (`cob_field *f, const int val`)
- void `cob_decimal_set_field` (`cob_decimal *d, cob_field *f`)
- int `cob_decimal_get_field` (`cob_decimal *d, cob_field *f, const int opt`)
- void `cob_decimal_add` (`cob_decimal *d1, cob_decimal *d2`)
- void `cob_decimal_sub` (`cob_decimal *d1, cob_decimal *d2`)
- void `cob_decimal_mul` (`cob_decimal *d1, cob_decimal *d2`)
- void `cob_decimal_div` (`cob_decimal *d1, cob_decimal *d2`)
- void `cob_decimal_pow` (`cob_decimal *d1, cob_decimal *d2`)
- int `cob_decimal_cmp` (`cob_decimal *d1, cob_decimal *d2`)
- int `cob_add` (`cob_field *f1, cob_field *f2, const int opt`)
- int `cob_sub` (`cob_field *f1, cob_field *f2, const int opt`)
- int `cob_add_int` (`cob_field *f, const int n`)
- int `cob_sub_int` (`cob_field *f, const int n`)
- int `cob_div_quotient` (`cob_field *dividend, cob_field *divisor, cob_field *quotient, const int opt`)
- int `cob_div_remainder` (`cob_field *fld_remainder, const int opt`)
- int `cob_cmp_int` (`cob_field *f1, const int n`)
- int `cob_cmp_uint` (`cob_field *f1, const unsigned int n`)
- int `cob_numeric_cmp` (`cob_field *f1, cob_field *f2`)
- int `cob_cmp_packed` (`cob_field *f, int n`)
- void `cob_init_numeric` (`void`)

- int [cob\\_cmp\\_numdisp](#) (const unsigned char \*data, const size\_t size, const int n)
- int [cob\\_cmp\\_long\\_numdisp](#) (const unsigned char \*data, const size\_t size, const int n)
- int [cob\\_cmp\\_sign\\_numdisp](#) (const unsigned char \*data, const size\_t size, const int n)
- int [cob\\_cmp\\_long\\_sign\\_numdisp](#) (const unsigned char \*data, const size\_t size, const int n)

### 7.40.1 Define Documentation

#### 7.40.1.1 #define COB\_LIB\_INCLUDE

Definition at line 35 of file numeric.c.

#### 7.40.1.2 #define COB\_MAX\_BINARY 36

Definition at line 45 of file numeric.c.

#### 7.40.1.3 #define DECIMAL\_CHECK( d1, d2 )

##### Value:

```
if (unlikely(d1->scale == DECIMAL_NAN || d2->scale == DECIMAL_NAN)) { \
    d1->scale = DECIMAL_NAN; \
    return; \
}
```

Definition at line 39 of file numeric.c.

#### 7.40.1.4 #define DECIMAL\_NAN -128

Definition at line 38 of file numeric.c.

### 7.40.2 Function Documentation

#### 7.40.2.1 int cob.add ( cob\_field \* f1, cob\_field \* f2, const int opt )

Definition at line 1215 of file numeric.c.

```
{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    cob_decimal_add (&cob_d1, &cob_d2);
    return cob_decimal_get_field (&cob_d1, f1, opt);
}
```

**7.40.2.2 int cob\_add\_int ( cob\_field \* f, const int n )**

Definition at line 1233 of file numeric.c.

```

{
    if (unlikely(n == 0)) {
        return 0;
    }
    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_display_add_int (f, n);
    case COB_TYPE_NUMERIC_PACKED:
        cob_add_packed (f, n);
        return 0;
    default:
        /* not optimized */
        cob_decimal_set_field (&cob_d1, f);
        mpz_set_si (cob_d2.value, n);
        cob_d2.scale = 0;
        if (cob_d1.scale) {
            mpz_ui_pow_ui (cob_mexp, 10, (unsigned int)cob_d1.scale);

            mpz_mul (cob_d2.value, cob_d2.value, cob_mexp);
            cob_d2.scale = cob_d1.scale;
        }
        mpz_add (cob_d1.value, cob_d1.value, cob_d2.value);
        return cob_decimal_get_field (&cob_d1, f, 0);
    }
}

```

**7.40.2.3 int cob\_cmp\_int ( cob\_field \* f1, const int n )**

Definition at line 1306 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    mpz_set_si (cob_d2.value, n);
    cob_d2.scale = 0;
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}

```

**7.40.2.4 int cob\_cmp\_long\_numdisp ( const unsigned char \* data, const size\_t size, const int n )**

Definition at line 1431 of file numeric.c.

```

{
    const unsigned char *p;
    long long val = 0;
    size_t inc;

    p = data;
    for (inc = 0; inc < size; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
}

```

```

    }
    return (val < n) ? -1 : (val > n);
}

```

#### 7.40.2.5 int cob\_cmp\_long\_sign\_numdisp ( const unsigned char \* data, const size\_t size, const int n )

Definition at line 1682 of file numeric.c.

```

{
    const unsigned char    *p;
    long long             val = 0;
    size_t                inc;

    p = data;
    for (inc = 0; inc < size - 1; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    val *= 10;
    if (*p >= '0' && *p <= '9') {
        val += (*p - (unsigned char)'0');
    } else {
        if (unlikely(cob_current_module->display_sign)) {
            if (cob_get_long_ebcdic_sign (p, &val)) {
                val = -val;
            }
        } else {
#ifdef COB_EBCDIC_MACHINE
            cob_get_long_ascii_sign (p, &val);
#else
            val += (*p - (unsigned char)'p');
#endif
            val = -val;
        }
    }
    return (val < n) ? -1 : (val > n);
}

```

#### 7.40.2.6 int cob\_cmp\_numdisp ( const unsigned char \* data, const size\_t size, const int n )

Definition at line 1417 of file numeric.c.

```

{
    const unsigned char    *p;
    size_t                 inc;
    int                    val = 0;

    p = data;
    for (inc = 0; inc < size; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    return (val < n) ? -1 : (val > n);
}

```

## 7.40.2.7 int cob\_cmp\_packed ( cob\_field \* f, int n )

Definition at line 1332 of file numeric.c.

```

{
    static int                lastval = 0;

    unsigned char            *p;
    size_t                   size;
    size_t                   inc = 0;
    int                      sign;
    unsigned char            vall[20];

    sign = cob_packed_get_sign (f);
    /* Field positive, value negative */
    if (sign >= 0 && n < 0) {
        return 1;
    }
    /* Field negative, value positive */
    if (sign < 0 && n >= 0) {
        return -1;
    }
    /* Both positive or both negative */
    p = f->data;
    for (size = 0; size < 20; size++) {
        if (size < 20 - f->size) {
            vall[size] = 0;
        } else {
            vall[size] = p[inc++];
        }
    }
    vall[19] &= 0xf0;
    if ((COB_FIELD_DIGITS(f) % 2) == 0) {
        vall[20 - f->size] &= 0x0f;
    }
    if (n != lastval) {
        lastval = n;
        if (n < 0) {
            n = -n;
        }
        memset (&packed_value[14], 0, 6);
        if (n) {
            p = &packed_value[19];
            *p = (n % 10) << 4;
            p--;
            n /= 10;
            for (; n;) {
                size = n % 100;
                *p = (unsigned char)((size % 10) | ((size / 10) <
< 4));
                n /= 100;
                p--;
            }
        }
    }
    for (size = 0; size < 20; size++) {
        if (vall[size] != packed_value[size]) {
            if (sign < 0) {
                return packed_value[size] - vall[size];
            } else {
                return vall[size] - packed_value[size];
            }
        }
    }
}

```

```

    }
    return 0;
}

```

#### 7.40.2.8 int cob\_cmp\_sign\_numdisp ( const unsigned char \* data, const size\_t size, const int n )

Definition at line 1651 of file numeric.c.

```

{
    const unsigned char    *p;
    int                    val = 0;
    size_t                 inc;

    p = data;
    for (inc = 0; inc < size - 1; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    val *= 10;
    if (*p >= '0' && *p <= '9') {
        val += (*p - (unsigned char)'0');
    } else {
        if (unlikely(cob_current_module->display_sign)) {
            if (cob_get_ebcdic_sign (p, &val)) {
                val = -val;
            }
        } else {
#ifdef COB_EBCDIC_MACHINE
            cob_get_ascii_sign (p, &val);
#else
            val += (*p - (unsigned char)'p');
#endif
            val = -val;
        }
    }
    return (val < n) ? -1 : (val > n);
}

```

#### 7.40.2.9 int cob\_cmp\_uint ( cob\_field \* f1, const unsigned int n )

Definition at line 1315 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    mpz_set_ui (cob_d2.value, n);
    cob_d2.scale = 0;
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}

```

#### 7.40.2.10 void cob\_decimal\_add ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 982 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);
    align_decimal (d1, d2);
    mpz_add (d1->value, d1->value, d2->value);
}

```

#### 7.40.2.11 int cob\_decimal\_cmp ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 1043 of file numeric.c.

```

{
    align_decimal (d1, d2);
    return mpz_cmp (d1->value, d2->value);
}

```

#### 7.40.2.12 void cob\_decimal\_div ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 1006 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);

    /* check for division by zero */
    if (unlikely(mpz_sgn (d2->value) == 0)) {
        d1->scale = DECIMAL_NAN;
        cob_set_exception (COB_EC_SIZE_ZERO_DIVIDE);
        return;
    }
    if (unlikely(mpz_sgn (d1->value) == 0)) {
        d1->scale = 0;
        return;
    }
    d1->scale -= d2->scale;
    shift_decimal (d1, 37 + ((d1->scale < 0) ? -d1->scale : 0));
    mpz_tdiv_q (d1->value, d1->value, d2->value);
}

```

#### 7.40.2.13 int cob\_decimal\_get\_field ( cob\_decimal \* d, cob\_field \* f, const int opt )

Definition at line 910 of file numeric.c.

```

{
    cob_field      temp;
    cob_field_attr attr;
    double         val;
    float          fval;
    int            sign;
    unsigned char  data[64];

    if (unlikely(d->scale == DECIMAL_NAN)) {
        cob_set_exception (COB_EC_SIZE_OVERFLOW);
        return cob_exception_code;
    }
}

```



```

    }

    /* work copy */
    if (d != &cob_d1) {
        cob_decimal_set (&cob_d1, d);
        d = &cob_d1;
    }

    /* rounding */
    if (opt & COB_STORE_ROUND) {
        if (COB_FIELD_SCALE(f) < d->scale) {
            sign = mpz_sgn (d->value);
            if (sign != 0) {
                shift_decimal (d, COB_FIELD_SCALE(f) - d->scale +
1);
                if (sign > 0) {
                    mpz_add_ui (d->value, d->value, 5);
                } else {
                    mpz_sub_ui (d->value, d->value, 5);
                }
            }
        }
    }

    /* append or truncate decimal digits */
    shift_decimal (d, COB_FIELD_SCALE(f) - d->scale);

    /* store number */
    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_BINARY:
        return cob_decimal_get_binary (d, f, opt);
    case COB_TYPE_NUMERIC_PACKED:
        return cob_decimal_get_packed (d, f, opt);
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_decimal_get_display (d, f, opt);
    case COB_TYPE_NUMERIC_FLOAT:
        fval = (float) cob_decimal_get_double (d);
        memcpy (f->data, (ucharptr)&fval, sizeof (float));
        return 0;
    case COB_TYPE_NUMERIC_DOUBLE:
        val = cob_decimal_get_double (d);
        memcpy (f->data, (ucharptr)&val, sizeof (double));
        return 0;
    default:
        COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, COB_FIELD_DIGITS(f),
            COB_FIELD_SCALE(f), COB_FLAG_HAVE_SIGN, NULL);
        temp.size = COB_FIELD_DIGITS(f);
        temp.data = data;
        temp.attr = &attr;
        if (cob_decimal_get_display (d, &temp, opt) == 0) {
            cob_move (&temp, f);
        }
        return cob_exception_code;
    }
}

```

#### 7.40.2.14 void cob\_decimal\_init ( cob\_decimal \* d )

Definition at line 277 of file numeric.c.

```
{
    mpz_init2 (d->value, 256);
    d->scale = 0;
}
```

#### 7.40.2.15 void cob\_decimal\_mul ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 998 of file numeric.c.

```
{
    DECIMAL_CHECK (d1, d2);
    d1->scale += d2->scale;
    mpz_mul (d1->value, d1->value, d2->value);
}
```

#### 7.40.2.16 void cob\_decimal\_pow ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 1026 of file numeric.c.

```
{
    unsigned int    n;

    DECIMAL_CHECK (d1, d2);

    if (d2->scale == 0 && mpz_fits_ulong_p (d2->value)) {
        n = mpz_get_ui (d2->value);
        mpz_pow_ui (d1->value, d1->value, n);
        d1->scale *= n;
    } else {
        cob_decimal_set_double (d1, pow (cob_decimal_get_double (d1),
                                         cob_decimal_get_double (d2)));
    }
}
```

#### 7.40.2.17 void cob\_decimal\_set\_field ( cob\_decimal \* d, cob\_field \* f )

Definition at line 883 of file numeric.c.

```
{
    double  dval;
    float   fval;

    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_BINARY:
        cob_decimal_set_binary (d, f);
        break;
    case COB_TYPE_NUMERIC_PACKED:
        cob_decimal_set_packed (d, f);
        break;
    case COB_TYPE_NUMERIC_FLOAT:
        memcpy ((ucharptr)&fval, f->data, sizeof(float));
        cob_decimal_set_double (d, (double)fval);
    }
}
```

```

        break;
    case COB_TYPE_NUMERIC_DOUBLE:
        memcpy ((ucharptr)&dval, f->data, sizeof(double));
        cob_decimal_set_double (d, dval);
        break;
    default:
        cob_decimal_set_display (d, f);
        break;
    }
}

```

#### 7.40.2.18 void cob\_decimal\_sub ( cob\_decimal \* d1, cob\_decimal \* d2 )

Definition at line 990 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);
    align_decimal (d1, d2);
    mpz_sub (d1->value, d1->value, d2->value);
}

```

#### 7.40.2.19 int cob\_div\_quotient ( cob\_field \* dividend, cob\_field \* divisor, cob\_field \* quotient, const int opt )

Definition at line 1269 of file numeric.c.

```

{
    int    ret;

    cob_decimal_set_field (&cob_d1, dividend);
    cob_decimal_set_field (&cob_d2, divisor);
    cob_decimal_set (&cob_d3, &cob_d1);

    /* compute quotient */
    cob_decimal_div (&cob_d1, &cob_d2);
    if (cob_d1.scale == DECIMAL_NAN) {
        cob_d3.scale = DECIMAL_NAN;
        return cob_exception_code;
    }

    /* set quotient */
    cob_decimal_set (&cob_d4, &cob_d1);
    ret = cob_decimal_get_field (&cob_d1, quotient, opt);

    /* truncate digits from the quotient */
    shift_decimal (&cob_d4, COB_FIELD_SCALE(quotient) - cob_d4.scale);

    /* compute remainder */
    cob_decimal_mul (&cob_d4, &cob_d2);
    cob_decimal_sub (&cob_d3, &cob_d4);

    return ret;
}

```

**7.40.2.20 int cob\_div\_remainder ( cob\_field \* fld\_remainder, const int opt )**

Definition at line 1300 of file numeric.c.

```
{
    return cob_decimal_get_field (&cob_d3, fld_remainder, opt);
}
```

**7.40.2.21 void cob\_init\_numeric ( void )**

Definition at line 1396 of file numeric.c.

```
{
    size_t i;

    cob_decimal_init (&cob_d1);
    cob_decimal_init (&cob_d2);
    cob_decimal_init (&cob_d3);
    cob_decimal_init (&cob_d4);
    mpz_init2 (cob_mpzt, 256);
    mpz_init2 (cob_mexp, 512);
    for (i = 0; i < COB_MAX_BINARY; i++) {
        mpz_init (cob_mpze10[i]);
        mpz_ui_pow_ui (cob_mpze10[i], 10, i);
    }
    num_buff_ptr = cob_malloc (2048);
    memset (packed_value, 0, sizeof(packed_value));
}
```

**7.40.2.22 int cob\_numeric\_cmp ( cob\_field \* f1, cob\_field \* f2 )**

Definition at line 1324 of file numeric.c.

```
{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}
```

**7.40.2.23 void cob\_set\_packed\_int ( cob\_field \* f, const int val )**

Definition at line 847 of file numeric.c.

```
{
    unsigned char *p;
    size_t sign = 0;
    int n;

    if (val < 0) {
        n = -val;
    }
}
```

```

        sign = 1;
    } else {
        n = val;
    }
    memset (f->data, 0, f->size);
    p = f->data + f->size - 1;
    *p = (n % 10) << 4;
    if (!COB_FIELD_HAVE_SIGN (f)) {
        *p |= 0x0f;
    } else if (sign) {
        *p |= 0x0d;
    } else {
        *p |= 0x0c;
    }
    n /= 10;
    p--;
    for (; n && p >= f->data; n /= 100, p--) {
        *p = packed_bytes[n % 100];
    }
    /* Fixme */
    if ((COB_FIELD_DIGITS(f) % 2) == 0) {
        *(f->data) &= 0x0f;
    }
}

```

#### 7.40.2.24 void cob\_set\_packed\_zero ( cob\_field \* f )

Definition at line 836 of file numeric.c.

```

{
    memset (f->data, 0, f->size);
    if (!COB_FIELD_HAVE_SIGN (f)) {
        *(f->data + f->size - 1) = 0x0f;
    } else {
        *(f->data + f->size - 1) = 0x0c;
    }
}

```

#### 7.40.2.25 int cob\_sub ( cob\_field \* f1, cob\_field \* f2, const int opt )

Definition at line 1224 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    cob_decimal_sub (&cob_d1, &cob_d2);
    return cob_decimal_get_field (&cob_d1, f1, opt);
}

```

#### 7.40.2.26 int cob\_sub\_int ( cob\_field \* f, const int n )

Definition at line 1260 of file numeric.c.

```

{
    if (unlikely(n == 0)) {
        return 0;
    }
    return cob_add_int (f, -n);
}

```

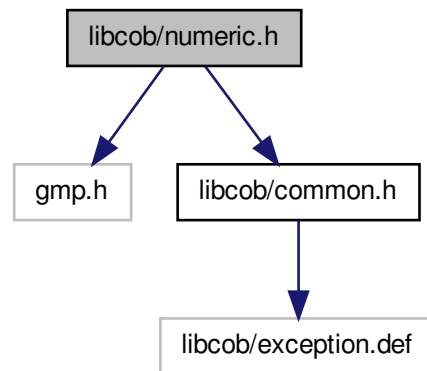
## 7.41 libcob/numeric.h File Reference

```

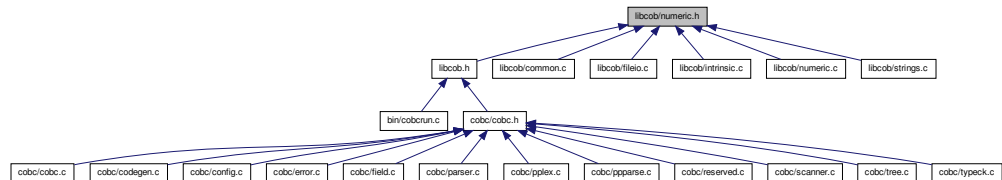
#include <gmp.h>
#include <libcob/common.h>

```

Include dependency graph for numeric.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [cob\\_decimal](#)

## Defines

- #define [COB\\_STORE\\_ROUND](#) 0x01
- #define [COB\\_STORE\\_KEEP\\_ON\\_OVERFLOW](#) 0x02
- #define [COB\\_STORE\\_TRUNC\\_ON\\_OVERFLOW](#) 0x04

## Functions

- void [cob\\_decimal\\_init](#) ([cob\\_decimal](#) \*)
- void [cob\\_decimal\\_set\\_field](#) ([cob\\_decimal](#) \*, [cob\\_field](#) \*)
- int [cob\\_decimal\\_get\\_field](#) ([cob\\_decimal](#) \*, [cob\\_field](#) \*, const int)
- void [cob\\_decimal\\_add](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- void [cob\\_decimal\\_sub](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- void [cob\\_decimal\\_mul](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- void [cob\\_decimal\\_div](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- void [cob\\_decimal\\_pow](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- int [cob\\_decimal\\_cmp](#) ([cob\\_decimal](#) \*, [cob\\_decimal](#) \*)
- int [cob\\_add](#) ([cob\\_field](#) \*, [cob\\_field](#) \*, const int)
- int [cob\\_sub](#) ([cob\\_field](#) \*, [cob\\_field](#) \*, const int)
- int [cob\\_add\\_int](#) ([cob\\_field](#) \*, const int)
- int [cob\\_sub\\_int](#) ([cob\\_field](#) \*, const int)
- int [cob\\_div\\_quotient](#) ([cob\\_field](#) \*, [cob\\_field](#) \*, [cob\\_field](#) \*, const int)
- int [cob\\_div\\_remainder](#) ([cob\\_field](#) \*, const int)
- int [cob\\_cmp\\_int](#) ([cob\\_field](#) \*, const int)
- int [cob\\_cmp\\_uint](#) ([cob\\_field](#) \*, const unsigned int)
- int [cob\\_cmp\\_packed](#) ([cob\\_field](#) \*, int)
- int [cob\\_cmp\\_numdisp](#) (const unsigned char \*, const size\_t, const int)
- int [cob\\_cmp\\_sign\\_numdisp](#) (const unsigned char \*, const size\_t, const int)
- int [cob\\_cmp\\_long\\_numdisp](#) (const unsigned char \*, const size\_t, const int)
- void [cob\\_set\\_packed\\_zero](#) ([cob\\_field](#) \*)
- void [cob\\_set\\_packed\\_int](#) ([cob\\_field](#) \*, const int)
- int [cob\\_cmp\\_long\\_sign\\_numdisp](#) (const unsigned char \*, const size\_t, const int)

### 7.41.1 Define Documentation

#### 7.41.1.1 #define COB\_STORE\_KEEP\_ON\_OVERFLOW 0x02

Definition at line 28 of file numeric.h.

### 7.41.1.2 #define COB\_STORE\_ROUND 0x01

Definition at line 27 of file numeric.h.

### 7.41.1.3 #define COB\_STORE\_TRUNC\_ON\_OVERFLOW 0x04

Definition at line 29 of file numeric.h.

## 7.41.2 Function Documentation

### 7.41.2.1 int cob\_add ( cob\_field \*, cob\_field \*, const int )

Definition at line 1215 of file numeric.c.

```
{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    cob_decimal_add (&cob_d1, &cob_d2);
    return cob_decimal_get_field (&cob_d1, f1, opt);
}
```

### 7.41.2.2 int cob\_add\_int ( cob\_field \*, const int )

Definition at line 1233 of file numeric.c.

```
{
    if (unlikely(n == 0)) {
        return 0;
    }
    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_DISPLAY:
        return cob_display_add_int (f, n);
    case COB_TYPE_NUMERIC_PACKED:
        cob_add_packed (f, n);
        return 0;
    default:
        /* not optimized */
        cob_decimal_set_field (&cob_d1, f);
        mpz_set_si (cob_d2.value, n);
        cob_d2.scale = 0;
        if (cob_d1.scale) {
            mpz_ui_pow_ui (cob_mexp, 10, (unsigned int)cob_d1.scale);

            mpz_mul (cob_d2.value, cob_d2.value, cob_mexp);
            cob_d2.scale = cob_d1.scale;
        }
        mpz_add (cob_d1.value, cob_d1.value, cob_d2.value);
        return cob_decimal_get_field (&cob_d1, f, 0);
    }
}
```



**7.41.2.3 int cob\_cmp\_int ( cob\_field \*, const int )**

Definition at line 1306 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    mpz_set_si (cob_d2.value, n);
    cob_d2.scale = 0;
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}

```

**7.41.2.4 int cob\_cmp\_long\_numdisp ( const unsigned char \*, const size\_t, const int )**

Definition at line 1431 of file numeric.c.

```

{
    const unsigned char    *p;
    long long              val = 0;
    size_t                 inc;

    p = data;
    for (inc = 0; inc < size; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    return (val < n) ? -1 : (val > n);
}

```

**7.41.2.5 int cob\_cmp\_long\_sign\_numdisp ( const unsigned char \*, const size\_t, const int )**

Definition at line 1682 of file numeric.c.

```

{
    const unsigned char    *p;
    long long              val = 0;
    size_t                 inc;

    p = data;
    for (inc = 0; inc < size - 1; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    val *= 10;
    if (*p >= '0' && *p <= '9') {
        val += (*p - (unsigned char)'0');
    } else {
        if (unlikely(cob_current_module->display_sign)) {
            if (cob_get_long_ebcdic_sign (p, &val)) {
                val = -val;
            }
        } else {
#ifdef COB_EBCDIC_MACHINE
            cob_get_long_ascii_sign (p, &val);
#else
            val += (*p - (unsigned char)'p');
#endif
        }
    }
}

```

```

        val = -val;
    }
}
return (val < n) ? -1 : (val > n);
}

```

#### 7.41.2.6 int cob\_cmp\_numdisp ( const unsigned char \*, const size\_t, const int )

Definition at line 1417 of file numeric.c.

```

{
    const unsigned char    *p;
    size_t                inc;
    int                    val = 0;

    p = data;
    for (inc = 0; inc < size; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    return (val < n) ? -1 : (val > n);
}

```

#### 7.41.2.7 int cob\_cmp\_packed ( cob\_field \*, int )

Definition at line 1332 of file numeric.c.

```

{
    static int                lastval = 0;

    unsigned char            *p;
    size_t                   size;
    size_t                   inc = 0;
    int                       sign;
    unsigned char            vall[20];

    sign = cob_packed_get_sign (f);
    /* Field positive, value negative */
    if (sign >= 0 && n < 0) {
        return 1;
    }
    /* Field negative, value positive */
    if (sign < 0 && n >= 0) {
        return -1;
    }
    /* Both positive or both negative */
    p = f->data;
    for (size = 0; size < 20; size++) {
        if (size < 20 - f->size) {
            vall[size] = 0;
        } else {
            vall[size] = p[inc++];
        }
    }
    vall[19] &= 0xf0;
    if ((COB_FIELD_DIGITS(f) % 2) == 0) {

```

```

        vall[20 - f->size] &= 0x0f;
    }
    if (n != lastval) {
        lastval = n;
        if (n < 0) {
            n = -n;
        }
        memset (&packed_value[14], 0, 6);
        if (n) {
            p = &packed_value[19];
            *p = (n % 10) << 4;
            p--;
            n /= 10;
            for (; n;) {
                size = n % 100;
                *p = (unsigned char)((size % 10) | ((size / 10) <
< 4));
                n /= 100;
                p--;
            }
        }
    }
    for (size = 0; size < 20; size++) {
        if (vall[size] != packed_value[size]) {
            if (sign < 0) {
                return packed_value[size] - vall[size];
            } else {
                return vall[size] - packed_value[size];
            }
        }
    }
    return 0;
}

```

#### 7.41.2.8 int cob\_cmp\_sign\_numdisp ( const unsigned char \*, const size\_t, const int )

Definition at line 1651 of file numeric.c.

```

{
    const unsigned char    *p;
    int                    val = 0;
    size_t                 inc;

    p = data;
    for (inc = 0; inc < size - 1; inc++, p++) {
        val = (val * 10) + (*p - (unsigned char)'0');
    }
    val *= 10;
    if (*p >= '0' && *p <= '9') {
        val += (*p - (unsigned char)'0');
    } else {
        if (unlikely(cob_current_module->display_sign)) {
            if (cob_get_ebcdic_sign (p, &val)) {
                val = -val;
            }
        } else {
#ifdef COB_EBCDIC_MACHINE
            cob_get_ascii_sign (p, &val);
#else

```

```

                                val += (*p - (unsigned char)'p');
#endif
                                val = -val;
                                }
                                }
                                return (val < n) ? -1 : (val > n);
                                }

```

#### 7.41.2.9 int cob\_cmp\_uint ( cob\_field \*, const unsigned int )

Definition at line 1315 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    mpz_set_ui (cob_d2.value, n);
    cob_d2.scale = 0;
    return cob_decimal_cmp (&cob_d1, &cob_d2);
}

```

#### 7.41.2.10 void cob\_decimal\_add ( cob\_decimal \*, cob\_decimal \* )

Definition at line 982 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);
    align_decimal (d1, d2);
    mpz_add (d1->value, d1->value, d2->value);
}

```

#### 7.41.2.11 int cob\_decimal\_cmp ( cob\_decimal \*, cob\_decimal \* )

Definition at line 1043 of file numeric.c.

```

{
    align_decimal (d1, d2);
    return mpz_cmp (d1->value, d2->value);
}

```

#### 7.41.2.12 void cob\_decimal\_div ( cob\_decimal \*, cob\_decimal \* )

Definition at line 1006 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);

    /* check for division by zero */
    if (unlikely(mpz_sgn (d2->value) == 0)) {
        d1->scale = DECIMAL_NAN;
    }
}

```

```

        cob_set_exception (COB_EC_SIZE_ZERO_DIVIDE);
        return;
    }
    if (unlikely(mpz_sgn (d1->value) == 0)) {
        d1->scale = 0;
        return;
    }
    d1->scale -= d2->scale;
    shift_decimal (d1, 37 + ((d1->scale < 0) ? -d1->scale : 0));
    mpz_tdiv_q (d1->value, d1->value, d2->value);
}

```

### 7.41.2.13 int cob\_decimal\_get\_field ( cob\_decimal \*, cob\_field \*, const int )

Definition at line 910 of file numeric.c.

```

{
    cob_field      temp;
    cob_field_attr attr;
    double         val;
    float          fval;
    int            sign;
    unsigned char  data[64];

    if (unlikely(d->scale == DECIMAL_NAN)) {
        cob_set_exception (COB_EC_SIZE_OVERFLOW);
        return cob_exception_code;
    }

    /* work copy */
    if (d != &cob_d1) {
        cob_decimal_set (&cob_d1, d);
        d = &cob_d1;
    }

    /* rounding */
    if (opt & COB_STORE_ROUND) {
        if (COB_FIELD_SCALE(f) < d->scale) {
            sign = mpz_sgn (d->value);
            if (sign != 0) {
                shift_decimal (d, COB_FIELD_SCALE(f) - d->scale +
1);
                    if (sign > 0) {
                        mpz_add_ui (d->value, d->value, 5);
                    } else {
                        mpz_sub_ui (d->value, d->value, 5);
                    }
                }
            }
        }

    /* append or truncate decimal digits */
    shift_decimal (d, COB_FIELD_SCALE(f) - d->scale);

    /* store number */
    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_BINARY:
        return cob_decimal_get_binary (d, f, opt);
    case COB_TYPE_NUMERIC_PACKED:

```

```

        return cob_decimal_get_packed (d, f, opt);
case COB_TYPE_NUMERIC_DISPLAY:
    return cob_decimal_get_display (d, f, opt);
case COB_TYPE_NUMERIC_FLOAT:
    fval = (float) cob_decimal_get_double (d);
    memcpy (f->data, (ucharptr)&fval, sizeof (float));
    return 0;
case COB_TYPE_NUMERIC_DOUBLE:
    val = cob_decimal_get_double (d);
    memcpy (f->data, (ucharptr)&val, sizeof (double));
    return 0;
default:
    COB_ATTR_INIT (COB_TYPE_NUMERIC_DISPLAY, COB_FIELD_DIGITS(f),
                  COB_FIELD_SCALE(f), COB_FLAG_HAVE_SIGN, NULL);
    temp.size = COB_FIELD_DIGITS(f);
    temp.data = data;
    temp.attr = &attr;
    if (cob_decimal_get_display (d, &temp, opt) == 0) {
        cob_move (&temp, f);
    }
    return cob_exception_code;
}
}

```

#### 7.41.2.14 void cob\_decimal\_init ( cob\_decimal \* )

Definition at line 277 of file numeric.c.

```

{
    mpz_init2 (d->value, 256);
    d->scale = 0;
}

```

#### 7.41.2.15 void cob\_decimal\_mul ( cob\_decimal \*, cob\_decimal \* )

Definition at line 998 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);
    d1->scale += d2->scale;
    mpz_mul (d1->value, d1->value, d2->value);
}

```

#### 7.41.2.16 void cob\_decimal\_pow ( cob\_decimal \*, cob\_decimal \* )

Definition at line 1026 of file numeric.c.

```

{
    unsigned int    n;

    DECIMAL_CHECK (d1, d2);
}

```

```

    if (d2->scale == 0 && mpz_fits_ulong_p (d2->value)) {
        n = mpz_get_ui (d2->value);
        mpz_pow_ui (d1->value, d1->value, n);
        d1->scale *= n;
    } else {
        cob_decimal_set_double (d1, pow (cob_decimal_get_double (d1),
                                         cob_decimal_get_double (d2)));
    }
}

```

#### 7.41.2.17 void cob\_decimal\_set\_field ( cob\_decimal \*, cob\_field \* )

Definition at line 883 of file numeric.c.

```

{
    double  dval;
    float   fval;

    switch (COB_FIELD_TYPE (f)) {
    case COB_TYPE_NUMERIC_BINARY:
        cob_decimal_set_binary (d, f);
        break;
    case COB_TYPE_NUMERIC_PACKED:
        cob_decimal_set_packed (d, f);
        break;
    case COB_TYPE_NUMERIC_FLOAT:
        memcpy ((ucharptr)&fval, f->data, sizeof(float));
        cob_decimal_set_double (d, (double)fval);
        break;
    case COB_TYPE_NUMERIC_DOUBLE:
        memcpy ((ucharptr)&dval, f->data, sizeof(double));
        cob_decimal_set_double (d, dval);
        break;
    default:
        cob_decimal_set_display (d, f);
        break;
    }
}

```

#### 7.41.2.18 void cob\_decimal\_sub ( cob\_decimal \*, cob\_decimal \* )

Definition at line 990 of file numeric.c.

```

{
    DECIMAL_CHECK (d1, d2);
    align_decimal (d1, d2);
    mpz_sub (d1->value, d1->value, d2->value);
}

```

#### 7.41.2.19 int cob\_div\_quotient ( cob\_field \*, cob\_field \*, cob\_field \*, const int )

Definition at line 1269 of file numeric.c.

```

{
    int    ret;

    cob_decimal_set_field (&cob_d1, dividend);
    cob_decimal_set_field (&cob_d2, divisor);
    cob_decimal_set (&cob_d3, &cob_d1);

    /* compute quotient */
    cob_decimal_div (&cob_d1, &cob_d2);
    if (cob_d1.scale == DECIMAL_NAN) {
        cob_d3.scale = DECIMAL_NAN;
        return cob_exception_code;
    }

    /* set quotient */
    cob_decimal_set (&cob_d4, &cob_d1);
    ret = cob_decimal_get_field (&cob_d1, quotient, opt);

    /* truncate digits from the quotient */
    shift_decimal (&cob_d4, COB_FIELD_SCALE(quotient) - cob_d4.scale);

    /* compute remainder */
    cob_decimal_mul (&cob_d4, &cob_d2);
    cob_decimal_sub (&cob_d3, &cob_d4);

    return ret;
}

```

#### 7.41.2.20 int cob\_div\_remainder ( cob\_field \*, const int )

Definition at line 1300 of file numeric.c.

```

{
    return cob_decimal_get_field (&cob_d3, fld_remainder, opt);
}

```

#### 7.41.2.21 void cob\_set\_packed\_int ( cob\_field \*, const int )

Definition at line 847 of file numeric.c.

```

{
    unsigned char *p;
    size_t        sign = 0;
    int           n;

    if (val < 0) {
        n = -val;
        sign = 1;
    } else {
        n = val;
    }
    memset (f->data, 0, f->size);
    p = f->data + f->size - 1;
    *p = (n % 10) << 4;
    if (!COB_FIELD_HAVE_SIGN (f)) {

```



```

        *p |= 0x0f;
    } else if (sign) {
        *p |= 0x0d;
    } else {
        *p |= 0x0c;
    }
    n /= 10;
    p--;
    for (; n && p >= f->data; n /= 100, p--) {
        *p = packed_bytes[n % 100];
    }
    /* Fixme */
    if ((COB_FIELD_DIGITS(f) % 2) == 0) {
        *(f->data) &= 0x0f;
    }
}

```

#### 7.41.2.22 void cob\_set\_packed\_zero ( cob\_field \* )

Definition at line 836 of file numeric.c.

```

{
    memset (f->data, 0, f->size);
    if (!COB_FIELD_HAVE_SIGN (f)) {
        *(f->data + f->size - 1) = 0x0f;
    } else {
        *(f->data + f->size - 1) = 0x0c;
    }
}

```

#### 7.41.2.23 int cob\_sub ( cob\_field \*, cob\_field \*, const int )

Definition at line 1224 of file numeric.c.

```

{
    cob_decimal_set_field (&cob_d1, f1);
    cob_decimal_set_field (&cob_d2, f2);
    cob_decimal_sub (&cob_d1, &cob_d2);
    return cob_decimal_get_field (&cob_d1, f1, opt);
}

```

#### 7.41.2.24 int cob\_sub\_int ( cob\_field \*, const int )

Definition at line 1260 of file numeric.c.

```

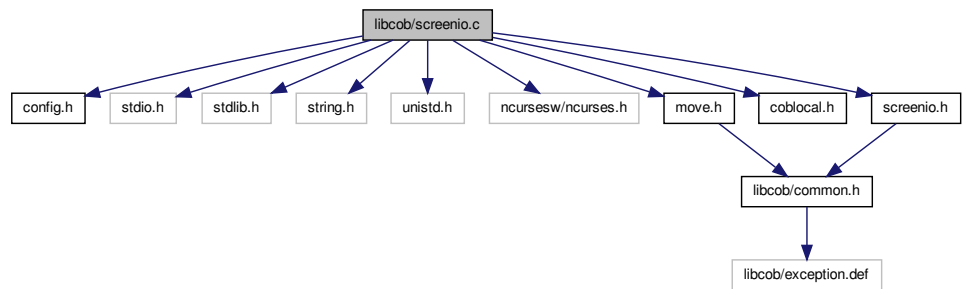
{
    if (unlikely(n == 0)) {
        return 0;
    }
    return cob_add_int (f, -n);
}

```

## 7.42 libcob/screenio.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <ncursesw/ncurses.h>
#include "move.h"
#include "coblocal.h"
#include "screenio.h"
```

Include dependency graph for screenio.c:



### Classes

- struct [cob\\_inp\\_struct](#)

### Defines

- #define [\\_XOPEN\\_SOURCE\\_EXTENDED](#)
- #define [COB\\_GEN\\_SCREENIO](#)
- #define [COB\\_INP\\_SIZE](#) 1920 \* sizeof(struct [cob\\_inp\\_struct](#))

### Functions

- void [cob\\_screen\\_terminate](#) (void)
- void [cob\\_screen\\_display](#) (cob\_screen \*s, cob\_field \*line, cob\_field \*column)
- void [cob\\_screen\\_accept](#) (cob\_screen \*s, cob\_field \*line, cob\_field \*column)

- void `cob_field_display` (`cob_field *f`, `cob_field *line`, `cob_field *column`, `cob_field *fgc`, `cob_field *bgc`, `cob_field *scroll`, `const int attr`)
- void `cob_field_accept` (`cob_field *f`, `cob_field *line`, `cob_field *column`, `cob_field *fgc`, `cob_field *bgc`, `cob_field *scroll`, `const int attr`)
- void `cob_screen_line_col` (`cob_field *f`, `const int l_or_c`)
- void `cob_screen_set_mode` (`const size_t smode`)

## Variables

- int `cob_screen_initialized` = 0
- int `cob_screen_mode` = 0

### 7.42.1 Define Documentation

#### 7.42.1.1 #define \_XOPEN\_SOURCE\_EXTENDED

Definition at line 30 of file screenio.c.

#### 7.42.1.2 #define COB\_GEN\_SCREENIO

Definition at line 33 of file screenio.c.

#### 7.42.1.3 #define COB\_INP\_SIZE 1920 \* sizeof(struct cob\_inp\_struct)

Definition at line 64 of file screenio.c.

### 7.42.2 Function Documentation

#### 7.42.2.1 void `cob_field_accept` ( `cob_field * f`, `cob_field * line`, `cob_field * column`, `cob_field * fgc`, `cob_field * bgc`, `cob_field * scroll`, `const int attr` )

Definition at line 855 of file screenio.c.

```
{
    unsigned char    *p;
    size_t           count;
    int              keyp;
    int              fret;
    int              sline;
    int              scolumn;
    int              cline;
    int              ccolumn;
    int              rightpos;
    int              ateof;
    int              gotbacksp;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
}
```

```

}

if (scroll) {
    keyp = cob_get_int (scroll);
    if (attr & COB_SCREEN_SCROLL_DOWN) {
        keyp = -keyp;
    }
    scrollok (stdscr, 1);
    scrll (keyp);
    scrollok (stdscr, 0);
    refresh ();
}
cob_exception_code = 0;
get_line_column (line, column, &sline, &scolumn);
move (sline, scolumn);
cob_screen_attr (fgc, bgc, attr);
p = f->data;
for (count = 0; count < f->size; count++) {
    if (attr & COB_SCREEN_SECURE) {
        addch ('*');
    } else if (attr & COB_SCREEN_UPDATE) {
        fret = *p++;
        addch ((unsigned int)fret);
    } else if (attr & COB_SCREEN_PROMPT) {
        addch ('_');
    } else {
        addch (' ');
    }
}
move (sline, scolumn);
if (!(attr & COB_SCREEN_UPDATE)) {
    if (COB_FIELD_IS_NUMERIC (f)) {
        cob_move (&cob_zero, f);
    } else {
        memset (f->data, ' ', f->size);
    }
}

fret = 0;
ateof = 0;
gotbacksp = 0;
rightpos = scolumn + f->size - 1;
p = f->data;
for (; ;) {
    refresh ();
    keyp = getch ();
    if (keyp == KEY_ENTER || keyp == '\n') {
        break;
    }
    if (keyp > KEY_F0 && keyp < KEY_F(65)) {
        fret = 1000 + keyp - KEY_F0;
        break;
    }
    if (cob_extended_status) {
        if (keyp == KEY_PPAGE) {
            fret = 2001;
            break;
        }
        if (keyp == KEY_NPAGE) {
            fret = 2002;
            break;
        }
    }
}

```

```
        if (keyp == KEY_UP) {
            fret = 2003;
            break;
        }
        if (keyp == KEY_DOWN) {
            fret = 2004;
            break;
        }
        if (keyp == KEY_PRINT) {
            fret = 2006;
            break;
        }
        if (cob_use_esc) {
            if (keyp == 033) {
                fret = 2005;
                break;
            }
        }
    }
    getyx (stdscr, cline, ccolumn);
    if (keyp == KEY_BACKSPACE || keyp == ('H' & 037) ||
        keyp == 0177) {
        if (ccolumn > scolumn) {
            if (gotbacksp || ccolumn != rightpos) {
                ccolumn--;
            } else {
                ateof = 0;
            }
            gotbacksp = 1;
            move (cline, ccolumn);
            if (attr & COB_SCREEN_SECURE) {
                addch ('*');
            } else {
                addch ('_');
            }
            move (cline, ccolumn);
            p = f->data + ccolumn - scolumn;
            *p = ' ';
            continue;
        }
    }
    if (keyp == KEY_HOME) {
        move (sline, scolumn);
        p = f->data;
        ateof = 0;
        gotbacksp = 0;
        continue;
    }
    if (keyp == KEY_END) {
        move (sline, rightpos);
        p = f->data + f->size - 1;
        ateof = 0;
        gotbacksp = 0;
        continue;
    }
    if (keyp == KEY_LEFT) {
        if (ccolumn > scolumn) {
            ccolumn--;
            move (cline, ccolumn);
            p = f->data + ccolumn - scolumn;
            continue;
        }
    }
}
```

```

        gotbacksp = 0;
    }
    if (keyp == KEY_RIGHT) {
        if (ccolumn < rightpos) {
            ccolumn++;
            move (cline, ccolumn);
            p = f->data + ccolumn - scolumn;
            continue;
        }
        gotbacksp = 0;
    }
    if (keyp > 037 && keyp < (int)A_CHARTEXT) {
        if (COB_FIELD_IS_NUMERIC (f)) {
            if (keyp < '0' || keyp > '9') {
                beep ();
                continue;
            }
        }
        gotbacksp = 0;
        *p = keyp;
        if (attr & COB_SCREEN_SECURE) {
            addch ('*');
        } else {
            addch ((unsigned int)keyp);
        }
        if (ccolumn == rightpos) {
            if (attr & COB_SCREEN_AUTO) {
                break;
            }
            move (cline, ccolumn);
            if (ateof) {
                beep ();
            } else {
                ateof = 1;
            }
        } else {
            p++;
        }
        continue;
    }
    gotbacksp = 0;
    beep ();
}
cob_check_pos_status (fret);
refresh ();
}

```

**7.42.2.2 void cob\_field\_display ( cob\_field \* f, cob\_field \* line, cob\_field \* column, cob\_field \* fg, cob\_field \* bg, cob\_field \* scroll, const int attr )**

Definition at line 826 of file screenio.c.

```

{
    int sline;
    int scolumn;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
}

```

```

    if (scroll) {
        sline = cob_get_int (scroll);
        if (attr & COB_SCREEN_SCROLL_DOWN) {
            sline = -sline;
        }
        scrollok (stdscr, 1);
        scl (sline);
        scrollok (stdscr, 0);
        refresh ();
    }
    get_line_column (line, column, &sline, &scolumn);
    move (sline, scolumn);
    cob_screen_attr (fgc, bgc, attr);
    addnstr ((char *)f->data, (int)f->size);
    refresh ();
}

```

#### 7.42.2.3 void cob\_screen\_accept ( cob\_screen \* s, cob\_field \* line, cob\_field \* column )

Definition at line 759 of file screenio.c.

```

{
    struct cob_inp_struct    *sptr;
    struct cob_inp_struct    *sptr2;
    size_t                   idx;
    size_t                   n;
    size_t                   posu;
    size_t                   posd;
    size_t                   prevy;
    size_t                   firsty;
    int                      starty;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
    if (!cob_base_inp) {
        cob_base_inp = cob_malloc (COB_INP_SIZE);
    } else {
        memset (cob_base_inp, 0, COB_INP_SIZE);
    }
    cob_exception_code = 0;
    cob_current_y = 0;
    cob_current_x = 0;
    totl_index = 0;
    move (0, 0);
    cob_prep_input (s);
    /* No input fields is an error */
    if (!totl_index) {
        cob_check_pos_status (8000);
        return;
    }
    qsort (cob_base_inp, totl_index, sizeof(struct cob_inp_struct), compare_y
x);
    sptr = cob_base_inp;
    starty = sptr->this_y;
    posu = 0;
    posd = 0;
    prevy = 0;
}

```

```

firsty = 0;
/* Set up array for Cursor UP/DOWN */
for (n = 0; n < totl_index; n++) {
    sptr = cob_base_inp + n;
    if (sptr->this_y > starty) {
        if (!firsty) {
            firsty = n;
        }
        starty = sptr->this_y;
        sptr2 = cob_base_inp + posd;
        for (idx = posd; idx < n; idx++, sptr2++) {
            sptr2->down_index = n;
        }
        posu = prevy;
        prevy = n;
        posd = n;
    }
    sptr->up_index = posu;
}
sptr = cob_base_inp;
for (n = 0; n < firsty; n++, sptr++) {
    sptr->up_index = posd;
}
curr_index = 0;
global_return = 0;
cob_screen_get_all ();
cob_check_pos_status (global_return);
}

```

**7.42.2.4 void cob\_screen.display ( cob\_screen \* s, cob\_field \* line, cob\_field \* column )**

Definition at line 726 of file screenio.c.

```

{
    int    n;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }

    switch (s->type) {
    case COB_SCREEN_TYPE_GROUP:
        for (s = s->child; s; s = s->next) {
            cob_screen_display (s, line, column);
        }
        break;
    case COB_SCREEN_TYPE_FIELD:
        cob_screen_puts (s, s->field);
        break;
    case COB_SCREEN_TYPE_VALUE:
        cob_screen_puts (s, s->value);
        if (s->occurs) {
            for (n = 1; n < s->occurs; ++n) {
                cob_screen_puts (s, s->value);
            }
        }
        break;
    case COB_SCREEN_TYPE_ATTRIBUTE:

```



```
        cob_screen_attr (s->foreg, s->backg, s->attr);
        break;
    }
    refresh ();
}
```

#### 7.42.2.5 void cob\_screen\_line\_col ( cob\_field \* f, const int l\_or\_c )

Definition at line 1045 of file screenio.c.

```
{
    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
    if (!l_or_c) {
        cob_set_int (f, (int)LINES);
    } else {
        cob_set_int (f, (int)COLS);
    }
}
```

#### 7.42.2.6 void cob\_screen\_set\_mode ( const size\_t smode )

Definition at line 1058 of file screenio.c.

```
{
    if (!smode) {
        refresh ();
        def_prog_mode ();
        endwin ();
    } else {
        reset_prog_mode ();
        refresh ();
    }
}
```

#### 7.42.2.7 void cob\_screen\_terminate ( void )

Definition at line 297 of file screenio.c.

```
{
    if (cob_screen_initialized) {
        cob_screen_initialized = 0;
        endwin ();
    }
}
```

### 7.42.3 Variable Documentation

#### 7.42.3.1 int cob\_screen\_initialized = 0

Definition at line 51 of file screenio.c.

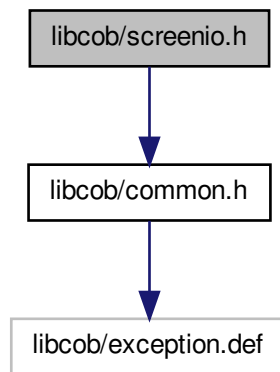
### 7.42.3.2 int cob\_screen\_mode = 0

Definition at line 52 of file screenio.c.

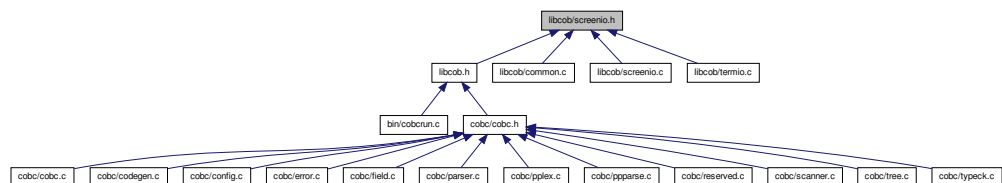
## 7.43 libcob/screenio.h File Reference

```
#include <libcob/common.h>
```

Include dependency graph for screenio.h:



This graph shows which files directly or indirectly include this file:



## Classes

- [struct \\_\\_cob\\_screen](#)

## Defines

- #define [COB\\_SCREEN\\_BLACK](#) 0
- #define [COB\\_SCREEN\\_BLUE](#) 1
- #define [COB\\_SCREEN\\_GREEN](#) 2
- #define [COB\\_SCREEN\\_CYAN](#) 3
- #define [COB\\_SCREEN\\_RED](#) 4
- #define [COB\\_SCREEN\\_MAGENTA](#) 5
- #define [COB\\_SCREEN\\_YELLOW](#) 6
- #define [COB\\_SCREEN\\_WHITE](#) 7
- #define [COB\\_SCREEN\\_LINE\\_PLUS](#) 0x00000001
- #define [COB\\_SCREEN\\_LINE\\_MINUS](#) 0x00000002
- #define [COB\\_SCREEN\\_COLUMN\\_PLUS](#) 0x00000004
- #define [COB\\_SCREEN\\_COLUMN\\_MINUS](#) 0x00000008
- #define [COB\\_SCREEN\\_AUTO](#) 0x00000010
- #define [COB\\_SCREEN\\_BELL](#) 0x00000020
- #define [COB\\_SCREEN\\_BLANK\\_LINE](#) 0x00000040
- #define [COB\\_SCREEN\\_BLANK\\_SCREEN](#) 0x00000080
- #define [COB\\_SCREEN\\_BLINK](#) 0x00000100
- #define [COB\\_SCREEN\\_ERASE\\_EOL](#) 0x00000200
- #define [COB\\_SCREEN\\_ERASE\\_EOS](#) 0x00000400
- #define [COB\\_SCREEN\\_FULL](#) 0x00000800
- #define [COB\\_SCREEN\\_HIGHLIGHT](#) 0x00001000
- #define [COB\\_SCREEN\\_LOWLIGHT](#) 0x00002000
- #define [COB\\_SCREEN\\_REQUIRED](#) 0x00004000
- #define [COB\\_SCREEN\\_REVERSE](#) 0x00008000
- #define [COB\\_SCREEN\\_SECURE](#) 0x00010000
- #define [COB\\_SCREEN\\_UNDERLINE](#) 0x00020000
- #define [COB\\_SCREEN\\_OVERLINE](#) 0x00040000
- #define [COB\\_SCREEN\\_PROMPT](#) 0x00080000
- #define [COB\\_SCREEN\\_UPDATE](#) 0x00100000
- #define [COB\\_SCREEN\\_INPUT](#) 0x00200000
- #define [COB\\_SCREEN\\_SCROLL\\_DOWN](#) 0x00400000
- #define [COB\\_SCREEN\\_TYPE\\_GROUP](#) 0
- #define [COB\\_SCREEN\\_TYPE\\_FIELD](#) 1
- #define [COB\\_SCREEN\\_TYPE\\_VALUE](#) 2
- #define [COB\\_SCREEN\\_TYPE\\_ATTRIBUTE](#) 3

## Typedefs

- typedef struct [\\_\\_cob\\_screen](#) [cob\\_screen](#)

## Functions

- void `cob_screen_line_col` (`cob_field *`, `const int`)
- void `cob_screen_display` (`cob_screen *`, `cob_field *`, `cob_field *`)
- void `cob_screen_accept` (`cob_screen *`, `cob_field *`, `cob_field *`)
- void `cob_field_display` (`cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `const int`)
- void `cob_field_accept` (`cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `cob_field *`, `const int`)

## Variables

- `DLL_EXPIMP int cob_screen_mode`

### 7.43.1 Define Documentation

#### 7.43.1.1 `#define COB_SCREEN_AUTO 0x00000010`

Definition at line 40 of file `screenio.h`.

#### 7.43.1.2 `#define COB_SCREEN_BELL 0x00000020`

Definition at line 41 of file `screenio.h`.

#### 7.43.1.3 `#define COB_SCREEN_BLACK 0`

Definition at line 26 of file `screenio.h`.

#### 7.43.1.4 `#define COB_SCREEN_BLANK_LINE 0x00000040`

Definition at line 42 of file `screenio.h`.

#### 7.43.1.5 `#define COB_SCREEN_BLANK_SCREEN 0x00000080`

Definition at line 43 of file `screenio.h`.

#### 7.43.1.6 `#define COB_SCREEN_BLINK 0x00000100`

Definition at line 44 of file `screenio.h`.

#### 7.43.1.7 `#define COB_SCREEN_BLUE 1`

Definition at line 27 of file `screenio.h`.

7.43.1.8 `#define COB_SCREEN_COLUMN_MINUS 0x00000008`

Definition at line 38 of file screenio.h.

7.43.1.9 `#define COB_SCREEN_COLUMN_PLUS 0x00000004`

Definition at line 37 of file screenio.h.

7.43.1.10 `#define COB_SCREEN_CYAN 3`

Definition at line 29 of file screenio.h.

7.43.1.11 `#define COB_SCREEN_ERASE_EOL 0x00000200`

Definition at line 45 of file screenio.h.

7.43.1.12 `#define COB_SCREEN_ERASE_EOS 0x00000400`

Definition at line 46 of file screenio.h.

7.43.1.13 `#define COB_SCREEN_FULL 0x00000800`

Definition at line 47 of file screenio.h.

7.43.1.14 `#define COB_SCREEN_GREEN 2`

Definition at line 28 of file screenio.h.

7.43.1.15 `#define COB_SCREEN_HIGHLIGHT 0x00001000`

Definition at line 48 of file screenio.h.

7.43.1.16 `#define COB_SCREEN_INPUT 0x00200000`

Definition at line 57 of file screenio.h.

7.43.1.17 `#define COB_SCREEN_LINE_MINUS 0x00000002`

Definition at line 36 of file screenio.h.

7.43.1.18 `#define COB_SCREEN_LINE_PLUS 0x00000001`

Definition at line 35 of file screenio.h.

7.43.1.19 `#define COB_SCREEN_LOWLIGHT 0x00002000`

Definition at line 49 of file screenio.h.

7.43.1.20 `#define COB_SCREEN_MAGENTA 5`

Definition at line 31 of file screenio.h.

7.43.1.21 `#define COB_SCREEN_OVERLINE 0x00040000`

Definition at line 54 of file screenio.h.

7.43.1.22 `#define COB_SCREEN_PROMPT 0x00080000`

Definition at line 55 of file screenio.h.

7.43.1.23 `#define COB_SCREEN_RED 4`

Definition at line 30 of file screenio.h.

7.43.1.24 `#define COB_SCREEN_REQUIRED 0x00004000`

Definition at line 50 of file screenio.h.

7.43.1.25 `#define COB_SCREEN_REVERSE 0x00008000`

Definition at line 51 of file screenio.h.

7.43.1.26 `#define COB_SCREEN_SCROLL_DOWN 0x00400000`

Definition at line 58 of file screenio.h.

7.43.1.27 `#define COB_SCREEN_SECURE 0x00010000`

Definition at line 52 of file screenio.h.

7.43.1.28 `#define COB_SCREEN_TYPE_ATTRIBUTE 3`

Definition at line 63 of file screenio.h.

7.43.1.29 `#define COB_SCREEN_TYPE_FIELD 1`

Definition at line 61 of file screenio.h.

7.43.1.30 `#define COB_SCREEN_TYPE_GROUP 0`

Definition at line 60 of file screenio.h.

7.43.1.31 `#define COB_SCREEN_TYPE_VALUE 2`

Definition at line 62 of file screenio.h.

7.43.1.32 `#define COB_SCREEN_UNDERLINE 0x00020000`

Definition at line 53 of file screenio.h.

7.43.1.33 `#define COB_SCREEN_UPDATE 0x00100000`

Definition at line 56 of file screenio.h.

7.43.1.34 `#define COB_SCREEN_WHITE 7`

Definition at line 33 of file screenio.h.

7.43.1.35 `#define COB_SCREEN_YELLOW 6`

Definition at line 32 of file screenio.h.

## 7.43.2 Typedef Documentation

7.43.2.1 `typedef struct __cob_screen cob_screen`

Definition at line 65 of file screenio.h.

## 7.43.3 Function Documentation

### 7.43.3.1 void cob\_field\_accept ( cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, const int )

Definition at line 855 of file screenio.c.

```

{
    unsigned char    *p;
    size_t          count;
    int             keyp;
    int             fret;
    int             sline;
    int             scolumn;
    int             cline;
    int             ccolumn;
    int             rightpos;
    int             ateof;
    int             gotbacksp;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }

    if (scroll) {
        keyp = cob_get_int (scroll);
        if (attr & COB_SCREEN_SCROLL_DOWN) {
            keyp = -keyp;
        }
        scrollok (stdscr, 1);
        scl (keyp);
        scrollok (stdscr, 0);
        refresh ();
    }
    cob_exception_code = 0;
    get_line_column (line, column, &sline, &scolumn);
    move (sline, scolumn);
    cob_screen_attr (fgc, bgc, attr);
    p = f->data;
    for (count = 0; count < f->size; count++) {
        if (attr & COB_SCREEN_SECURE) {
            addch ('*');
        } else if (attr & COB_SCREEN_UPDATE) {
            fret = *p++;
            addch ((unsigned int)fret);
        } else if (attr & COB_SCREEN_PROMPT) {
            addch ('_');
        } else {
            addch (' ');
        }
    }
    move (sline, scolumn);
    if (!(attr & COB_SCREEN_UPDATE)) {
        if (COB_FIELD_IS_NUMERIC (f)) {
            cob_move (&cob_zero, f);
        } else {
            memset (f->data, ' ', f->size);
        }
    }

    fret = 0;
    ateof = 0;
    gotbacksp = 0;
}

```



```

rightpos = scolumn + f->size - 1;
p = f->data;
for (; ; ) {
    refresh ();
    keyp = getch ();
    if (keyp == KEY_ENTER || keyp == '\n') {
        break;
    }
    if (keyp > KEY_F0 && keyp < KEY_F(65)) {
        fret = 1000 + keyp - KEY_F0;
        break;
    }
    if (cob_extended_status) {
        if (keyp == KEY_PPAGE) {
            fret = 2001;
            break;
        }
        if (keyp == KEY_NPAGE) {
            fret = 2002;
            break;
        }
        if (keyp == KEY_UP) {
            fret = 2003;
            break;
        }
        if (keyp == KEY_DOWN) {
            fret = 2004;
            break;
        }
        if (keyp == KEY_PRINT) {
            fret = 2006;
            break;
        }
        if (cob_use_esc) {
            if (keyp == 033) {
                fret = 2005;
                break;
            }
        }
    }
}
getyx (stdscr, cline, ccolumn);
if (keyp == KEY_BACKSPACE || keyp == ('H' & 037) ||
    keyp == 0177) {
    if (ccolumn > scolumn) {
        if (gotbacksp || ccolumn != rightpos) {
            ccolumn--;
        } else {
            ateof = 0;
        }
        gotbacksp = 1;
        move (cline, ccolumn);
        if (attr & COB_SCREEN_SECURE) {
            addch ('*');
        } else {
            addch ('_');
        }
        move (cline, ccolumn);
        p = f->data + ccolumn - scolumn;
        *p = ' ';
        continue;
    }
}

```

```

if (keyp == KEY_HOME) {
    move (sline, scolumn);
    p = f->data;
    ateof = 0;
    gotbacksp = 0;
    continue;
}
if (keyp == KEY_END) {
    move (sline, rightpos);
    p = f->data + f->size - 1;
    ateof = 0;
    gotbacksp = 0;
    continue;
}
if (keyp == KEY_LEFT) {
    if (ccolumn > scolumn) {
        ccolumn--;
        move (cline, ccolumn);
        p = f->data + ccolumn - scolumn;
        continue;
    }
    gotbacksp = 0;
}
if (keyp == KEY_RIGHT) {
    if (ccolumn < rightpos) {
        ccolumn++;
        move (cline, ccolumn);
        p = f->data + ccolumn - scolumn;
        continue;
    }
    gotbacksp = 0;
}
if (keyp > 037 && keyp < (int)A_CHARTEXT) {
    if (COB_FIELD_IS_NUMERIC (f)) {
        if (keyp < '0' || keyp > '9') {
            beep ();
            continue;
        }
    }
    gotbacksp = 0;
    *p = keyp;
    if (attr & COB_SCREEN_SECURE) {
        addch ('*');
    } else {
        addch ((unsigned int)keyp);
    }
    if (ccolumn == rightpos) {
        if (attr & COB_SCREEN_AUTO) {
            break;
        }
        move (cline, ccolumn);
        if (ateof) {
            beep ();
        } else {
            ateof = 1;
        }
    } else {
        p++;
    }
    continue;
}
gotbacksp = 0;

```

```

        beep ();
    }
    cob_check_pos_status (fret);
    refresh ();
}

```

### 7.43.3.2 void cob\_field\_display ( cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, cob\_field \*, const int )

Definition at line 826 of file screenio.c.

```

{
    int sline;
    int scolumn;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }

    if (scroll) {
        sline = cob_get_int (scroll);
        if (attr & COB_SCREEN_SCROLL_DOWN) {
            sline = -sline;
        }
        scrollok (stdscr, 1);
        scl (sline);
        scrollok (stdscr, 0);
        refresh ();
    }
    get_line_column (line, column, &sline, &scolumn);
    move (sline, scolumn);
    cob_screen_attr (fgc, bgc, attr);
    addnstr ((char *)f->data, (int)f->size);
    refresh ();
}

```

### 7.43.3.3 void cob\_screen\_accept ( cob\_screen \*, cob\_field \*, cob\_field \* )

Definition at line 759 of file screenio.c.

```

{
    struct cob_inp_struct *sptr;
    struct cob_inp_struct *sptr2;
    size_t idx;
    size_t n;
    size_t posu;
    size_t posd;
    size_t prevy;
    size_t firsty;
    int starty;

    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
    if (!cob_base_inp) {

```

```

        cob_base_inp = cob_malloc (COB_INP_SIZE);
    } else {
        memset (cob_base_inp, 0, COB_INP_SIZE);
    }
    cob_exception_code = 0;
    cob_current_y = 0;
    cob_current_x = 0;
    totl_index = 0;
    move (0, 0);
    cob_prep_input (s);
    /* No input fields is an error */
    if (!totl_index) {
        cob_check_pos_status (8000);
        return;
    }
    qsort (cob_base_inp, totl_index, sizeof(struct cob_inp_struct), compare_y
x);
    sptr = cob_base_inp;
    starty = sptr->this_y;
    posu = 0;
    posd = 0;
    prevy = 0;
    firsty = 0;
    /* Set up array for Cursor UP/DOWN */
    for (n = 0; n < totl_index; n++) {
        sptr = cob_base_inp + n;
        if (sptr->this_y > starty) {
            if (!firsty) {
                firsty = n;
            }
            starty = sptr->this_y;
            sptr2 = cob_base_inp + posd;
            for (idx = posd; idx < n; idx++, sptr2++) {
                sptr2->down_index = n;
            }
            posu = prevy;
            prevy = n;
            posd = n;
        }
        sptr->up_index = posu;
    }
    sptr = cob_base_inp;
    for (n = 0; n < firsty; n++, sptr++) {
        sptr->up_index = posd;
    }
    curr_index = 0;
    global_return = 0;
    cob_screen_get_all ();
    cob_check_pos_status (global_return);
}

```

#### 7.43.3.4 void cob\_screen\_display ( cob\_screen \*, cob\_field \*, cob\_field \* )

Definition at line 726 of file screenio.c.

```

{
    int    n;

    if (!cob_screen_initialized) {

```

```

        cob_screen_init ();
    }

    switch (s->type) {
    case COB_SCREEN_TYPE_GROUP:
        for (s = s->child; s; s = s->next) {
            cob_screen_display (s, line, column);
        }
        break;
    case COB_SCREEN_TYPE_FIELD:
        cob_screen_puts (s, s->field);
        break;
    case COB_SCREEN_TYPE_VALUE:
        cob_screen_puts (s, s->value);
        if (s->occurs) {
            for (n = 1; n < s->occurs; ++n) {
                cob_screen_puts (s, s->value);
            }
        }
        break;
    case COB_SCREEN_TYPE_ATTRIBUTE:
        cob_screen_attr (s->foreg, s->backg, s->attr);
        break;
    }
    refresh ();
}

```

#### 7.43.3.5 void cob\_screen\_line\_col ( cob\_field \*, const int )

Definition at line 1045 of file screenio.c.

```

{
    if (!cob_screen_initialized) {
        cob_screen_init ();
    }
    if (!l_or_c) {
        cob_set_int (f, (int)LINES);
    } else {
        cob_set_int (f, (int)COLS);
    }
}

```

### 7.43.4 Variable Documentation

#### 7.43.4.1 DLL\_EXPIMP int cob\_screen\_mode

Definition at line 52 of file screenio.c.

## 7.44 libcob/strings.c File Reference

```

#include "config.h"
#include <stdio.h>

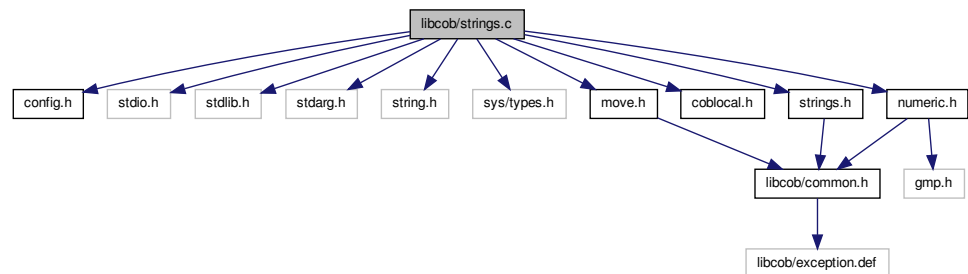
```

```

#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include "move.h"
#include "coblocal.h"
#include "numeric.h"
#include "strings.h"

```

Include dependency graph for strings.c:



## Classes

- struct [dlm\\_struct](#)

## Defines

- #define [INSPECT\\_ALL](#) 0
- #define [INSPECT\\_LEADING](#) 1
- #define [INSPECT\\_FIRST](#) 2
- #define [INSPECT\\_TRAILING](#) 3
- #define [DLM\\_DEFAULT\\_NUM](#) 8

## Functions

- void [cob\\_inspect\\_init](#) ([cob\\_field](#) \*var, const int replacing)
- void [cob\\_inspect\\_start](#) (void)
- void [cob\\_inspect\\_before](#) (const [cob\\_field](#) \*str)
- void [cob\\_inspect\\_after](#) (const [cob\\_field](#) \*str)
- void [cob\\_inspect\\_characters](#) ([cob\\_field](#) \*f1)

- void `cob_inspect_all` (`cob_field *f1`, `cob_field *f2`)
- void `cob_inspect_leading` (`cob_field *f1`, `cob_field *f2`)
- void `cob_inspect_first` (`cob_field *f1`, `cob_field *f2`)
- void `cob_inspect_trailing` (`cob_field *f1`, `cob_field *f2`)
- void `cob_inspect_converting` (`cob_field *f1`, `cob_field *f2`)
- void `cob_inspect_finish` (void)
- void `cob_string_init` (`cob_field *dst`, `cob_field *ptr`)
- void `cob_string_delimited` (`cob_field *dlim`)
- void `cob_string_append` (`cob_field *src`)
- void `cob_string_finish` (void)
- void `cob_unstring_init` (`cob_field *src`, `cob_field *ptr`, const size\_t num\_dlm)
- void `cob_unstring_delimited` (`cob_field *dlim`, const int all)
- void `cob_unstring_into` (`cob_field *dst`, `cob_field *dlim`, `cob_field *cnt`)
- void `cob_unstring_tallying` (`cob_field *f`)
- void `cob_unstring_finish` (void)
- void `cob_init_strings` (void)

### 7.44.1 Define Documentation

#### 7.44.1.1 #define DLM\_DEFAULT\_NUM 8

Definition at line 39 of file strings.c.

#### 7.44.1.2 #define INSPECT\_ALL 0

Definition at line 34 of file strings.c.

#### 7.44.1.3 #define INSPECT\_FIRST 2

Definition at line 36 of file strings.c.

#### 7.44.1.4 #define INSPECT\_LEADING 1

Definition at line 35 of file strings.c.

#### 7.44.1.5 #define INSPECT\_TRAILING 3

Definition at line 37 of file strings.c.

## 7.44.2 Function Documentation

### 7.44.2.1 void cob\_init\_strings ( void )

Definition at line 595 of file strings.c.

```
{
    inspect_mark = cob_malloc (COB_MEDIUM_BUFF);
    lastsize = COB_MEDIUM_BUFF;
    alpha_attr.type = COB_TYPE_ALPHANUMERIC;
    alpha_attr.digits = 0;
    alpha_attr.scale = 0;
    alpha_attr.flags = 0;
    alpha_attr.pic = NULL;
    alpha_fld.size = 0;
    alpha_fld.data = NULL;
    alpha_fld.attr = &alpha_attr;
}
```

### 7.44.2.2 void cob\_inspect\_after ( const cob\_field \* str )

Definition at line 251 of file strings.c.

```
{
    unsigned char *p;

    for (p = inspect_start; p < inspect_end - str->size + 1; p++) {
        if (memcmp (p, str->data, str->size) == 0) {
            inspect_start = p + str->size;
            return;
        }
    }
    inspect_start = inspect_end;
}
```

### 7.44.2.3 void cob\_inspect\_all ( cob\_field \* f1, cob\_field \* f2 )

Definition at line 297 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_ALL);
}
```

### 7.44.2.4 void cob\_inspect\_before ( const cob\_field \* str )

Definition at line 238 of file strings.c.

```
{
    unsigned char *p;
```



```

    for (p = inspect_start; p < inspect_end - str->size + 1; p++) {
        if (memcmp (p, str->data, str->size) == 0) {
            inspect_end = p;
            return;
        }
    }
}

```

#### 7.44.2.5 void cob\_inspect\_characters ( cob\_field \* f1 )

Definition at line 265 of file strings.c.

```

{
    int    *mark;
    int    i;
    int    n;
    int    len;

    mark = &inspect_mark[inspect_start - inspect_data];
    len = (int)(inspect_end - inspect_start);
    if (inspect_replacing) {
        /* INSPECT REPLACING CHARACTERS f1 */
        for (i = 0; i < len; i++) {
            if (mark[i] == -1) {
                mark[i] = f1->data[0];
            }
        }
    } else {
        /* INSPECT TALLYING f1 CHARACTERS */
        n = 0;
        for (i = 0; i < len; i++) {
            if (mark[i] == -1) {
                mark[i] = 1;
                n++;
            }
        }
        if (n > 0) {
            cob_add_int (f1, n);
        }
    }
}

```

#### 7.44.2.6 void cob\_inspect\_converting ( cob\_field \* f1, cob\_field \* f2 )

Definition at line 321 of file strings.c.

```

{
    size_t  i;
    size_t  j;
    size_t  len;

    len = (size_t)(inspect_end - inspect_start);
    for (j = 0; j < f1->size; j++) {
        for (i = 0; i < len; i++) {
            if (inspect_mark[i] == -1 && inspect_start[i] == f1->

```

```

        data[j]) {
                                inspect_start[i] = f2->data[j];
                                inspect_mark[i] = 1;
                                }
        }
}

```

#### 7.44.2.7 void cob\_inspect\_finish ( void )

Definition at line 339 of file strings.c.

```

{
    size_t i;

    if (inspect_replacing) {
        for (i = 0; i < inspect_size; i++) {
            if (inspect_mark[i] != -1) {
                inspect_data[i] = inspect_mark[i];
            }
        }
    }

    cob_put_sign (inspect_var, inspect_sign);
}

```

#### 7.44.2.8 void cob\_inspect\_first ( cob\_field \* f1, cob\_field \* f2 )

Definition at line 309 of file strings.c.

```

{
    inspect_common (f1, f2, INSPECT_FIRST);
}

```

#### 7.44.2.9 void cob\_inspect\_init ( cob\_field \* var, const int replacing )

Definition at line 205 of file strings.c.

```

{
    size_t i;
    size_t digcount;

    inspect_var_copy = *var;
    inspect_var = &inspect_var_copy;
    inspect_replacing = replacing;
    inspect_sign = cob_get_sign (var);
    inspect_size = COB_FIELD_SIZE (var);
    inspect_data = COB_FIELD_DATA (var);
    inspect_start = NULL;
    inspect_end = NULL;
    digcount = inspect_size * sizeof (int);
    if (digcount > lastsize) {

```

```
        free (inspect_mark);
        inspect_mark = cob_malloc (digcount);
        lastsize = digcount;
    }
    for (i = 0; i < inspect_size; i++) {
        inspect_mark[i] = -1;
    }
    cob_exception_code = 0;
}
```

#### 7.44.2.10 void cob\_inspect\_leading ( cob\_field \* f1, cob\_field \* f2 )

Definition at line 303 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_LEADING);
}
```

#### 7.44.2.11 void cob\_inspect\_start ( void )

Definition at line 231 of file strings.c.

```
{
    inspect_start = inspect_data;
    inspect_end = inspect_data + inspect_size;
}
```

#### 7.44.2.12 void cob\_inspect\_trailing ( cob\_field \* f1, cob\_field \* f2 )

Definition at line 315 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_TRAILING);
}
```

#### 7.44.2.13 void cob\_string\_append ( cob\_field \* src )

Definition at line 390 of file strings.c.

```
{
    size_t  src_size;
    int     i;
    int     size;

    if (cob_exception_code) {
        return;
    }
}
```

```

src_size = src->size;
if (string_dlm) {
    size = (int)(src_size - string_dlm->size + 1);
    for (i = 0; i < size; i++) {
        if (memcmp (src->data + i, string_dlm->data, string_dlm->
size) == 0) {
            src_size = i;
            break;
        }
    }
}

if (src_size <= string_dst->size - string_offset) {
    memcpy (string_dst->data + string_offset, src->data, src_size);
    string_offset += (int) src_size;
} else {
    size = (int)(string_dst->size - string_offset);
    memcpy (string_dst->data + string_offset, src->data, (size_t)size
);
    string_offset += size;
    cob_set_exception (COB_EC_OVERFLOW_STRING);
}
}

```

#### 7.44.2.14 void cob\_string\_delimited ( cob\_field \* dlm )

Definition at line 380 of file strings.c.

```

{
    string_dlm = NULL;
    if (dlm) {
        string_dlm_copy = *dlm;
        string_dlm = &string_dlm_copy;
    }
}

```

#### 7.44.2.15 void cob\_string\_finish ( void )

Definition at line 423 of file strings.c.

```

{
    if (string_ptr) {
        cob_set_int (string_ptr, string_offset + 1);
    }
}

```

#### 7.44.2.16 void cob\_string\_init ( cob\_field \* dst, cob\_field \* ptr )

Definition at line 359 of file strings.c.

```

{
    string_dst_copy = *dst;
}

```

```

string_dst = &string_dst_copy;
string_ptr = NULL;
if (ptr) {
    string_ptr_copy = *ptr;
    string_ptr = &string_ptr_copy;
}
string_offset = 0;
cob_exception_code = 0;

if (string_ptr) {
    string_offset = cob_get_int (string_ptr) - 1;
    if (string_offset < 0 || string_offset >= (int)string_dst->size)
    {
        cob_set_exception (COB_EC_OVERFLOW_STRING);
    }
}
}

```

#### 7.44.2.17 void cob\_unstring\_delimited ( cob\_field \* dlm, const int all )

Definition at line 476 of file strings.c.

```

{
    dlm_list[unstring_ndlms].uns_dlm = dlm;
    dlm_list[unstring_ndlms].uns_all = all;
    unstring_ndlms++;
}

```

#### 7.44.2.18 void cob\_unstring\_finish ( void )

Definition at line 581 of file strings.c.

```

{
    if (unstring_offset < (int)unstring_src->size) {
        cob_set_exception (COB_EC_OVERFLOW_UNSTRING);
    }

    if (unstring_ptr) {
        cob_set_int (unstring_ptr, unstring_offset + 1);
    }
}

```

#### 7.44.2.19 void cob\_unstring\_init ( cob\_field \* src, cob\_field \* ptr, const size\_t num\_dlm )

Definition at line 435 of file strings.c.

```

{
    static size_t    udmlcount = 0;

    unstring_src_copy = *src;
    unstring_src = &unstring_src_copy;
    unstring_ptr = NULL;
}

```

```

if (ptr) {
    unstring_ptr_copy = *ptr;
    unstring_ptr = &unstring_ptr_copy;
}

unstring_offset = 0;
unstring_count = 0;
unstring_ndlms = 0;
cob_exception_code = 0;
if (!dml_list) {
    if (num_dlm <= DLM_DEFAULT_NUM) {
        dml_list = cob_malloc (DLM_DEFAULT_NUM * sizeof(struct
dml_struct));
        udlmcount = DLM_DEFAULT_NUM;
    } else {
        dml_list = cob_malloc (num_dlm * sizeof(struct
dml_struct));
        udlmcount = num_dlm;
    }
} else {
    if (num_dlm > udlmcount) {
        free (dml_list);
        dml_list = cob_malloc (num_dlm * sizeof(struct
dml_struct));
        udlmcount = num_dlm;
    }
}

if (unstring_ptr) {
    unstring_offset = cob_get_int (unstring_ptr) - 1;
    if (unstring_offset < 0 || unstring_offset >= (int)unstring_src->
size) {
        cob_set_exception (COB_EC_OVERFLOW_UNSTRING);
    }
}
}

```

#### 7.44.2.20 void cob\_unstring\_into ( cob\_field \* dst, cob\_field \* dlm, cob\_field \* cnt )

Definition at line 484 of file strings.c.

```

{
    unsigned char *p;
    unsigned char *dp;
    unsigned char *s;
    unsigned char *dml_data;
    unsigned char *start;
    size_t dml_size = 0;
    int i;
    int srsz;
    int dlsz;
    int match_size = 0;
    int brkpt = 0;

    if (cob_exception_code) {
        return;
    }

    if (unstring_offset >= (int)unstring_src->size) {

```

```

        return;
    }

    start = unstring_src->data + unstring_offset;
    dlm_data = NULL;
    if (unstring_ndlms == 0) {
        match_size = cob_min_int ((int)COB_FIELD_SIZE (dst),
                                  (int)unstring_src->size - unstring_offs
et);
        cob_memcpy (dst, start, match_size);
        unstring_offset += match_size;
    } else {
        srsz = (int) unstring_src->size;
        s = unstring_src->data + srsz;
        for (p = start; p < s; p++) {
            for (i = 0; i < unstring_ndlms; i++) {
                dlsz = (int) dlm_list[i].uns_dlm->size;
                dp = dlm_list[i].uns_dlm->data;
                if (p + dlsz > s) {
                    break;
                }
                if (!memcmp (p, dp, (size_t)dlsz)) {
                    match_size = (int)(p - start);
                    cob_memcpy (dst, start, match_size);
                    unstring_offset += match_size + dlsz;
                    dlm_data = dp;
                    dlm_size = dlsz;
                    if (dlm_list[i].uns_all) {
                        for (p++ ; p < s; p++) {
                            if (p + dlsz > s) {
                                break;
                            }
                            if (memcmp (p, dp, (size_
t)dlsz)) {
                                break;
                            }
                            unstring_offset += dlsz
;
                        }
                    }
                    brkpt = 1;
                    break;
                }
            }
        }
        if (brkpt) {
            break;
        }
    }
    if (!brkpt) {
        /* no match */
        match_size = (int)(unstring_src->size - unstring_offset);

        cob_memcpy (dst, start, match_size);
        unstring_offset = (int) unstring_src->size;
        dlm_data = NULL;
    }
}
unstring_count++;

if (dlm) {
    if (dlm_data) {

```

```
        cob_memcpy (d1m, d1m_data, (int) d1m_size);
    } else if (COB_FIELD_IS_NUMERIC (d1m)) {
        cob_move (&cob_zero, d1m);
    } else {
        cob_move (&cob_space, d1m);
    }
}

if (cnt) {
    cob_set_int (cnt, match_size);
}
}
```

#### 7.44.2.21 void cob\_unstring\_tallying ( cob\_field \* f )

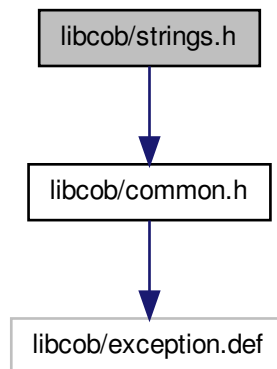
Definition at line 575 of file strings.c.

```
{
    cob_add_int (f, unstring_count);
}
```

## 7.45 libcob/strings.h File Reference

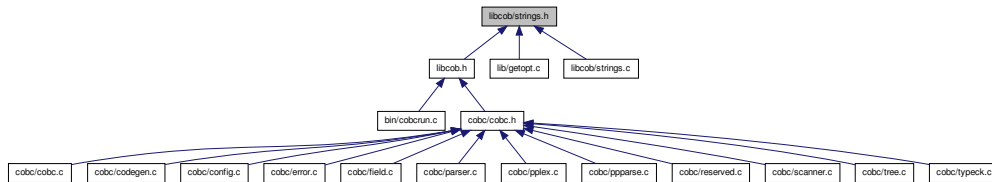
```
#include <libcob/common.h>
```

Include dependency graph for strings.h:





This graph shows which files directly or indirectly include this file:



## Functions

- void [cob\\_inspect\\_init](#) (cob\_field \*, const int)
- void [cob\\_inspect\\_start](#) (void)
- void [cob\\_inspect\\_before](#) (const cob\_field \*)
- void [cob\\_inspect\\_after](#) (const cob\_field \*)
- void [cob\\_inspect\\_characters](#) (cob\_field \*)
- void [cob\\_inspect\\_all](#) (cob\_field \*, cob\_field \*)
- void [cob\\_inspect\\_leading](#) (cob\_field \*, cob\_field \*)
- void [cob\\_inspect\\_first](#) (cob\_field \*, cob\_field \*)
- void [cob\\_inspect\\_trailing](#) (cob\_field \*, cob\_field \*)
- void [cob\\_inspect\\_converting](#) (cob\_field \*, cob\_field \*)
- void [cob\\_inspect\\_finish](#) (void)
- void [cob\\_string\\_init](#) (cob\_field \*, cob\_field \*)
- void [cob\\_string\\_delimited](#) (cob\_field \*)
- void [cob\\_string\\_append](#) (cob\_field \*)
- void [cob\\_string\\_finish](#) (void)
- void [cob\\_unstring\\_init](#) (cob\_field \*, cob\_field \*, const size\_t)
- void [cob\\_unstring\\_delimited](#) (cob\_field \*, const int)
- void [cob\\_unstring\\_into](#) (cob\_field \*, cob\_field \*, cob\_field \*)
- void [cob\\_unstring\\_tallying](#) (cob\_field \*)
- void [cob\\_unstring\\_finish](#) (void)

### 7.45.1 Function Documentation

#### 7.45.1.1 void cob.inspect\_after ( const cob\_field \* )

Definition at line 251 of file strings.c.

```

{
    unsigned char    *p;

    for (p = inspect_start; p < inspect_end - str->size + 1; p++) {
        if (memcmp (p, str->data, str->size) == 0) {
            inspect_start = p + str->size;
            return;
        }
    }
}

```

```

        }
    }
    inspect_start = inspect_end;
}

```

#### 7.45.1.2 void cob\_inspect\_all ( cob\_field \*, cob\_field \* )

Definition at line 297 of file strings.c.

```

{
    inspect_common (f1, f2, INSPECT_ALL);
}

```

#### 7.45.1.3 void cob\_inspect\_before ( const cob\_field \* )

Definition at line 238 of file strings.c.

```

{
    unsigned char *p;

    for (p = inspect_start; p < inspect_end - str->size + 1; p++) {
        if (memcmp (p, str->data, str->size) == 0) {
            inspect_end = p;
            return;
        }
    }
}

```

#### 7.45.1.4 void cob\_inspect\_characters ( cob\_field \* )

Definition at line 265 of file strings.c.

```

{
    int *mark;
    int i;
    int n;
    int len;

    mark = &inspect_mark[inspect_start - inspect_data];
    len = (int)(inspect_end - inspect_start);
    if (inspect_replacing) {
        /* INSPECT REPLACING CHARACTERS f1 */
        for (i = 0; i < len; i++) {
            if (mark[i] == -1) {
                mark[i] = f1->data[0];
            }
        }
    } else {
        /* INSPECT TALLYING f1 CHARACTERS */
        n = 0;
        for (i = 0; i < len; i++) {
            if (mark[i] == -1) {

```

```

                mark[i] = 1;
                n++;
            }
        }
        if (n > 0) {
            cob_add_int (f1, n);
        }
    }
}

```

#### 7.45.1.5 void cob.inspect\_converting ( cob\_field \*, cob\_field \* )

Definition at line 321 of file strings.c.

```

{
    size_t i;
    size_t j;
    size_t len;

    len = (size_t)(inspect_end - inspect_start);
    for (j = 0; j < f1->size; j++) {
        for (i = 0; i < len; i++) {
            if (inspect_mark[i] == -1 && inspect_start[i] == f1->data
[j]) {
                inspect_start[i] = f2->data[j];
                inspect_mark[i] = 1;
            }
        }
    }
}

```

#### 7.45.1.6 void cob.inspect\_finish ( void )

Definition at line 339 of file strings.c.

```

{
    size_t i;

    if (inspect_replacing) {
        for (i = 0; i < inspect_size; i++) {
            if (inspect_mark[i] != -1) {
                inspect_data[i] = inspect_mark[i];
            }
        }
    }

    cob_put_sign (inspect_var, inspect_sign);
}

```

#### 7.45.1.7 void cob.inspect\_first ( cob\_field \*, cob\_field \* )

Definition at line 309 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_FIRST);
}
```

#### 7.45.1.8 void cob\_inspect\_init ( cob\_field \*, const int )

Definition at line 205 of file strings.c.

```
{
    size_t          i;
    size_t          digcount;

    inspect_var_copy = *var;
    inspect_var = &inspect_var_copy;
    inspect_replacing = replacing;
    inspect_sign = cob_get_sign (var);
    inspect_size = COB_FIELD_SIZE (var);
    inspect_data = COB_FIELD_DATA (var);
    inspect_start = NULL;
    inspect_end = NULL;
    digcount = inspect_size * sizeof (int);
    if (digcount > lastsize) {
        free (inspect_mark);
        inspect_mark = cob_malloc (digcount);
        lastsize = digcount;
    }
    for (i = 0; i < inspect_size; i++) {
        inspect_mark[i] = -1;
    }
    cob_exception_code = 0;
}
```

#### 7.45.1.9 void cob\_inspect\_leading ( cob\_field \*, cob\_field \* )

Definition at line 303 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_LEADING);
}
```

#### 7.45.1.10 void cob\_inspect\_start ( void )

Definition at line 231 of file strings.c.

```
{
    inspect_start = inspect_data;
    inspect_end = inspect_data + inspect_size;
}
```

**7.45.1.11 void cob\_inspect\_trailing ( cob\_field \*, cob\_field \* )**

Definition at line 315 of file strings.c.

```
{
    inspect_common (f1, f2, INSPECT_TRAILING);
}
```

**7.45.1.12 void cob\_string\_append ( cob\_field \* )**

Definition at line 390 of file strings.c.

```
{
    size_t src_size;
    int i;
    int size;

    if (cob_exception_code) {
        return;
    }

    src_size = src->size;
    if (string_dlm) {
        size = (int)(src_size - string_dlm->size + 1);
        for (i = 0; i < size; i++) {
            if (memcmp (src->data + i, string_dlm->data, string_dlm->
size) == 0) {
                src_size = i;
                break;
            }
        }
    }

    if (src_size <= string_dst->size - string_offset) {
        memcpy (string_dst->data + string_offset, src->data, src_size);
        string_offset += (int) src_size;
    } else {
        size = (int)(string_dst->size - string_offset);
        memcpy (string_dst->data + string_offset, src->data, (size_t)size
);
        string_offset += size;
        cob_set_exception (COB_EC_OVERFLOW_STRING);
    }
}
```

**7.45.1.13 void cob\_string\_delimited ( cob\_field \* )**

Definition at line 380 of file strings.c.

```
{
    string_dlm = NULL;
    if (dlm) {
        string_dlm_copy = *dlm;
        string_dlm = &string_dlm_copy;
    }
}
```

**7.45.1.14 void cob\_string\_finish ( void )**

Definition at line 423 of file strings.c.

```

{
    if (string_ptr) {
        cob_set_int (string_ptr, string_offset + 1);
    }
}

```

**7.45.1.15 void cob\_string\_init ( cob\_field \*, cob\_field \* )**

Definition at line 359 of file strings.c.

```

{
    string_dst_copy = *dst;
    string_dst = &string_dst_copy;
    string_ptr = NULL;
    if (ptr) {
        string_ptr_copy = *ptr;
        string_ptr = &string_ptr_copy;
    }
    string_offset = 0;
    cob_exception_code = 0;

    if (string_ptr) {
        string_offset = cob_get_int (string_ptr) - 1;
        if (string_offset < 0 || string_offset >= (int)string_dst->size)
        {
            cob_set_exception (COB_EC_OVERFLOW_STRING);
        }
    }
}

```

**7.45.1.16 void cob\_unstring\_delimited ( cob\_field \*, const int )**

Definition at line 476 of file strings.c.

```

{
    dlm_list[unstring_ndlms].uns_dlm = dlm;
    dlm_list[unstring_ndlms].uns_all = all;
    unstring_ndlms++;
}

```

**7.45.1.17 void cob\_unstring\_finish ( void )**

Definition at line 581 of file strings.c.

```

{
    if (unstring_offset < (int)unstring_src->size) {

```

```

        cob_set_exception (COB_EC_OVERFLOW_UNSTRING);
    }

    if (unstring_ptr) {
        cob_set_int (unstring_ptr, unstring_offset + 1);
    }
}

```

#### 7.45.1.18 void cob\_unstring\_init( cob\_field \*, cob\_field \*, const size\_t )

Definition at line 435 of file strings.c.

```

{
    static size_t  udlmcount = 0;

    unstring_src_copy = *src;
    unstring_src = &unstring_src_copy;
    unstring_ptr = NULL;
    if (ptr) {
        unstring_ptr_copy = *ptr;
        unstring_ptr = &unstring_ptr_copy;
    }

    unstring_offset = 0;
    unstring_count = 0;
    unstring_ndlms = 0;
    cob_exception_code = 0;
    if (!dml_list) {
        if (num_dlm <= DLM_DEFAULT_NUM) {
            dml_list = cob_malloc (DLM_DEFAULT_NUM * sizeof(struct
dml_struct));
            udlmcount = DLM_DEFAULT_NUM;
        } else {
            dml_list = cob_malloc (num_dlm * sizeof(struct
dml_struct));
            udlmcount = num_dlm;
        }
    } else {
        if (num_dlm > udlmcount) {
            free (dml_list);
            dml_list = cob_malloc (num_dlm * sizeof(struct
dml_struct));
            udlmcount = num_dlm;
        }
    }

    if (unstring_ptr) {
        unstring_offset = cob_get_int (unstring_ptr) - 1;
        if (unstring_offset < 0 || unstring_offset >= (int)unstring_src->
size) {
            cob_set_exception (COB_EC_OVERFLOW_UNSTRING);
        }
    }
}

```

## 7.45.1.19 void cob\_unstring\_into ( cob\_field \*, cob\_field \*, cob\_field \* )

Definition at line 484 of file strings.c.

```

{
    unsigned char *p;
    unsigned char *dp;
    unsigned char *s;
    unsigned char *dml_data;
    unsigned char *start;
    size_t dlm_size = 0;
    int i;
    int srsize;
    int dlsize;
    int match_size = 0;
    int brkpt = 0;

    if (cob_exception_code) {
        return;
    }

    if (unstring_offset >= (int)unstring_src->size) {
        return;
    }

    start = unstring_src->data + unstring_offset;
    dlm_data = NULL;
    if (unstring_ndlms == 0) {
        match_size = cob_min_int ((int)COB_FIELD_SIZE (dst),
                                   (int)unstring_src->size - unstring_offs
et);
        cob_memcpy (dst, start, match_size);
        unstring_offset += match_size;
    } else {
        srsize = (int) unstring_src->size;
        s = unstring_src->data + srsize;
        for (p = start; p < s; p++) {
            for (i = 0; i < unstring_ndlms; i++) {
                dlsize = (int) dlm_list[i].uns_dlm->size;
                dp = dlm_list[i].uns_dlm->data;
                if (p + dlsize > s) {
                    break;
                }
                if (!memcmp (p, dp, (size_t)dlsize)) {
                    match_size = (int)(p - start);
                    cob_memcpy (dst, start, match_size);
                    unstring_offset += match_size + dlsize;
                    dlm_data = dp;
                    dlm_size = dlsize;
                    if (dlm_list[i].uns_all) {
                        for (p++ ; p < s; p++) {
                            if (p + dlsize > s) {
                                break;
                            }
                        }
                        if (memcmp (p, dp, (size_
t)dlsize)) {
                            break;
                        }
                    }
                    unstring_offset += dlsize
;
                }
            }
        }
    }
}

```



```

        }
        brkpt = 1;
        break;
    }
    }
    if (brkpt) {
        break;
    }
}
if (!brkpt) {
    /* no match */
    match_size = (int)(unstring_src->size - unstring_offset);

    cob_memcpy (dst, start, match_size);
    unstring_offset = (int) unstring_src->size;
    dlm_data = NULL;
}
}
unstring_count++;

if (dlm) {
    if (dlm_data) {
        cob_memcpy (dlm, dlm_data, (int) dlm_size);
    } else if (COB_FIELD_IS_NUMERIC (dlm)) {
        cob_move (&cob_zero, dlm);
    } else {
        cob_move (&cob_space, dlm);
    }
}

if (cnt) {
    cob_set_int (cnt, match_size);
}
}

```

#### 7.45.1.20 void cob\_unstring\_tallying ( cob\_field \* )

Definition at line 575 of file strings.c.

```

{
    cob_add_int (f, unstring_count);
}

```

## 7.46 libcob/termio.c File Reference

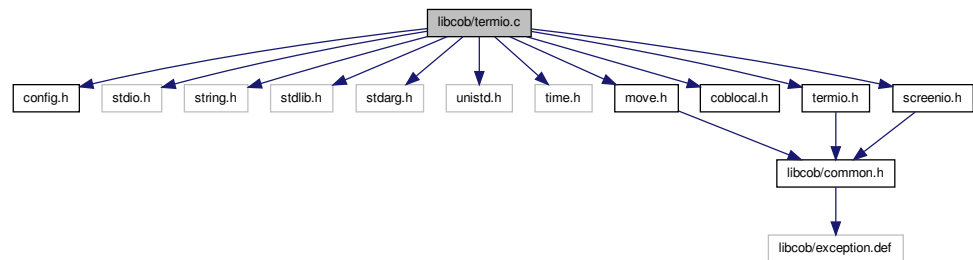
```

#include "config.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <unistd.h>

```

```
#include <time.h>
#include "move.h"
#include "coblocal.h"
#include "termio.h"
#include "screenio.h"
```

Include dependency graph for termio.c:



## Functions

- void [cob\\_display](#) (const int outorr, const int newline, const int varcnt,...)
- void [cob\\_accept](#) (cob\_field \*f)
- void [cob\\_init\\_termio](#) (void)

### 7.46.1 Function Documentation

#### 7.46.1.1 void cob\_accept ( cob\_field \* f )

Definition at line 234 of file termio.c.

```
{
/* RXW
size_t          size;
*/
cob_field_attr attr;
cob_field      temp;

if (cob_screen_initialized) {
cob_field_accept (E, NULL, NULL, NULL, NULL, NULL, 0);
return;
}
temp.data = term_buff;
temp.attr = &attr;
COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
/* read a line */
if (fgets ((char *)term_buff, COB_MEDIUM_BUFFER, stdin) == NULL) {
```

```

        temp.size = 1;
        term_buff[0] = ' ';
        term_buff[1] = 0;
    } else {
        temp.size = strlen ((char *)term_buff) - 1;
    }
    if (COB_FIELD_TYPE(f) == COB_TYPE_NUMERIC_DISPLAY) {
        if (temp.size > f->size) {
            temp.size = f->size;
        }
    }
    cob_move (&temp, f);
/* RXW
    if (isatty (fileno (stdin))) {
        temp.size = strlen ((char *)term_buff) - 1;
        cob_move (&temp, f);
    } else {
        size = strlen ((char *)term_buff) - 1;
        if (size > f->size) {
            size = f->size;
        }
        memcpy (f->data, term_buff, size);
        memset (f->data + size, ' ', f->size - size);
    }
*/
}

```

#### 7.46.1.2 void cob\_display ( const int outorerr, const int newline, const int varcnt, ... )

Definition at line 205 of file termio.c.

```

{
    FILE          *fp;
    cob_field     *f;
    int           i;
    va_list       args;

    if (!outorerr && !cob_screen_initialized) {
        fp = stdout;
    } else {
        fp = stderr;
    }
    va_start (args, varcnt);
    for (i = 0; i < varcnt; ++i) {
        f = va_arg (args, cob_field *);
        display (f, fp);
    }
    va_end (args);
    if (newline) {
        putc ('\n', fp);
        fflush (fp);
    }
}

```

#### 7.46.1.3 void cob\_init\_termio ( void )

Definition at line 279 of file termio.c.

```

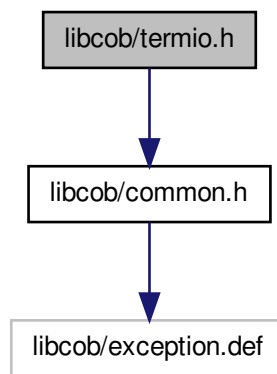
{
    term_buff = cob_malloc (COB_MEDIUM_BUFF);
}

```

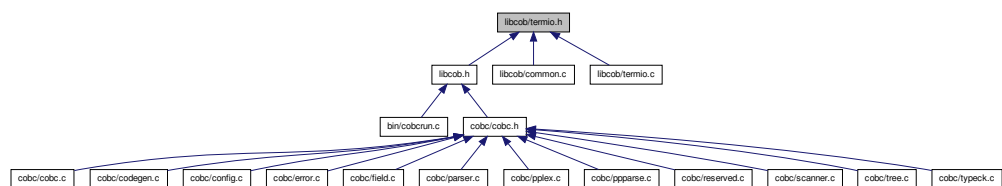
## 7.47 libcob/termio.h File Reference

```
#include <libcob/common.h>
```

Include dependency graph for termio.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [cob\\_display](#) (const int, const int, const int,...)
- void [cob\\_accept](#) (cob\_field \*)

### 7.47.1 Function Documentation

#### 7.47.1.1 void cob\_accept ( cob\_field \* )

Definition at line 234 of file termio.c.

```

{
/* RXW
   size_t          size;
*/
   cob_field_attr  attr;
   cob_field       temp;

   if (cob_screen_initialized) {
       cob_field_accept (f, NULL, NULL, NULL, NULL, NULL, 0);
       return;
   }
   temp.data = term_buff;
   temp.attr = &attr;
   COB_ATTR_INIT (COB_TYPE_ALPHANUMERIC, 0, 0, 0, NULL);
   /* read a line */
   if (fgets ((char *)term_buff, COB_MEDIUM_BUFF, stdin) == NULL) {
       temp.size = 1;
       term_buff[0] = ' ';
       term_buff[1] = 0;
   } else {
       temp.size = strlen ((char *)term_buff) - 1;
   }
   if (COB_FIELD_TYPE(f) == COB_TYPE_NUMERIC_DISPLAY) {
       if (temp.size > f->size) {
           temp.size = f->size;
       }
   }
   cob_move (&temp, f);
/* RXW
   if (isatty (fileno (stdin))) {
       temp.size = strlen ((char *)term_buff) - 1;
       cob_move (&temp, f);
   } else {
       size = strlen ((char *)term_buff) - 1;
       if (size > f->size) {
           size = f->size;
       }
       memcpy (f->data, term_buff, size);
       memset (f->data + size, ' ', f->size - size);
   }
*/
}

```

#### 7.47.1.2 void cob\_display ( const int , const int , const int , ... )

Definition at line 205 of file termio.c.

```

{
   FILE          *fp;
   cob_field     *f;
   int           i;

```

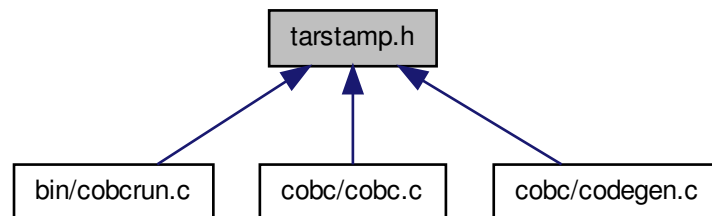
```
va_list      args;

if (!outorrerr && !cob_screen_initialized) {
    fp = stdout;
} else {
    fp = stderr;
}
va_start (args, varcnt);
for (i = 0; i < varcnt; ++i) {
    f = va_arg (args, cob_field *);
    display (f, fp);
}
va_end (args);
if (newline) {
    putc ('\n', fp);
    fflush (fp);
}
}
```

## 7.48 mainpage.h File Reference

## 7.49 tarstamp.h File Reference

This graph shows which files directly or indirectly include this file:



# Index

- - getopt.c, [811](#)
  - gettext.h, [830](#)
- \_FILE\_OFFSET\_BITS
  - fileio.c, [895](#)
- \_GETOPT\_H
  - getopt.h, [825](#)
- \_LARGEFILE64\_SOURCE
  - fileio.c, [895](#)
- \_LFS64\_LARGEFILE
  - fileio.c, [895](#)
- \_LFS64\_STDIO
  - fileio.c, [895](#)
- \_NO\_PROTO
  - getopt.c, [811](#)
- \_XOPEN\_SOURCE\_EXTENDED
  - screenio.c, [1105](#)
- \_\_USE\_GNU
  - call.c, [836](#)
- \_\_USE\_STRING\_INLINES
  - config.h, [798](#)
- \_\_cob\_screen, [13](#)
  - attr, [14](#)
  - backg, [14](#)
  - child, [14](#)
  - column, [14](#)
  - field, [14](#)
  - foreg, [14](#)
  - line, [14](#)
  - next, [15](#)
  - occurs, [15](#)
  - type, [15](#)
  - value, [15](#)
- \_\_getopt\_internal
  - getopt.c, [812](#)
  - getopt.h, [825](#)
- ABORT
  - cobc.h, [157](#)
- ACCEPT
  - parser.c, [224](#), [276](#), [288](#)
- parser.h, [326](#), [370](#), [382](#)
- ppparse.c, [433](#), [444](#)
- ppparse.h, [468](#), [479](#)
- ACCESS
  - parser.c, [224](#), [276](#), [288](#)
  - parser.h, [326](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- access\_mode
  - cb\_file, [41](#)
  - cob\_file, [99](#)
- ADD
  - parser.c, [224](#), [276](#), [288](#)
  - parser.h, [326](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- ADDRESS
  - parser.c, [224](#), [276](#), [288](#)
  - parser.h, [326](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- ADVANCING
  - parser.c, [224](#), [276](#), [288](#)
  - parser.h, [326](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- AFTER
  - parser.c, [224](#), [276](#), [288](#)
  - parser.h, [327](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- ALL
  - parser.c, [224](#), [277](#), [288](#)
  - parser.h, [327](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- all
  - cb\_literal, [62](#)
  - cb\_reference, [80](#)
- ALLOCATE
  - parser.c, [224](#), [277](#), [288](#)

- parser.h, [327](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- ALPHABET
  - parser.c, [224](#), [277](#), [288](#)
  - parser.h, [327](#), [370](#), [382](#)
  - ppparse.c, [433](#), [444](#)
  - ppparse.h, [468](#), [479](#)
- alphabet\_name\_list
  - cb\_program, [73](#)
- ALPHABETIC
  - parser.c, [224](#), [277](#), [288](#)
  - parser.h, [327](#), [370](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [479](#)
- ALPHABETIC\_LOWER
  - parser.c, [277](#), [288](#)
  - parser.h, [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [479](#)
- ALPHABETIC\_UPPER
  - parser.c, [277](#), [288](#)
  - parser.h, [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [479](#)
- ALPHABETIC\_LOWER
  - parser.c, [225](#)
  - parser.h, [327](#)
- ALPHABETIC\_UPPER
  - parser.c, [225](#)
  - parser.h, [327](#)
- ALPHANUMERIC
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [327](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [479](#)
- ALPHANUMERIC\_EDITED
  - parser.c, [277](#), [288](#)
  - parser.h, [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ALPHANUMERIC\_EDITED
  - parser.c, [225](#)
  - parser.h, [327](#)
- ALSO
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [327](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- alt\_ebcdic
  - cobc.c, [151](#)
  - cobc.h, [167](#)
- alt\_key\_list
  - cb\_file, [41](#)
- ALTER
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ALTERNATE
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ambiguous\_error
  - error.c, [192](#)
  - tree.h, [589](#)
- AND
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ANY
  - config.c, [186](#)
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ARE
  - parser.c, [225](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- AREA
  - parser.c, [226](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- argc
  - cb\_funcall, [45](#)
- args
  - cb\_call, [23](#)
  - cb\_intrinsic, [52](#)
  - cb\_intrinsic\_table, [53](#)
- ARGUMENT\_NUMBER
  - parser.c, [277](#), [288](#)
  - parser.h, [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ARGUMENT\_VALUE



- parser.c, [277](#), [288](#)
- parser.h, [371](#), [382](#)
- ppparse.c, [433](#), [445](#)
- ppparse.h, [468](#), [480](#)
- ARGUMENT\_NUMBER
  - parser.c, [226](#)
  - parser.h, [328](#)
- ARGUMENT\_VALUE
  - parser.c, [226](#)
  - parser.h, [328](#)
- argv
  - cb\_funcall, [45](#)
- AS
  - parser.c, [226](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ASCENDING
  - parser.c, [226](#), [277](#), [288](#)
  - parser.h, [328](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- ASSIGN
  - parser.c, [226](#), [277](#), [288](#)
  - parser.h, [329](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- assign
  - cb\_file, [41](#)
  - cob\_file, [99](#)
- AT
  - parser.c, [226](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [382](#)
  - ppparse.c, [433](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- attr
  - \_\_cob\_screen, [14](#)
  - cob\_field, [96](#)
- attribute\_hidden
  - getopt.c, [811](#), [823](#)
- AUTO
  - parser.c, [226](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [382](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- AUTOMATIC
  - parser.c, [226](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [382](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- backg
  - \_\_cob\_screen, [14](#)
- BACKGROUND\_COLOR
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [382](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- BACKGROUND\_COLOR
  - parser.c, [226](#)
  - parser.h, [329](#)
- BASED
  - parser.c, [227](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [468](#), [480](#)
- bdb\_file\_lock
  - indexed\_file, [123](#)
- bdb\_lock\_id
  - indexed\_file, [123](#)
- bdb\_record\_lock
  - indexed\_file, [123](#)
- BEFORE
  - parser.c, [227](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BEGIN
  - pplex.c, [404](#)
  - scanner.c, [499](#)
- BELL
  - parser.c, [227](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- bin/ Directory Reference, [9](#)
- bin/cobcrun.c, [139](#)
- BINARY
  - parser.c, [227](#), [277](#), [289](#)
  - parser.h, [329](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BINARY\_C\_LONG
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BINARY\_CHAR
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)

- ppparse.h, [469](#), [480](#)
- BINARY\_DOUBLE
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BINARY\_LONG
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BINARY\_SHORT
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BINARY\_C\_LONG
  - parser.c, [227](#)
  - parser.h, [329](#)
- BINARY\_CHAR
  - parser.c, [227](#)
  - parser.h, [330](#)
- BINARY\_DOUBLE
  - parser.c, [227](#)
  - parser.h, [330](#)
- BINARY\_LONG
  - parser.c, [227](#)
  - parser.h, [330](#)
- BINARY\_SHORT
  - parser.c, [227](#)
  - parser.h, [330](#)
- bind\_textdomain\_codeset
  - gettext.h, [830](#)
- bindtextdomain
  - gettext.h, [830](#)
- BLANK
  - parser.c, [227](#), [277](#), [289](#)
  - parser.h, [330](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BLANK\_LINE
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BLANK\_SCREEN
  - parser.c, [277](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BLANK\_LINE
  - parser.c, [228](#)
  - parser.h, [330](#)
- BLANK\_SCREEN
  - parser.c, [228](#)
  - parser.h, [330](#)
- BLINK
  - parser.c, [228](#), [277](#), [289](#)
  - parser.h, [330](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BLOCK
  - parser.c, [228](#), [277](#), [289](#)
  - parser.h, [330](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- block\_byte
  - cobitem, [111](#)
- body
  - cb\_perform, [65](#)
  - cb\_statement, [85](#)
- BOOLEAN
  - config.c, [186](#)
- BOTTOM
  - parser.c, [228](#), [278](#), [289](#)
  - parser.h, [330](#), [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- build\_file
  - tree.c, [512](#)
  - tree.h, [590](#)
- build\_literal
  - tree.c, [512](#)
  - tree.h, [590](#)
- BY
  - parser.c, [228](#), [278](#), [289](#), [300](#)
  - parser.h, [331](#), [371](#), [383](#), [393](#), [394](#)
  - ppparse.c, [424](#), [434](#), [445](#), [456](#)
  - ppparse.h, [466](#), [469](#), [480](#), [491](#)
- BYTE\_LENGTH
  - parser.c, [278](#), [289](#)
  - parser.h, [371](#), [383](#)
  - ppparse.c, [434](#), [445](#)
  - ppparse.h, [469](#), [480](#)
- BYTE\_LENGTH
  - parser.c, [228](#)
  - parser.h, [331](#)
- byteswap.h
  - COB\_BSWAP\_16, [833](#)
  - COB\_BSWAP\_16\_CONSTANT, [833](#)

- COB\_BSWAP\_32, 833
- COB\_BSWAP\_32\_CONSTANT, 833
- COB\_BSWAP\_64, 834
- COB\_BSWAP\_64\_CONSTANT, 834
- CALL
  - parser.c, 228, 278, 289
  - parser.h, 331, 371, 383
  - ppparse.c, 434, 446
  - ppparse.h, 469, 480
- call.c
  - \_\_USE\_GNU, 836
  - CALL\_BUFF\_SIZE, 836
  - CALL\_FILEBUFF\_SIZE, 836
  - cob\_call\_error, 837
  - cob\_call\_resolve, 837
  - cob\_call\_resolve\_1, 838
  - cob\_field\_cancel, 838
  - cob\_init\_call, 838
  - COB\_MAX\_COBCALL\_PARAMS, 836
  - cob\_resolve, 840
  - cob\_resolve\_1, 842
  - cob\_resolve\_error, 842
  - cob\_set\_cancel, 842
  - COB\_SYSTEM\_GEN, 836
  - cobcall, 843
  - cobcancel, 843
  - cobfunc, 844
  - coblongjmp, 844
  - cobsavenv, 845
  - cobsavenv2, 845
  - HASH\_SIZE, 836
  - lt\_dlclose, 836
  - lt\_dLError, 837
  - lt\_dlhandle, 837
  - lt\_dlopen, 837
  - lt\_dlsym, 837
  - PATHSEPC, 837
  - PATHSEPS, 837
- call.h
  - cob\_call\_error, 847
  - cob\_call\_resolve, 847
  - cob\_call\_resolve\_1, 847
  - cob\_field\_cancel, 848
  - cob\_resolve, 848
  - cob\_resolve\_1, 850
  - cob\_resolve\_error, 850
  - cob\_set\_cancel, 851
  - cobcall, 851
  - cobcancel, 852
- cobfunc, 852
- coblongjmp, 853
- cobsavenv, 853
- cobsavenv2, 853
- cobsetjmp, 847
- CALL\_BUFF\_SIZE
  - call.c, 836
- CALL\_FILEBUFF\_SIZE
  - call.c, 836
- call\_hash, 15
  - cancel, 16
  - flag\_is\_active, 16
  - func, 16
  - name, 16
  - next, 16
- CANCEL
  - parser.c, 228, 278, 289
  - parser.h, 331, 372, 383
  - ppparse.c, 434, 446
  - ppparse.h, 469, 480
- cancel
  - call\_hash, 16
- category
  - cb\_intrinsic\_table, 53
  - cb\_picture, 69
  - cb\_system\_name, 88
  - cb\_tree\_common, 90
  - reserved.c, 496
- CB\_ALPHABET\_CUSTOM
  - tree.h, 577
- CB\_ALPHABET\_EBCDIC
  - tree.h, 577
- CB\_ALPHABET\_NATIVE
  - tree.h, 577
- CB\_ALPHABET\_STANDARD\_1
  - tree.h, 577
- CB\_ALPHABET\_STANDARD\_2
  - tree.h, 577
- CB\_ARCHAIC
  - cobc.h, 160
- CB\_ASSIGN\_COBOL2002
  - cobc.h, 159
- CB\_ASSIGN\_IBM
  - cobc.h, 159
- CB\_ASSIGN\_MF
  - cobc.h, 159
- CB\_BINARY\_SIZE\_1\_2\_4\_8
  - cobc.h, 159
- CB\_BINARY\_SIZE\_1\_\_8
  - cobc.h, 159

- |  |   |
|--|---|
| CB_BINARY_SIZE_2_4_8<br>cobc.h, <a href="#">159</a>            | CB_CLASS_INDEX<br>tree.h, <a href="#">578</a>           |
| CB_BYTEORDER_BIG_ENDIAN<br>cobc.h, <a href="#">159</a>         | CB_CLASS_NATIONAL<br>tree.h, <a href="#">578</a>        |
| CB_BYTEORDER_NATIVE<br>cobc.h, <a href="#">159</a>             | CB_CLASS_NUMERIC<br>tree.h, <a href="#">578</a>         |
| CB_CALL_CONVENTION_NAME<br>tree.h, <a href="#">586</a>         | CB_CLASS_OBJECT<br>tree.h, <a href="#">578</a>          |
| CB_CAST_ADDR_OF_ADDR<br>tree.h, <a href="#">577</a>            | CB_CLASS_POINTER<br>tree.h, <a href="#">579</a>         |
| CB_CAST_ADDRESS<br>tree.h, <a href="#">577</a>                 | CB_CLASS_UNKNOWN<br>tree.h, <a href="#">578</a>         |
| CB_CAST_INTEGER<br>tree.h, <a href="#">577</a>                 | CB_CODE_NAME<br>tree.h, <a href="#">586</a>             |
| CB_CAST_LENGTH<br>tree.h, <a href="#">577</a>                  | CB_COMPUTER_NAME<br>tree.h, <a href="#">586</a>         |
| CB_CAST_PROGRAM_POINTER<br>tree.h, <a href="#">577</a>         | CB_DEVICE_CONSOLE<br>tree.h, <a href="#">579</a>        |
| CB_CATEGORY_ALPHABETIC<br>tree.h, <a href="#">578</a>          | CB_DEVICE_NAME<br>tree.h, <a href="#">586</a>           |
| CB_CATEGORY_ALPHANUMERIC<br>tree.h, <a href="#">578</a>        | CB_DEVICE_SYSERR<br>tree.h, <a href="#">579</a>         |
| CB_CATEGORY_ALPHANUMERIC_EDITED<br>tree.h, <a href="#">578</a> | CB_DEVICE_SYSIN<br>tree.h, <a href="#">579</a>          |
| CB_CATEGORY_BOOLEAN<br>tree.h, <a href="#">578</a>             | CB_DEVICE_SYSOUT<br>tree.h, <a href="#">579</a>         |
| CB_CATEGORY_DATA_POINTER<br>tree.h, <a href="#">578</a>        | CB_ENTRY_CONVENTION_NAME<br>tree.h, <a href="#">586</a> |
| CB_CATEGORY_INDEX<br>tree.h, <a href="#">578</a>               | CB_ERROR<br>cobc.h, <a href="#">160</a>                 |
| CB_CATEGORY_NATIONAL<br>tree.h, <a href="#">578</a>            | CB_EXTERNAL_LOCALE_NAME<br>tree.h, <a href="#">586</a>  |
| CB_CATEGORY_NATIONAL_EDITED<br>tree.h, <a href="#">578</a>     | CB_FEATURE_C01<br>tree.h, <a href="#">579</a>           |
| CB_CATEGORY_NUMERIC<br>tree.h, <a href="#">578</a>             | CB_FEATURE_C02<br>tree.h, <a href="#">579</a>           |
| CB_CATEGORY_NUMERIC_EDITED<br>tree.h, <a href="#">578</a>      | CB_FEATURE_C03<br>tree.h, <a href="#">579</a>           |
| CB_CATEGORY_OBJECT_REFERENCE<br>tree.h, <a href="#">578</a>    | CB_FEATURE_C04<br>tree.h, <a href="#">579</a>           |
| CB_CATEGORY_PROGRAM_POINTER<br>tree.h, <a href="#">578</a>     | CB_FEATURE_C05<br>tree.h, <a href="#">579</a>           |
| CB_CATEGORY_UNKNOWN<br>tree.h, <a href="#">578</a>             | CB_FEATURE_C06<br>tree.h, <a href="#">579</a>           |
| CB_CLASS_ALPHABETIC<br>tree.h, <a href="#">578</a>             | CB_FEATURE_C07<br>tree.h, <a href="#">579</a>           |
| CB_CLASS_ALPHANUMERIC<br>tree.h, <a href="#">578</a>           | CB_FEATURE_C08<br>tree.h, <a href="#">579</a>           |
| CB_CLASS_BOOLEAN<br>tree.h, <a href="#">578</a>                | CB_FEATURE_C09<br>tree.h, <a href="#">580</a>           |

- CB\_FEATURE\_C10  
tree.h, 580
- CB\_FEATURE\_C11  
tree.h, 580
- CB\_FEATURE\_C12  
tree.h, 580
- CB\_FEATURE\_FORMFEED  
tree.h, 579
- CB\_FEATURE\_NAME  
tree.h, 586
- CB\_IGNORE  
cobc.h, 160
- CB\_INTR\_ABS  
tree.h, 580
- CB\_INTR\_ACOS  
tree.h, 580
- CB\_INTR\_ANNUITY  
tree.h, 580
- CB\_INTR\_ASIN  
tree.h, 580
- CB\_INTR\_ATAN  
tree.h, 580
- CB\_INTR\_BOOLEAN\_OF\_INTEGER  
tree.h, 580
- CB\_INTR\_BYTE\_LENGTH  
tree.h, 580
- CB\_INTR\_CHAR  
tree.h, 580
- CB\_INTR\_CHAR\_NATIONAL  
tree.h, 580
- CB\_INTR\_COMBINED\_DATETIME  
tree.h, 580
- CB\_INTR\_CONCATENATE  
tree.h, 580
- CB\_INTR\_COS  
tree.h, 580
- CB\_INTR\_CURRENT\_DATE  
tree.h, 580
- CB\_INTR\_DATE\_OF\_INTEGER  
tree.h, 580
- CB\_INTR\_DATE\_TO\_YYYYMMDD  
tree.h, 580
- CB\_INTR\_DAY\_OF\_INTEGER  
tree.h, 580
- CB\_INTR\_DAY\_TO\_YYYYDDD  
tree.h, 580
- CB\_INTR\_DISPLAY\_OF  
tree.h, 580
- CB\_INTR\_E  
tree.h, 581
- CB\_INTR\_EXCEPTION\_FILE  
tree.h, 581
- CB\_INTR\_EXCEPTION\_FILE\_N  
tree.h, 581
- CB\_INTR\_EXCEPTION\_LOCATION  
tree.h, 581
- CB\_INTR\_EXCEPTION\_LOCATION\_N  
tree.h, 581
- CB\_INTR\_EXCEPTION\_STATEMENT  
tree.h, 581
- CB\_INTR\_EXCEPTION\_STATUS  
tree.h, 581
- CB\_INTR\_EXP  
tree.h, 581
- CB\_INTR\_EXP10  
tree.h, 581
- CB\_INTR\_FACTORIAL  
tree.h, 581
- CB\_INTR\_FRACTION\_PART  
tree.h, 581
- CB\_INTR\_HIGHEST\_ALGEBRAIC  
tree.h, 581
- CB\_INTR\_INTEGER  
tree.h, 581
- CB\_INTR\_INTEGER\_OF\_BOOLEAN  
tree.h, 581
- CB\_INTR\_INTEGER\_OF\_DATE  
tree.h, 581
- CB\_INTR\_INTEGER\_OF\_DAY  
tree.h, 581
- CB\_INTR\_INTEGER\_PART  
tree.h, 581
- CB\_INTR\_LENGTH  
tree.h, 581
- CB\_INTR\_LOCALE\_COMPARE  
tree.h, 581
- CB\_INTR\_LOCALE\_DATE  
tree.h, 581
- CB\_INTR\_LOCALE\_TIME  
tree.h, 581
- CB\_INTR\_LOCALE\_TIME\_FROM\_SECS  
tree.h, 581
- CB\_INTR\_LOG  
tree.h, 581
- CB\_INTR\_LOG10  
tree.h, 581
- CB\_INTR\_LOWER\_CASE  
tree.h, 581
- CB\_INTR\_LOWEST\_ALGEBRAIC  
tree.h, 581

- CB\_INTR\_MAX  
     tree.h, [581](#)
- CB\_INTR\_MEAN  
     tree.h, [581](#)
- CB\_INTR\_MEDIAN  
     tree.h, [581](#)
- CB\_INTR\_MIDRANGE  
     tree.h, [581](#)
- CB\_INTR\_MIN  
     tree.h, [581](#)
- CB\_INTR\_MOD  
     tree.h, [581](#)
- CB\_INTR\_NATIONAL\_OF  
     tree.h, [581](#)
- CB\_INTR\_NUMVAL  
     tree.h, [581](#)
- CB\_INTR\_NUMVAL\_C  
     tree.h, [581](#)
- CB\_INTR\_NUMVAL\_F  
     tree.h, [581](#)
- CB\_INTR\_ORD  
     tree.h, [581](#)
- CB\_INTR\_ORD\_MAX  
     tree.h, [581](#)
- CB\_INTR\_ORD\_MIN  
     tree.h, [582](#)
- CB\_INTR\_PI  
     tree.h, [582](#)
- CB\_INTR\_PRESENT\_VALUE  
     tree.h, [582](#)
- CB\_INTR\_RANDOM  
     tree.h, [582](#)
- CB\_INTR\_RANGE  
     tree.h, [582](#)
- CB\_INTR\_REM  
     tree.h, [582](#)
- CB\_INTR\_REVERSE  
     tree.h, [582](#)
- CB\_INTR\_SECONDS\_FROM\_FORMATTED\_  
     TIME  
     tree.h, [582](#)
- CB\_INTR\_SECONDS\_PAST\_MIDNIGHT  
     tree.h, [582](#)
- CB\_INTR\_SIGN  
     tree.h, [582](#)
- CB\_INTR\_SIN  
     tree.h, [582](#)
- CB\_INTR\_SQRT  
     tree.h, [582](#)
- CB\_INTR\_STANDARD\_COMPARE  
     tree.h, [582](#)
- CB\_INTR\_STANDARD\_DEVIATION  
     tree.h, [582](#)
- CB\_INTR\_STORED\_CHAR\_LENGTH  
     tree.h, [582](#)
- CB\_INTR\_SUBSTITUTE  
     tree.h, [582](#)
- CB\_INTR\_SUBSTITUTE\_CASE  
     tree.h, [582](#)
- CB\_INTR\_SUM  
     tree.h, [582](#)
- CB\_INTR\_TAN  
     tree.h, [582](#)
- CB\_INTR\_TEST\_DATE\_YYYYMMDD  
     tree.h, [582](#)
- CB\_INTR\_TEST\_DAY\_YYYYDDD  
     tree.h, [582](#)
- CB\_INTR\_TEST\_NUMVAL  
     tree.h, [582](#)
- CB\_INTR\_TEST\_NUMVAL\_C  
     tree.h, [582](#)
- CB\_INTR\_TEST\_NUMVAL\_F  
     tree.h, [582](#)
- CB\_INTR\_TRIM  
     tree.h, [582](#)
- CB\_INTR\_UPPER\_CASE  
     tree.h, [582](#)
- CB\_INTR\_VARIANCE  
     tree.h, [582](#)
- CB\_INTR\_WHEN\_COMPILED  
     tree.h, [582](#)
- CB\_INTR\_YEAR\_TO\_YYYY  
     tree.h, [582](#)
- CB\_LEVEL\_ASSEMBLE  
     cobc.c, [145](#)
- CB\_LEVEL\_COMPILE  
     cobc.c, [145](#)
- CB\_LEVEL\_EXECUTABLE  
     cobc.c, [145](#)
- CB\_LEVEL\_LIBRARY  
     cobc.c, [145](#)
- CB\_LEVEL\_MODULE  
     cobc.c, [145](#)
- CB\_LEVEL\_PREPROCESS  
     cobc.c, [145](#)
- CB\_LEVEL\_TRANSLATE  
     cobc.c, [145](#)
- CB\_LIBRARY\_NAME  
     tree.h, [586](#)
- CB\_OBSOLETE

- cobc.h, [160](#)
- CB\_OK
  - cobc.h, [160](#)
- CB\_OPERATION\_ASSIGN
  - cobc.h, [160](#)
- CB\_OPERATION\_READ
  - cobc.h, [160](#)
- CB\_OPERATION\_WRITE
  - cobc.h, [160](#)
- CB\_PERFORM\_EXIT
  - tree.h, [584](#)
- CB\_PERFORM\_FOREVER
  - tree.h, [584](#)
- CB\_PERFORM\_ONCE
  - tree.h, [584](#)
- CB\_PERFORM\_TIMES
  - tree.h, [584](#)
- CB\_PERFORM\_UNTIL
  - tree.h, [584](#)
- CB\_RECEIVING\_OPERAND
  - tree.h, [584](#)
- CB\_SENDING\_OPERAND
  - tree.h, [584](#)
- CB\_SKIP
  - cobc.h, [160](#)
- CB\_STORAGE\_COMMUNICATION
  - tree.h, [585](#)
- CB\_STORAGE\_CONSTANT
  - tree.h, [585](#)
- CB\_STORAGE\_FILE
  - tree.h, [585](#)
- CB\_STORAGE\_LINKAGE
  - tree.h, [585](#)
- CB\_STORAGE\_LOCAL
  - tree.h, [585](#)
- CB\_STORAGE\_REPORT
  - tree.h, [585](#)
- CB\_STORAGE\_SCREEN
  - tree.h, [585](#)
- CB\_STORAGE\_WORKING
  - tree.h, [585](#)
- CB\_SWITCH\_1
  - tree.h, [585](#)
- CB\_SWITCH\_2
  - tree.h, [585](#)
- CB\_SWITCH\_3
  - tree.h, [585](#)
- CB\_SWITCH\_4
  - tree.h, [585](#)
- CB\_SWITCH\_5
  - tree.h, [585](#)
- CB\_SWITCH\_6
  - tree.h, [585](#)
- CB\_SWITCH\_7
  - tree.h, [585](#)
- CB\_SWITCH\_8
  - tree.h, [585](#)
- CB\_SWITCH\_NAME
  - tree.h, [586](#)
- CB\_TAG\_ALPHABET\_NAME
  - tree.h, [586](#)
- CB\_TAG\_ASSIGN
  - tree.h, [587](#)
- CB\_TAG\_BINARY\_OP
  - tree.h, [587](#)
- CB\_TAG\_CALL
  - tree.h, [587](#)
- CB\_TAG\_CAST
  - tree.h, [587](#)
- CB\_TAG\_CLASS\_NAME
  - tree.h, [586](#)
- CB\_TAG\_CONST
  - tree.h, [586](#)
- CB\_TAG\_CONTINUE
  - tree.h, [587](#)
- CB\_TAG\_DECIMAL
  - tree.h, [586](#)
- CB\_TAG\_FIELD
  - tree.h, [586](#)
- CB\_TAG\_FILE
  - tree.h, [586](#)
- CB\_TAG\_FUNCALL
  - tree.h, [587](#)
- CB\_TAG\_GOTO
  - tree.h, [587](#)
- CB\_TAG\_IF
  - tree.h, [587](#)
- CB\_TAG\_INITIALIZE
  - tree.h, [587](#)
- CB\_TAG\_INTEGER
  - tree.h, [586](#)
- CB\_TAG\_INTRINSIC
  - tree.h, [587](#)
- CB\_TAG\_LABEL
  - tree.h, [587](#)
- CB\_TAG\_LIST
  - tree.h, [587](#)
- CB\_TAG\_LITERAL
  - tree.h, [586](#)
- CB\_TAG\_LOCALE\_NAME

- tree.h, [586](#)
- CB\_TAG\_PERFORM
  - tree.h, [587](#)
- CB\_TAG\_PERFORM\_VARYING
  - tree.h, [587](#)
- CB\_TAG\_PICTURE
  - tree.h, [587](#)
- CB\_TAG\_REFERENCE
  - tree.h, [586](#)
- CB\_TAG\_SEARCH
  - tree.h, [587](#)
- CB\_TAG\_STATEMENT
  - tree.h, [587](#)
- CB\_TAG\_STRING
  - tree.h, [586](#)
- CB\_TAG\_SYSTEM\_NAME
  - tree.h, [586](#)
- CB\_TEXT\_NAME
  - tree.h, [586](#)
- CB\_UNCONFORMABLE
  - cobc.h, [160](#)
- CB\_USAGE\_BINARY
  - tree.h, [588](#)
- CB\_USAGE\_BIT
  - tree.h, [588](#)
- CB\_USAGE\_COMP\_5
  - tree.h, [588](#)
- CB\_USAGE\_COMP\_X
  - tree.h, [588](#)
- CB\_USAGE\_DISPLAY
  - tree.h, [588](#)
- CB\_USAGE\_DOUBLE
  - tree.h, [588](#)
- CB\_USAGE\_FLOAT
  - tree.h, [588](#)
- CB\_USAGE\_INDEX
  - tree.h, [588](#)
- CB\_USAGE\_LENGTH
  - tree.h, [588](#)
- CB\_USAGE\_NATIONAL
  - tree.h, [588](#)
- CB\_USAGE\_OBJECT
  - tree.h, [588](#)
- CB\_USAGE\_PACKED
  - tree.h, [588](#)
- CB\_USAGE\_POINTER
  - tree.h, [588](#)
- CB\_USAGE\_PROGRAM
  - tree.h, [588](#)
- CB\_USAGE\_PROGRAM\_POINTER
  - tree.h, [588](#)
- CB\_USAGE\_SIGNED\_CHAR
  - tree.h, [588](#)
- CB\_USAGE\_SIGNED\_INT
  - tree.h, [588](#)
- CB\_USAGE\_SIGNED\_LONG
  - tree.h, [588](#)
- CB\_USAGE\_SIGNED\_SHORT
  - tree.h, [588](#)
- CB\_USAGE\_UNSIGNED\_CHAR
  - tree.h, [588](#)
- CB\_USAGE\_UNSIGNED\_INT
  - tree.h, [588](#)
- CB\_USAGE\_UNSIGNED\_LONG
  - tree.h, [588](#)
- CB\_USAGE\_UNSIGNED\_SHORT
  - tree.h, [588](#)
- CB\_WARNING
  - cobc.h, [160](#)
- cb\_add\_78
  - tree.h, [591](#)
- CB\_AFTER
  - tree.h, [564](#)
- CB\_ALPHABET\_NAME
  - tree.h, [564](#)
- cb\_alphabet\_name, [16](#)
  - cname, [17](#)
  - common, [17](#)
  - custom\_list, [17](#)
  - high\_val\_char, [18](#)
  - low\_val\_char, [18](#)
  - name, [18](#)
  - type, [18](#)
- CB\_ALPHABET\_NAME\_P
  - tree.h, [564](#)
- cb\_alphabet\_name\_type
  - tree.h, [577](#)
- cb\_alt\_key, [18](#)
  - duplicates, [19](#)
  - key, [19](#)
  - next, [19](#)
  - offset, [19](#)
- cb\_any
  - tree.c, [549](#)
  - tree.h, [719](#)
- CB\_ASSIGN
  - tree.h, [565](#)
- cb\_assign, [19](#)
  - common, [20](#)
  - val, [20](#)



- var, 20
- cb\_assign\_clause
  - cobc.h, 159
- CB\_ASSIGN\_P
  - tree.h, 565
- cb\_attr\_id
  - cobc.c, 151
  - cobc.h, 168
- CB\_BEFORE
  - tree.h, 565
- cb\_binary\_byteorder
  - cobc.h, 159
- CB\_BINARY\_OP
  - tree.h, 565
- cb\_binary\_op, 21
  - common, 21
  - op, 21
  - x, 22
  - y, 22
- CB\_BINARY\_OP\_P
  - tree.h, 565
- cb\_binary\_size
  - cobc.h, 159
- cb\_build\_add
  - tree.h, 591
  - typeck.c, 727
- cb\_build\_address
  - tree.h, 591
  - typeck.c, 727
- cb\_build\_alphabet\_name
  - tree.c, 512
  - tree.h, 592
- cb\_build\_alphanumeric\_literal
  - tree.c, 513
  - tree.h, 592
- cb\_build\_any\_intrinsic
  - tree.c, 513
  - tree.h, 592
- cb\_build\_assign
  - tree.c, 513
  - tree.h, 592
- cb\_build\_assignment\_name
  - tree.h, 593
  - typeck.c, 727
- cb\_build\_binary\_list
  - tree.c, 513
  - tree.h, 594
- cb\_build\_binary\_op
  - tree.c, 514
  - tree.h, 594
- cb\_build\_call
  - tree.c, 515
  - tree.h, 595
- cb\_build\_cast
  - tree.c, 515
  - tree.h, 596
- cb\_build\_cast\_addr\_of\_addr
  - tree.h, 565
- cb\_build\_cast\_address
  - tree.h, 565
- cb\_build\_cast\_integer
  - tree.h, 565
- cb\_build\_cast\_length
  - tree.h, 565
- cb\_build\_cast\_ppointer
  - tree.h, 565
- cb\_build\_class\_name
  - tree.c, 516
  - tree.h, 596
- cb\_build\_cond
  - tree.h, 596
  - typeck.c, 729
- cb\_build\_const\_length
  - tree.h, 598
  - typeck.c, 731
- cb\_build\_constant
  - tree.c, 516
  - tree.h, 599
- cb\_build\_continue
  - tree.c, 516
  - tree.h, 599
- cb\_build\_converting
  - tree.h, 600
  - typeck.c, 731
- cb\_build\_decimal
  - tree.c, 516
  - tree.h, 600
- cb\_build\_display\_upon
  - tree.h, 600
  - typeck.c, 732
- cb\_build\_display\_upon\_direct
  - tree.h, 600
  - typeck.c, 732
- cb\_build\_expr
  - tree.h, 601
  - typeck.c, 733
- cb\_build\_field
  - tree.c, 517
  - tree.h, 602
- cb\_build\_field\_reference

- tree.c, 517
- tree.h, 603
- cb\_build\_field\_tree
  - field.c, 198
  - tree.h, 603
- cb\_build\_filler
  - tree.c, 517
  - tree.h, 606
- cb\_build\_funcall
  - tree.c, 518
  - tree.h, 606
- cb\_build\_funcall\_0
  - tree.h, 566
- cb\_build\_funcall\_1
  - tree.h, 566
- cb\_build\_funcall\_2
  - tree.h, 566
- cb\_build\_funcall\_3
  - tree.h, 566
- cb\_build\_funcall\_4
  - tree.h, 566
- cb\_build\_funcall\_5
  - tree.h, 566
- cb\_build\_funcall\_6
  - tree.h, 566
- cb\_build\_funcall\_7
  - tree.h, 566
- cb\_build\_goto
  - tree.c, 518
  - tree.h, 606
- cb\_build\_identifier
  - tree.h, 606
  - typeck.c, 734
- cb\_build\_if
  - tree.c, 518
  - tree.h, 610
- cb\_build\_implicit\_field
  - tree.c, 519
  - tree.h, 610
- cb\_build\_index
  - tree.h, 610
  - typeck.c, 737
- cb\_build\_initialize
  - tree.c, 519
  - tree.h, 611
- cb\_build\_inspect\_region
  - tree.h, 611
  - typeck.c, 737
- cb\_build\_inspect\_region\_start
  - tree.h, 611
- typeck.c, 738
- cb\_build\_intrinsic
  - tree.c, 519
  - tree.h, 611
- cb\_build\_label
  - tree.c, 522
  - tree.h, 614
- cb\_build\_length
  - tree.h, 614
  - typeck.c, 738
- cb\_build\_list
  - tree.c, 522
  - tree.h, 615
- cb\_build\_locale\_name
  - tree.c, 523
  - tree.h, 615
- cb\_build\_move
  - tree.h, 616
  - typeck.c, 738
- cb\_build\_negation
  - tree.h, 566
- cb\_build\_numeric\_literal
  - tree.c, 523
  - tree.h, 617
- cb\_build\_pair
  - tree.h, 567
- cb\_build\_parenthesis
  - tree.h, 567
- cb\_build\_perform
  - tree.c, 523
  - tree.h, 617
- cb\_build\_perform\_exit
  - tree.h, 618
  - typeck.c, 740
- cb\_build\_perform\_forever
  - tree.h, 618
  - typeck.c, 740
- cb\_build\_perform\_once
  - tree.h, 618
  - typeck.c, 740
- cb\_build\_perform\_times
  - tree.h, 618
  - typeck.c, 741
- cb\_build\_perform\_until
  - tree.h, 619
  - typeck.c, 741
- cb\_build\_perform\_varying
  - tree.c, 523
  - tree.h, 619
- cb\_build\_picture

- tree.c, [524](#)
- tree.h, [619](#)
- cb\_build\_ppointer
  - tree.h, [625](#)
  - typeck.c, [741](#)
- cb\_build\_program
  - tree.c, [529](#)
  - tree.h, [625](#)
- cb\_build\_program\_id
  - tree.h, [626](#)
  - typeck.c, [742](#)
- cb\_build\_reference
  - tree.c, [530](#)
  - tree.h, [626](#)
- cb\_build\_registers
  - tree.h, [627](#)
  - typeck.c, [742](#)
- cb\_build\_replacing\_all
  - tree.h, [628](#)
  - typeck.c, [744](#)
- cb\_build\_replacing\_characters
  - tree.h, [628](#)
  - typeck.c, [744](#)
- cb\_build\_replacing\_first
  - tree.h, [628](#)
  - typeck.c, [744](#)
- cb\_build\_replacing\_leading
  - tree.h, [628](#)
  - typeck.c, [744](#)
- cb\_build\_replacing\_trailing
  - tree.h, [629](#)
  - typeck.c, [744](#)
- cb\_build\_search
  - tree.c, [530](#)
  - tree.h, [629](#)
- cb\_build\_section\_name
  - tree.h, [629](#)
  - typeck.c, [744](#)
- cb\_build\_statement
  - tree.c, [530](#)
  - tree.h, [630](#)
- cb\_build\_string
  - tree.c, [531](#)
  - tree.h, [630](#)
- cb\_build\_string0
  - tree.h, [567](#)
- cb\_build\_sub
  - tree.h, [630](#)
  - typeck.c, [745](#)
- cb\_build\_system\_name
  - tree.c, [531](#)
  - tree.h, [631](#)
- cb\_build\_tarrying\_all
  - tree.h, [631](#)
  - typeck.c, [746](#)
- cb\_build\_tarrying\_characters
  - tree.h, [631](#)
  - typeck.c, [746](#)
- cb\_build\_tarrying\_data
  - tree.h, [632](#)
  - typeck.c, [746](#)
- cb\_build\_tarrying\_leading
  - tree.h, [632](#)
  - typeck.c, [746](#)
- cb\_build\_tarrying\_trailing
  - tree.h, [632](#)
  - typeck.c, [746](#)
- cb\_build\_tarrying\_value
  - tree.h, [632](#)
  - typeck.c, [747](#)
- cb\_build\_unstring\_delimited
  - tree.h, [633](#)
  - typeck.c, [747](#)
- cb\_build\_unstring\_into
  - tree.h, [633](#)
  - typeck.c, [747](#)
- cb\_build\_write\_advancing\_lines
  - tree.h, [633](#)
  - typeck.c, [747](#)
- cb\_build\_write\_advancing\_mnemonic
  - tree.h, [634](#)
  - typeck.c, [748](#)
- cb\_build\_write\_advancing\_page
  - tree.h, [634](#)
  - typeck.c, [748](#)
- CB\_CALL
  - tree.h, [567](#)
- cb\_call, [22](#)
  - args, [23](#)
  - common, [23](#)
  - is\_system, [23](#)
  - name, [23](#)
  - returning, [23](#)
  - stmt1, [23](#)
  - stmt2, [23](#)
- CB\_CALL\_BY\_CONTENT
  - tree.h, [567](#)
- CB\_CALL\_BY\_REFERENCE
  - tree.h, [567](#)
- CB\_CALL\_BY\_VALUE

- tree.h, 567
- CB\_CALL\_P
  - tree.h, 567
- cb\_call\_params
  - cb\_program, 73
- CB\_CAST
  - tree.h, 567
- cb\_cast, 24
  - common, 24
  - type, 24
  - val, 24
- CB\_CAST\_P
  - tree.h, 567
- cb\_cast\_type
  - tree.h, 577
- cb\_category
  - tree.h, 577
- CB\_CHAIN
  - tree.h, 568
- cb\_check\_numeric\_value
  - tree.h, 634
  - typeck.c, 748
- cb\_class
  - tree.h, 578
- CB\_CLASS\_NAME
  - tree.h, 568
- cb\_class\_name, 25
  - cname, 25
  - common, 26
  - list, 26
  - name, 26
- CB\_CLASS\_NAME\_P
  - tree.h, 568
- cb\_clear\_real\_field
  - field.c, 200
  - tree.h, 635
- cb\_compile\_level
  - cobc.c, 144
- cb\_concat\_literals
  - tree.c, 531
  - tree.h, 635
- CB\_CONFIG\_ANY
  - cobc.h, 157
  - config.c, 185
- CB\_CONFIG\_BOOLEAN
  - cobc.h, 157
  - config.c, 185
- CB\_CONFIG\_INT
  - cobc.h, 158
  - config.c, 185
- CB\_CONFIG\_STRING
  - cobc.h, 158
  - config.c, 185, 186
- CB\_CONFIG\_SUPPORT
  - cobc.h, 158
  - config.c, 186
- cb\_config\_type
  - config.c, 186
- cb\_cons
  - tree.h, 568
- CB\_CONST
  - tree.h, 568
- cb\_const, 26
  - common, 27
  - val, 27
- CB\_CONST\_P
  - tree.h, 568
- CB\_CONTINUE
  - tree.h, 568
- cb\_continue, 27
  - common, 28
- CB\_CONTINUE\_P
  - tree.h, 568
- CB\_DECIMAL
  - tree.h, 568
- cb\_decimal, 28
  - common, 29
  - id, 29
- CB\_DECIMAL\_P
  - tree.h, 568
- cb\_define
  - tree.c, 532
  - tree.h, 636
- cb\_define\_switch\_name
  - tree.h, 636
  - typeck.c, 749
- cb\_define\_system\_name
  - tree.c, 533
  - tree.h, 637
- cb\_depend\_file
  - cobc.c, 152
  - cobc.h, 168
- cb\_depend\_list
  - cobc.c, 152
  - cobc.h, 168
- cb\_depend\_target
  - cobc.c, 152
  - cobc.h, 168
- cb\_device\_name
  - tree.h, 579

- cb\_display\_sign
  - cobc.c, [152](#)
  - cobc.h, [168](#)
- cb\_emit
  - typeck.c, [726](#)
- cb\_emit\_accept
  - tree.h, [637](#)
  - typeck.c, [749](#)
- cb\_emit\_accept\_arg\_number
  - tree.h, [639](#)
  - typeck.c, [751](#)
- cb\_emit\_accept\_arg\_value
  - tree.h, [639](#)
  - typeck.c, [751](#)
- cb\_emit\_accept\_command\_line
  - tree.h, [639](#)
  - typeck.c, [751](#)
- cb\_emit\_accept\_date
  - tree.h, [639](#)
  - typeck.c, [752](#)
- cb\_emit\_accept\_date\_yyyymmdd
  - tree.h, [640](#)
  - typeck.c, [752](#)
- cb\_emit\_accept\_day
  - tree.h, [640](#)
  - typeck.c, [752](#)
- cb\_emit\_accept\_day\_of\_week
  - tree.h, [640](#)
  - typeck.c, [752](#)
- cb\_emit\_accept\_day\_yyyddd
  - tree.h, [640](#)
  - typeck.c, [752](#)
- cb\_emit\_accept\_environment
  - tree.h, [641](#)
  - typeck.c, [753](#)
- cb\_emit\_accept\_line\_or\_col
  - tree.h, [641](#)
  - typeck.c, [753](#)
- cb\_emit\_accept\_mnemonic
  - tree.h, [641](#)
  - typeck.c, [753](#)
- cb\_emit\_accept\_name
  - tree.h, [641](#)
  - typeck.c, [753](#)
- cb\_emit\_accept\_time
  - tree.h, [642](#)
  - typeck.c, [754](#)
- cb\_emit\_allocate
  - tree.h, [642](#)
  - typeck.c, [754](#)
- cb\_emit\_arg\_number
  - tree.h, [643](#)
  - typeck.c, [755](#)
- cb\_emit\_arithmetic
  - tree.h, [643](#)
  - typeck.c, [755](#)
- cb\_emit\_call
  - tree.h, [644](#)
  - typeck.c, [756](#)
- cb\_emit\_cancel
  - tree.h, [646](#)
  - typeck.c, [758](#)
- cb\_emit\_close
  - tree.h, [646](#)
  - typeck.c, [758](#)
- cb\_emit\_command\_line
  - tree.h, [646](#)
  - typeck.c, [758](#)
- cb\_emit\_commit
  - tree.h, [647](#)
  - typeck.c, [759](#)
- cb\_emit\_continue
  - tree.h, [647](#)
  - typeck.c, [759](#)
- cb\_emit\_corresponding
  - tree.h, [647](#)
  - typeck.c, [759](#)
- cb\_emit\_delete
  - tree.h, [647](#)
  - typeck.c, [759](#)
- cb\_emit\_display
  - tree.h, [648](#)
  - typeck.c, [760](#)
- cb\_emit\_divide
  - tree.h, [650](#)
  - typeck.c, [762](#)
- cb\_emit\_env\_name
  - tree.h, [650](#)
  - typeck.c, [762](#)
- cb\_emit\_env\_value
  - tree.h, [650](#)
  - typeck.c, [762](#)
- cb\_emit\_evaluate
  - tree.h, [651](#)
  - typeck.c, [763](#)
- cb\_emit\_exit
  - tree.h, [651](#)
  - typeck.c, [763](#)
- cb\_emit\_free
  - tree.h, [651](#)

- typeck.c, [763](#)
- cb\_emit\_get\_environment
  - tree.h, [652](#)
  - typeck.c, [764](#)
- cb\_emit\_goto
  - tree.h, [652](#)
  - typeck.c, [764](#)
- cb\_emit\_if
  - tree.h, [653](#)
  - typeck.c, [765](#)
- cb\_emit\_initialize
  - tree.h, [653](#)
  - typeck.c, [765](#)
- cb\_emit\_inspect
  - tree.h, [653](#)
  - typeck.c, [765](#)
- cb\_emit\_list
  - typeck.c, [726](#)
- cb\_emit\_move
  - tree.h, [654](#)
  - typeck.c, [766](#)
- cb\_emit\_move\_corresponding
  - tree.h, [654](#)
  - typeck.c, [766](#)
- cb\_emit\_open
  - tree.h, [655](#)
  - typeck.c, [767](#)
- cb\_emit\_perform
  - tree.h, [655](#)
  - typeck.c, [767](#)
- cb\_emit\_read
  - tree.h, [655](#)
  - typeck.c, [767](#)
- cb\_emit\_release
  - tree.h, [656](#)
  - typeck.c, [768](#)
- cb\_emit\_return
  - tree.h, [657](#)
  - typeck.c, [769](#)
- cb\_emit\_rewrite
  - tree.h, [658](#)
  - typeck.c, [770](#)
- cb\_emit\_rollback
  - tree.h, [658](#)
  - typeck.c, [770](#)
- cb\_emit\_search
  - tree.h, [659](#)
  - typeck.c, [771](#)
- cb\_emit\_search\_all
  - tree.h, [659](#)
  - typeck.c, [771](#)
- cb\_emit\_set\_false
  - tree.h, [659](#)
  - typeck.c, [771](#)
- cb\_emit\_set\_on\_off
  - tree.h, [660](#)
  - typeck.c, [772](#)
- cb\_emit\_set\_to
  - tree.h, [660](#)
  - typeck.c, [772](#)
- cb\_emit\_set\_true
  - tree.h, [662](#)
  - typeck.c, [774](#)
- cb\_emit\_set\_up\_down
  - tree.h, [662](#)
  - typeck.c, [774](#)
- cb\_emit\_setenv
  - tree.h, [663](#)
  - typeck.c, [775](#)
- cb\_emit\_sort\_finish
  - tree.h, [663](#)
  - typeck.c, [775](#)
- cb\_emit\_sort\_giving
  - tree.h, [663](#)
  - typeck.c, [775](#)
- cb\_emit\_sort\_init
  - tree.h, [663](#)
  - typeck.c, [775](#)
- cb\_emit\_sort\_input
  - tree.h, [664](#)
  - typeck.c, [776](#)
- cb\_emit\_sort\_output
  - tree.h, [665](#)
  - typeck.c, [777](#)
- cb\_emit\_sort\_using
  - tree.h, [665](#)
  - typeck.c, [777](#)
- cb\_emit\_start
  - tree.h, [665](#)
  - typeck.c, [777](#)
- cb\_emit\_stop\_run
  - tree.h, [665](#)
  - typeck.c, [777](#)
- cb\_emit\_string
  - tree.h, [666](#)
  - typeck.c, [778](#)
- cb\_emit\_unlock
  - tree.h, [666](#)
  - typeck.c, [778](#)
- cb\_emit\_unstring

- tree.h, 667
- typeck.c, 779
- cb\_emit\_write
  - tree.h, 667
  - typeck.c, 779
- cb\_encode\_program\_id
  - tree.h, 668
  - typeck.c, 780
- cb\_error
  - cobc.h, 161
  - error.c, 193
- cb\_error\_node
  - tree.c, 549
  - tree.h, 719
- cb\_error\_x
  - error.c, 193
  - tree.h, 669
- cb\_exception, 29
  - code, 29
  - enable, 29
  - name, 29
- CB\_EXCEPTION\_CODE
  - cobc.h, 158
- CB\_EXCEPTION\_ENABLE
  - cobc.h, 158
- CB\_EXCEPTION\_NAME
  - cobc.h, 158
- cb\_exception\_table
  - cobc.c, 152
  - cobc.h, 168
- cb\_extension\_list
  - cobc.c, 152
  - cobc.h, 168
- cb\_false
  - tree.c, 549
  - tree.h, 719
- cb\_feature\_name
  - tree.h, 579
- CB\_FIELD
  - tree.h, 569
- cb\_field, 30
  - children, 32
  - common, 32
  - count, 32
  - ename, 32
  - false\_88, 33
  - file, 33
  - flag\_any\_length, 33
  - flag\_anylen\_done, 33
  - flag\_base, 33
  - flag\_binary\_swap, 33
  - flag\_blank\_zero, 33
  - flag\_chained, 33
  - flag\_external, 33
  - flag\_field, 33
  - flag\_indexed\_by, 34
  - flag\_invalid, 34
  - flag\_is\_c\_long, 34
  - flag\_is\_global, 34
  - flag\_is\_pdiv\_parm, 34
  - flag\_is\_pointer, 34
  - flag\_is\_verified, 34
  - flag\_item\_78, 34
  - flag\_item\_based, 34
  - flag\_item\_external, 34
  - flag\_justified, 35
  - flag\_local, 35
  - flag\_local\_allocated, 35
  - flag\_no\_init, 35
  - flag\_occurs, 35
  - flag\_real\_binary, 35
  - flag\_sign\_leading, 35
  - flag\_sign\_separate, 35
  - flag\_spare, 35
  - flag\_synchronized, 35
  - id, 36
  - index\_list, 36
  - index\_qual, 36
  - indexes, 36
  - keys, 36
  - level, 36
  - memory\_size, 36
  - name, 36
  - nkeys, 36
  - occursDepending, 36
  - occurs\_max, 36
  - occurs\_min, 37
  - offset, 37
  - param\_num, 37
  - parent, 37
  - pic, 37
  - redefines, 37
  - rename\_thru, 37
  - screen\_backg, 37
  - screen\_column, 37
  - screen\_flag, 37
  - screen\_foreg, 38
  - screen\_from, 38
  - screen\_line, 38
  - screen\_to, 38

- sister, [38](#)
- size, [38](#)
- storage, [38](#)
- storage\_id, [38](#)
- tree.c, [533](#)
- tree.h, [669](#)
- usage, [38](#)
- values, [38](#)
- cb\_field::cb\_key, [54](#)
  - dir, [55](#)
  - key, [55](#)
  - ref, [55](#)
  - val, [55](#)
- cb\_field\_add
  - tree.c, [533](#)
  - tree.h, [669](#)
- cb\_field\_founder
  - tree.c, [533](#)
  - tree.h, [670](#)
- cb\_field\_id
  - cobc.c, [152](#)
  - cobc.h, [168](#)
- CB\_FIELD\_P
  - tree.h, [569](#)
- cb\_field\_size
  - tree.c, [534](#)
  - tree.h, [670](#)
- cb\_field\_subordinate
  - tree.c, [534](#)
  - tree.h, [671](#)
- cb\_field\_variable\_address
  - tree.c, [535](#)
  - tree.h, [671](#)
- cb\_field\_variable\_size
  - tree.c, [535](#)
  - tree.h, [671](#)
- CB\_FILE
  - tree.h, [569](#)
- cb\_file, [39](#)
  - access\_mode, [41](#)
  - alt\_key\_list, [41](#)
  - assign, [41](#)
  - cname, [42](#)
  - common, [42](#)
  - external, [42](#)
  - external\_assign, [42](#)
  - file\_status, [42](#)
  - fileid\_assign, [42](#)
  - finalized, [42](#)
  - global, [42](#)
  - handler, [42](#)
  - handler\_prog, [42](#)
  - key, [43](#)
  - latbot, [43](#)
  - latfoot, [43](#)
  - laptop, [43](#)
  - linage, [43](#)
  - linage\_ctr, [43](#)
  - lock\_mode, [43](#)
  - name, [43](#)
  - optional, [43](#)
  - organization, [43](#)
  - record, [44](#)
  - record\_depending, [44](#)
  - record\_max, [44](#)
  - record\_min, [44](#)
  - same\_clause, [44](#)
  - sharing, [44](#)
  - special, [44](#)
- CB\_FILE\_P
  - tree.h, [569](#)
- cb\_fits\_int
  - tree.c, [535](#)
  - tree.h, [672](#)
- cb\_fits\_long\_long
  - tree.c, [536](#)
  - tree.h, [673](#)
- CB\_FLAG
  - cobc.c, [143](#)
  - cobc.h, [158](#)
- cb\_flag\_main
  - cobc.c, [152](#)
  - cobc.h, [168](#)
- CB\_FORMAT\_FIXED
  - cobc.h, [158](#)
- CB\_FORMAT\_FREE
  - cobc.h, [158](#)
- CB\_FUNCALL
  - tree.h, [569](#)
- cb\_funcall, [44](#)
  - argc, [45](#)
  - argv, [45](#)
  - common, [45](#)
  - name, [45](#)
  - screenptr, [46](#)
  - varcnt, [46](#)
- CB\_FUNCALL\_P
  - tree.h, [569](#)
- CB\_FUNCTION\_TYPE
  - tree.h, [569](#)



- cb\_get\_int
  - tree.c, 537
  - tree.h, 673
- cb\_get\_level
  - field.c, 200
  - tree.h, 674
- cb\_get\_long\_long
  - tree.c, 537
  - tree.h, 674
- CB\_GOTO
  - tree.h, 569
- cb\_goto, 46
  - common, 47
  - depending, 47
  - target, 47
- CB\_GOTO\_P
  - tree.h, 569
- cb\_high
  - tree.c, 549
  - tree.h, 719
- cb\_i
  - tree.c, 549
  - tree.h, 719
- cb\_id
  - cobc.c, 152
  - cobc.h, 168
- CB\_IF
  - tree.h, 569
- cb\_if, 47
  - common, 48
  - stmt1, 48
  - stmt2, 48
  - test, 48
- CB\_IF\_P
  - tree.h, 570
- cb\_include\_list
  - cobc.c, 152
  - cobc.h, 168
- CB\_INDEX\_P
  - tree.h, 570
- cb\_init\_constants
  - tree.c, 538
  - tree.h, 675
- cb\_init\_reserved
  - reserved.c, 493
  - tree.h, 676
- cb\_init\_tarrying
  - tree.h, 676
  - typeck.c, 781
- CB\_INITIALIZE
  - tree.h, 570
- cb\_initialize, 48
  - common, 49
  - def, 49
  - flag\_statement, 49
  - rep, 50
  - val, 50
  - var, 50
- CB\_INITIALIZE\_P
  - tree.h, 570
- cb\_int
  - tree.c, 539
  - tree.h, 676
- cb\_int0
  - tree.c, 549
  - tree.h, 720
- cb\_int1
  - tree.c, 549
  - tree.h, 720
- cb\_int2
  - tree.c, 549
  - tree.h, 720
- cb\_int3
  - tree.c, 549
  - tree.h, 720
- cb\_int4
  - tree.c, 549
  - tree.h, 720
- cb\_int5
  - tree.c, 550
  - tree.h, 720
- CB\_INTEGER
  - tree.h, 570
- cb\_integer, 50
  - common, 51
  - val, 51
- CB\_INTEGER\_P
  - tree.h, 570
- cb\_intr\_e
  - tree.c, 550
  - tree.h, 720
- cb\_intr\_enum
  - tree.h, 580
- cb\_intr\_pi
  - tree.c, 550
  - tree.h, 720
- cb\_intr\_whencomp
  - tree.c, 550
  - tree.h, 720
- CB\_INTRINSIC

- tree.h, 570
- cb\_intrinsic, 51
  - args, 52
  - common, 52
  - intr\_field, 52
  - intr\_tab, 52
  - length, 52
  - name, 52
  - offset, 52
- CB\_INTRINSIC\_P
  - tree.h, 570
- cb\_intrinsic\_table, 53
  - args, 53
  - category, 53
  - implemented, 53
  - intr\_enum, 53
  - intr\_routine, 53
  - name, 54
  - refmod, 54
- CB\_LABEL
  - tree.h, 570
- cb\_label, 55
  - children, 56
  - common, 57
  - exit\_label, 57
  - exit\_label\_ref, 57
  - id, 57
  - is\_entry, 57
  - is\_global, 57
  - is\_section, 57
  - name, 57
  - need\_begin, 57
  - need\_return, 57
  - orig\_name, 58
  - section, 58
  - spare, 58
- CB\_LABEL\_P
  - tree.h, 571
- cb\_level\_78, 59
  - fld78, 60
  - next, 60
- CB\_LIST
  - tree.h, 571
- cb\_list, 60
  - chain, 61
  - common, 61
  - purpose, 61
  - sizes, 61
  - value, 61
- cb\_list\_add
  - tree.c, 539
  - tree.h, 677
- cb\_list\_append
  - tree.c, 539
  - tree.h, 677
- cb\_list\_init
  - tree.h, 571
- cb\_list\_intrinsics
  - reserved.c, 493
  - tree.h, 677
- cb\_list\_length
  - tree.c, 540
  - tree.h, 678
- cb\_list\_map
  - tree.c, 540
  - tree.h, 678
- cb\_list\_mnemonics
  - reserved.c, 494
  - tree.h, 678
- CB\_LIST\_P
  - tree.h, 571
- cb\_list\_reserved
  - reserved.c, 494
  - tree.h, 678
- cb\_list\_reverse
  - tree.c, 540
  - tree.h, 679
- cb\_listing\_file
  - cobc.c, 153
  - cobc.h, 169
- CB\_LITERAL
  - tree.h, 571
- cb\_literal, 61
  - all, 62
  - common, 62
  - data, 62
  - scale, 62
  - sign, 63
  - size, 63
  - spare, 63
- cb\_literal\_id
  - cobc.c, 153
  - cobc.h, 169
- CB\_LITERAL\_P
  - tree.h, 571
- cb\_load\_conf
  - cobc.h, 161
  - config.c, 186
- cb\_load\_std
  - cobc.h, 164

- config.c, [190](#)
- CB\_LOCALE\_NAME
  - tree.h, [571](#)
- cb\_locale\_name, [63](#)
  - cname, [64](#)
  - common, [64](#)
  - list, [64](#)
  - name, [64](#)
- CB\_LOCALE\_NAME\_P
  - tree.h, [571](#)
- cb\_low
  - tree.c, [550](#)
  - tree.h, [720](#)
- CB\_NAME
  - tree.h, [571](#)
- cb\_name
  - tree.c, [540](#)
  - tree.h, [679](#)
- cb\_needs\_01
  - field.c, [204](#)
  - tree.h, [721](#)
- cb\_norm\_high
  - tree.c, [550](#)
  - tree.h, [721](#)
- cb\_norm\_low
  - tree.c, [550](#)
  - tree.h, [721](#)
- cb\_null
  - tree.c, [550](#)
  - tree.h, [721](#)
- CB\_NUMERIC\_LITERAL\_P
  - tree.h, [571](#)
- cb\_oc\_build\_stamp
  - cobc.c, [153](#)
  - cobc.h, [169](#)
- cb\_one
  - tree.c, [550](#)
  - tree.h, [721](#)
- cb\_operand\_type
  - tree.h, [584](#)
- cb\_operation\_type
  - cobc.h, [160](#)
- CB\_PAIR\_P
  - tree.h, [572](#)
- CB\_PAIR\_X
  - tree.h, [572](#)
- CB\_PAIR\_Y
  - tree.h, [572](#)
- CB\_PERFORM
  - tree.h, [572](#)
- cb\_perform, [64](#)
  - body, [65](#)
  - common, [65](#)
  - cycle\_label, [66](#)
  - data, [66](#)
  - exit\_label, [66](#)
  - test, [66](#)
  - type, [66](#)
  - varying, [66](#)
- CB\_PERFORM\_P
  - tree.h, [572](#)
- cb\_perform\_type
  - tree.h, [584](#)
- CB\_PERFORM\_VARYING
  - tree.h, [572](#)
- cb\_perform\_varying, [66](#)
  - common, [67](#)
  - from, [67](#)
  - name, [67](#)
  - step, [68](#)
  - until, [68](#)
- CB\_PICTURE
  - tree.h, [572](#)
- cb\_picture, [68](#)
  - category, [69](#)
  - common, [69](#)
  - digits, [69](#)
  - have\_sign, [69](#)
  - lenstr, [69](#)
  - orig, [69](#)
  - scale, [69](#)
  - size, [69](#)
  - spare, [70](#)
  - str, [70](#)
- CB\_PICTURE\_P
  - tree.h, [572](#)
- CB\_PREFIX\_ATTR
  - tree.h, [572](#)
- CB\_PREFIX\_BASE
  - tree.h, [572](#)
- CB\_PREFIX\_CONST
  - tree.h, [573](#)
- CB\_PREFIX\_DECIMAL
  - tree.h, [573](#)
- CB\_PREFIX\_FIELD
  - tree.h, [573](#)
- CB\_PREFIX\_FILE
  - tree.h, [573](#)
- CB\_PREFIX\_KEYS
  - tree.h, [573](#)

- CB\_PREFIX\_LABEL
  - tree.h, [573](#)
- CB\_PREFIX\_SEQUENCE
  - tree.h, [573](#)
- cb\_program, [70](#)
  - alphabet\_name\_list, [73](#)
  - cb\_call\_params, [73](#)
  - cb\_return\_code, [73](#)
  - cb\_sort\_return, [73](#)
  - class\_name\_list, [73](#)
  - class\_spec\_list, [73](#)
  - collating\_sequence, [73](#)
  - crt\_status, [73](#)
  - currency\_symbol, [74](#)
  - cursor\_pos, [74](#)
  - decimal\_index, [74](#)
  - decimal\_index\_max, [74](#)
  - decimal\_point, [74](#)
  - entry\_list, [74](#)
  - exec\_list, [74](#)
  - file\_list, [74](#)
  - flag\_chained, [74](#)
  - flag\_common, [74](#)
  - flag\_global\_use, [75](#)
  - flag\_initial, [75](#)
  - flag\_main, [75](#)
  - flag\_recursive, [75](#)
  - flag\_screen, [75](#)
  - flag\_validated, [75](#)
  - function\_spec\_list, [75](#)
  - gen\_decset, [75](#)
  - gen\_file\_error, [75](#)
  - gen\_ptrmanip, [75](#)
  - gen\_udecset, [76](#)
  - global\_file\_list, [76](#)
  - global\_handler, [76](#)
  - global\_list, [76](#)
  - interface\_spec\_list, [76](#)
  - label\_list, [76](#)
  - linkage\_storage, [76](#)
  - local\_file\_list, [76](#)
  - local\_storage, [76](#)
  - local\_storage\_file, [76](#)
  - local\_storage\_name, [77](#)
  - locale\_list, [77](#)
  - loop\_counter, [77](#)
  - nested\_level, [77](#)
  - next\_program, [77](#)
  - numeric\_separator, [77](#)
  - orig\_source\_name, [77](#)
  - parameter\_list, [77](#)
  - prog\_type, [77](#)
  - program\_id, [77](#)
  - program\_spec\_list, [78](#)
  - property\_spec\_list, [78](#)
  - reference\_list, [78](#)
  - returning, [78](#)
  - screen\_storage, [78](#)
  - source\_name, [78](#)
  - spare, [78](#)
  - symbolic\_list, [78](#)
  - word\_table, [78](#)
  - working\_storage, [78](#)
- CB\_PROGRAM\_TYPE
  - tree.h, [573](#)
- CB\_PURPOSE
  - tree.h, [573](#)
- CB\_PURPOSE\_INT
  - tree.h, [573](#)
- cb\_quote
  - tree.c, [550](#)
  - tree.h, [721](#)
- cb\_ref
  - tree.c, [541](#)
  - tree.h, [680](#)
- CB\_REF\_OR\_FIELD\_P
  - tree.h, [574](#)
- CB\_REFERENCE
  - tree.h, [574](#)
- cb\_reference, [79](#)
  - all, [80](#)
  - chain, [80](#)
  - check, [80](#)
  - common, [80](#)
  - length, [80](#)
  - offset, [80](#)
  - subs, [80](#)
  - type, [80](#)
  - value, [81](#)
  - word, [81](#)
- CB\_REFERENCE\_P
  - tree.h, [574](#)
- cb\_replace\_list, [81](#)
  - new\_text, [82](#)
  - next, [82](#)
  - old\_text, [82](#)
- cb\_reset\_78
  - tree.h, [682](#)
- cb\_reset\_in\_procedure
  - tree.h, [683](#)

- cb\_resolve\_redefines
  - field.c, 201
  - tree.h, 683
- cb\_return\_code
  - cb\_program, 73
- cb\_saveargc
  - cobc.c, 153
  - cobc.h, 169
- cb\_saveargv
  - cobc.c, 153
  - cobc.h, 169
- CB\_SEARCH
  - tree.h, 574
- cb\_search, 82
  - common, 83
  - end\_stmt, 83
  - flag\_all, 83
  - table, 84
  - var, 84
  - whens, 84
- CB\_SEARCH\_P
  - tree.h, 574
- cb\_set\_in\_procedure
  - tree.h, 684
- CB\_SIZE\_1
  - tree.h, 574
- CB\_SIZE\_2
  - tree.h, 574
- CB\_SIZE\_4
  - tree.h, 574
- CB\_SIZE\_8
  - tree.h, 574
- CB\_SIZE\_AUTO
  - tree.h, 574
- CB\_SIZE\_UNSIGNED
  - tree.h, 575
- CB\_SIZES
  - tree.h, 575
- CB\_SIZES\_INT
  - tree.h, 575
- CB\_SIZES\_INT\_UNSIGNED
  - tree.h, 575
- cb\_sort\_return
  - cb\_program, 73
- cb\_source\_file
  - cobc.c, 153
  - cobc.h, 169
- cb\_source\_format
  - cobc.c, 153
  - cobc.h, 169
- cb\_source\_line
  - cobc.c, 153
  - cobc.h, 169
- cb\_space
  - tree.c, 551
  - tree.h, 721
- cb\_standard\_error\_handler
  - tree.c, 551
  - tree.h, 721
- CB\_STATEMENT
  - tree.h, 575
- cb\_statement, 84
  - body, 85
  - common, 85
  - file, 86
  - handler1, 86
  - handler2, 86
  - handler3, 86
  - handler\_id, 86
  - name, 86
  - need\_terminator, 86
  - null\_check, 86
- CB\_STATEMENT\_P
  - tree.h, 575
- cb\_storage
  - tree.h, 584
- cb\_storage\_file
  - cobc.c, 153
  - cobc.h, 169
- cb\_storage\_file\_name
  - cobc.c, 153
  - cobc.h, 169
- cb\_storage\_id
  - cobc.c, 154
  - cobc.h, 170
- CB\_STRING
  - tree.h, 575
- cb\_string, 87
  - common, 87
  - data, 87
  - size, 87
- CB\_STRING\_P
  - tree.h, 575
- cb\_support
  - cobc.h, 160
- cb\_switch\_name
  - tree.h, 585
- CB\_SYSTEM\_NAME
  - tree.h, 575
- cb\_system\_name, 88

- category, [88](#)
- common, [88](#)
- token, [88](#)
- cb\_system\_name\_category
  - tree.h, [585](#)
- CB\_SYSTEM\_NAME\_P
  - tree.h, [575](#)
- cb\_tag
  - tree.h, [586](#)
- cb\_text\_list, [89](#)
  - next, [89](#)
  - text, [89](#)
- cb\_text\_list\_add
  - cobc.c, [145](#)
  - cobc.h, [165](#)
- CB\_TREE
  - tree.h, [576](#)
- cb\_tree
  - tree.h, [577](#)
- CB\_TREE\_CAST
  - tree.h, [576](#)
- CB\_TREE\_CATEGORY
  - tree.h, [576](#)
- cb\_tree\_category
  - tree.c, [544](#)
  - tree.h, [684](#)
- CB\_TREE\_CLASS
  - tree.h, [576](#)
- cb\_tree\_class
  - tree.c, [545](#)
  - tree.h, [685](#)
- cb\_tree\_common, [90](#)
  - category, [90](#)
  - source\_file, [90](#)
  - source\_line, [90](#)
  - tag, [90](#)
- CB\_TREE\_TAG
  - tree.h, [576](#)
- cb\_tree\_type
  - tree.c, [545](#)
  - tree.h, [685](#)
- cb\_true
  - tree.c, [551](#)
  - tree.h, [721](#)
- cb\_usage
  - tree.h, [588](#)
- cb\_validate\_78\_item
  - field.c, [202](#)
  - tree.h, [686](#)
- cb\_validate\_88\_item
  - field.c, [203](#)
  - tree.h, [687](#)
- cb\_validate\_field
  - field.c, [203](#)
  - tree.h, [687](#)
- cb\_validate\_program\_body
  - tree.h, [688](#)
  - typeck.c, [781](#)
- cb\_validate\_program\_data
  - tree.h, [688](#)
  - typeck.c, [782](#)
- cb\_validate\_program\_environment
  - tree.h, [691](#)
  - typeck.c, [784](#)
- CB\_VALUE
  - tree.h, [576](#)
- cb\_verify
  - cobc.h, [165](#)
  - error.c, [193](#)
- CB\_WARNDEF
  - cobc.c, [143](#), [144](#)
  - cobc.h, [158](#)
- cb\_warning
  - cobc.h, [166](#)
  - error.c, [194](#)
- cb\_warning\_x
  - error.c, [194](#)
  - tree.h, [694](#)
- cb\_word, [90](#)
  - count, [91](#)
  - error, [91](#)
  - items, [91](#)
  - name, [91](#)
  - next, [92](#)
- CB\_WORD\_HASH\_SIZE
  - tree.h, [576](#)
- cb\_zero
  - tree.c, [551](#)
  - tree.h, [721](#)
- cbj\_int
  - cobjmp\_buf, [112](#)
- cbj\_jmp\_buf
  - cobjmp\_buf, [112](#)
- cbj\_ptr
  - cobjmp\_buf, [112](#)
- cbj\_ptr\_rest
  - cobjmp\_buf, [112](#)
- CBL\_AND
  - common.h, [887](#)
- CBL\_CHANGE\_DIR

- fileio.c, [897](#)
- fileio.h, [936](#)
- CBL\_CHECK\_FILE\_EXIST
  - fileio.c, [897](#)
  - fileio.h, [936](#)
- CBL\_CLOSE\_FILE
  - fileio.c, [898](#)
  - fileio.h, [937](#)
- CBL\_COPY\_FILE
  - fileio.c, [898](#)
  - fileio.h, [937](#)
- CBL\_CREATE\_DIR
  - fileio.c, [899](#)
  - fileio.h, [938](#)
- CBL\_CREATE\_FILE
  - fileio.c, [900](#)
  - fileio.h, [938](#)
- CBL\_DELETE\_DIR
  - fileio.c, [900](#)
  - fileio.h, [939](#)
- CBL\_DELETE\_FILE
  - fileio.c, [900](#)
  - fileio.h, [939](#)
- CBL\_EQ
  - common.h, [887](#)
- CBL\_ERROR\_PROC
  - common.h, [887](#)
- CBL\_EXIT\_PROC
  - common.h, [887](#)
- CBL\_FLUSH\_FILE
  - fileio.c, [901](#)
  - fileio.h, [939](#)
- CBL\_GET\_CURRENT\_DIR
  - fileio.c, [901](#)
  - fileio.h, [940](#)
- CBL\_IMP
  - common.h, [887](#)
- CBL\_NIMP
  - common.h, [887](#)
- CBL\_NOR
  - common.h, [887](#)
- CBL\_NOT
  - common.h, [887](#)
- CBL\_OC\_NANOSLEEP
  - common.h, [888](#)
- CBL\_OPEN\_FILE
  - fileio.c, [902](#)
  - fileio.h, [940](#)
- CBL\_OR
  - common.h, [888](#)
- CBL\_READ\_FILE
  - fileio.c, [902](#)
  - fileio.h, [941](#)
- CBL\_RENAME\_FILE
  - fileio.c, [903](#)
  - fileio.h, [941](#)
- CBL\_TOLOWER
  - common.h, [888](#)
- CBL\_Toupper
  - common.h, [888](#)
- CBL\_WRITE\_FILE
  - fileio.c, [903](#)
  - fileio.h, [942](#)
- CBL\_X91
  - common.h, [888](#)
- CBL\_XF4
  - common.h, [888](#)
- CBL\_XF5
  - common.h, [888](#)
- CBL\_XOR
  - common.h, [888](#)
- CH
  - parser.c, [228](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [480](#)
- chain
  - cb\_list, [61](#)
  - cb\_reference, [80](#)
- CHAINING
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [480](#)
- CHARACTER
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- CHARACTERS
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- check
  - cb\_reference, [80](#)
- check\_filler\_name
  - error.c, [195](#)
- tree.h, [695](#)
- check\_level\_78

- tree.h, [695](#)
- child
  - \_\_cob\_screen, [14](#)
- children
  - cb\_field, [32](#)
  - cb\_label, [56](#)
- CLASS
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- class\_name\_list
  - cb\_program, [73](#)
- class\_spec\_list
  - cb\_program, [73](#)
- CLOSE
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [331](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- close
  - cb\_fileio\_funcs, [104](#)
- cname
  - cb\_alphabet\_name, [17](#)
  - cb\_class\_name, [25](#)
  - cb\_file, [42](#)
  - cb\_locale\_name, [64](#)
- COB\_EC\_MAX
  - common.h, [887](#)
- COB\_EC\_ZERO
  - common.h, [887](#)
- cob\_accept
  - termio.c, [1144](#)
  - termio.h, [1147](#)
- cob\_accept\_arg\_number
  - common.h, [888](#)
- cob\_accept\_arg\_value
  - common.h, [888](#)
- cob\_accept\_command\_line
  - common.h, [888](#)
- cob\_accept\_date
  - common.h, [888](#)
- cob\_accept\_date\_yyyymmdd
  - common.h, [888](#)
- cob\_accept\_day
  - common.h, [888](#)
- cob\_accept\_day\_of\_week
  - common.h, [888](#)
- cob\_accept\_day\_yyyddd
  - common.h, [888](#)
- cob\_accept\_environment
  - common.h, [888](#)
- cob\_accept\_time
  - common.h, [888](#)
- COB\_ACCESS\_DYNAMIC
  - fileio.h, [927](#)
- COB\_ACCESS\_RANDOM
  - fileio.h, [927](#)
- COB\_ACCESS\_SEQUENTIAL
  - fileio.h, [927](#)
- cob\_acuw\_chdir
  - fileio.c, [904](#)
  - fileio.h, [942](#)
- cob\_acuw\_copyfile
  - fileio.c, [904](#)
  - fileio.h, [943](#)
- cob\_acuw\_file\_delete
  - fileio.c, [905](#)
  - fileio.h, [943](#)
- cob\_acuw\_file\_info
  - fileio.c, [905](#)
  - fileio.h, [944](#)
- cob\_acuw\_justify
  - common.h, [888](#)
- cob\_acuw\_mkdir
  - fileio.c, [906](#)
  - fileio.h, [944](#)
- cob\_acuw\_sleep
  - common.h, [888](#)
- cob\_add
  - numeric.c, [1081](#)
  - numeric.h, [1094](#)
- cob\_add\_align\_s16\_binary
  - codegen.h, [865](#)
- cob\_add\_align\_s32\_binary
  - codegen.h, [865](#)
- cob\_add\_align\_s64\_binary
  - codegen.h, [865](#)
- cob\_add\_align\_u16\_binary
  - codegen.h, [865](#)
- cob\_add\_align\_u32\_binary
  - codegen.h, [865](#)
- cob\_add\_align\_u64\_binary
  - codegen.h, [865](#)
- cob\_add\_int
  - numeric.c, [1081](#)
  - numeric.h, [1094](#)
- cob\_add\_packed\_int
  - codegen.h, [865](#)
- cob\_add\_s16\_binary



- codegen.h, [865](#)
- cob\_add\_s24\_binary
  - codegen.h, [865](#)
- cob\_add\_s32\_binary
  - codegen.h, [865](#)
- cob\_add\_s40\_binary
  - codegen.h, [865](#)
- cob\_add\_s48\_binary
  - codegen.h, [865](#)
- cob\_add\_s56\_binary
  - codegen.h, [866](#)
- cob\_add\_s64\_binary
  - codegen.h, [866](#)
- cob\_add\_s8\_binary
  - codegen.h, [866](#)
- cob\_add\_u16\_binary
  - codegen.h, [866](#)
- cob\_add\_u24\_binary
  - codegen.h, [866](#)
- cob\_add\_u32\_binary
  - codegen.h, [866](#)
- cob\_add\_u40\_binary
  - codegen.h, [866](#)
- cob\_add\_u48\_binary
  - codegen.h, [866](#)
- cob\_add\_u56\_binary
  - codegen.h, [866](#)
- cob\_add\_u64\_binary
  - codegen.h, [866](#)
- cob\_add\_u8\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s16\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s24\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s32\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s40\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s48\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s56\_binary
  - codegen.h, [866](#)
- cob\_addswp\_s64\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u16\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u24\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u32\_binary
  - codegen.h, [866](#)
- codegen.h, [866](#)
- cob\_addswp\_u40\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u48\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u56\_binary
  - codegen.h, [866](#)
- cob\_addswp\_u64\_binary
  - codegen.h, [867](#)
- cob\_alloc\_cache, [92](#)
- cob\_pointer, [92](#)
  - next, [92](#)
  - size, [93](#)
- cob\_allocate
  - common.h, [888](#)
- COB\_ASCENDING
  - fileio.h, [927](#)
- COB\_ATTR\_INIT
  - coblocal.h, [855](#)
- COB\_BSWAP\_16
  - byteswap.h, [833](#)
- COB\_BSWAP\_16\_CONSTANT
  - byteswap.h, [833](#)
- COB\_BSWAP\_32
  - byteswap.h, [833](#)
- COB\_BSWAP\_32\_CONSTANT
  - byteswap.h, [833](#)
- COB\_BSWAP\_64
  - byteswap.h, [834](#)
- COB\_BSWAP\_64\_CONSTANT
  - byteswap.h, [834](#)
- cob\_call\_error
  - call.c, [837](#)
  - call.h, [847](#)
- cob\_call\_params
  - common.c, [873](#)
  - common.h, [891](#)
- cob\_call\_resolve
  - call.c, [837](#)
  - call.h, [847](#)
- cob\_call\_resolve\_1
  - call.c, [838](#)
  - call.h, [847](#)
- COB\_CC
  - defaults.h, [808](#)
- COB\_CFLAGS
  - defaults.h, [808](#)
- cob\_chain\_setup
  - common.h, [888](#)
- cob\_check\_based

- common.h, [888](#)
- cob\_check\_numeric
  - common.h, [888](#)
- cob\_check\_odo
  - common.h, [889](#)
- cob\_check\_ref\_mod
  - common.h, [889](#)
- cob\_check\_subscript
  - common.h, [889](#)
- cob\_check\_version
  - common.h, [889](#)
- COB\_CHK\_PARAMS
  - coblocal.h, [855](#)
- cob\_close
  - fileio.c, [906](#)
  - fileio.h, [945](#)
- COB\_CLOSE\_LOCK
  - fileio.h, [927](#)
- COB\_CLOSE\_NO\_REWIND
  - fileio.h, [927](#)
- COB\_CLOSE\_NORMAL
  - fileio.h, [927](#)
- COB\_CLOSE\_UNIT
  - fileio.h, [927](#)
- COB\_CLOSE\_UNIT\_REMOVAL
  - fileio.h, [927](#)
- cob\_cmp
  - common.h, [889](#)
- cob\_cmp\_align\_s16\_binary
  - codegen.h, [867](#)
- cob\_cmp\_align\_s32\_binary
  - codegen.h, [867](#)
- cob\_cmp\_align\_s64\_binary
  - codegen.h, [867](#)
- cob\_cmp\_align\_u16\_binary
  - codegen.h, [867](#)
- cob\_cmp\_align\_u32\_binary
  - codegen.h, [867](#)
- cob\_cmp\_align\_u64\_binary
  - codegen.h, [867](#)
- cob\_cmp\_int
  - numeric.c, [1082](#)
  - numeric.h, [1094](#)
- cob\_cmp\_long\_numdisp
  - numeric.c, [1082](#)
  - numeric.h, [1095](#)
- cob\_cmp\_long\_sign\_numdisp
  - numeric.c, [1083](#)
  - numeric.h, [1095](#)
- cob\_cmp\_numdisp
  - numeric.c, [1083](#)
  - numeric.h, [1096](#)
- cob\_cmp\_packed
  - numeric.c, [1083](#)
  - numeric.h, [1096](#)
- cob\_cmp\_packed\_int
  - codegen.h, [867](#)
- cob\_cmp\_s16\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s24\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s32\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s40\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s48\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s56\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s64\_binary
  - codegen.h, [867](#)
- cob\_cmp\_s8\_binary
  - codegen.h, [867](#)
- cob\_cmp\_sign\_numdisp
  - numeric.c, [1085](#)
  - numeric.h, [1097](#)
- cob\_cmp\_u16\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u24\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u32\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u40\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u48\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u56\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u64\_binary
  - codegen.h, [867](#)
- cob\_cmp\_u8\_binary
  - codegen.h, [867](#)
- cob\_cmp\_uint
  - numeric.c, [1085](#)
  - numeric.h, [1098](#)
- cob\_cmpswp\_align\_s16\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_align\_s32\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_align\_s64\_binary

- codegen.h, [868](#)
- cob\_cmpswp\_align\_u16\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_align\_u32\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_align\_u64\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s16\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s24\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s32\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s40\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s48\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s56\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_s64\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u16\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u24\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u32\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u40\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u48\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u56\_binary
  - codegen.h, [868](#)
- cob\_cmpswp\_u64\_binary
  - codegen.h, [868](#)
- cob\_commit
  - fileio.c, [907](#)
  - fileio.h, [946](#)
- COB\_CONFIG\_DIR
  - defaults.h, [808](#)
- cob\_config\_dir
  - cobc.c, [154](#)
  - cobc.h, [170](#)
- COB\_COPY\_DIR
  - defaults.h, [808](#)
- cob\_current\_module
  - common.c, [873](#)
  - common.h, [891](#)
- cob\_d2i
  - common.h, [880](#)
- cob\_dbtsize\_t
  - fileio.c, [895](#)
- cob\_decimal, [93](#)
  - scale, [93](#)
  - value, [93](#)
- cob\_decimal\_add
  - numeric.c, [1085](#)
  - numeric.h, [1098](#)
- cob\_decimal\_cmp
  - numeric.c, [1086](#)
  - numeric.h, [1098](#)
- cob\_decimal\_div
  - numeric.c, [1086](#)
  - numeric.h, [1098](#)
- cob\_decimal\_get\_field
  - numeric.c, [1086](#)
  - numeric.h, [1099](#)
- cob\_decimal\_init
  - numeric.c, [1087](#)
  - numeric.h, [1100](#)
- cob\_decimal\_mul
  - numeric.c, [1088](#)
  - numeric.h, [1100](#)
- cob\_decimal\_pow
  - numeric.c, [1088](#)
  - numeric.h, [1100](#)
- cob\_decimal\_set\_field
  - numeric.c, [1088](#)
  - numeric.h, [1101](#)
- cob\_decimal\_sub
  - numeric.c, [1089](#)
  - numeric.h, [1101](#)
- cob\_default\_error\_handle
  - fileio.c, [907](#)
  - fileio.h, [946](#)
- cob\_delete
  - fileio.c, [909](#)
  - fileio.h, [947](#)
- COB\_DESCENDING
  - fileio.h, [928](#)
- cob\_display
  - termio.c, [1145](#)
  - termio.h, [1147](#)
- cob\_display\_arg\_number
  - common.h, [889](#)
- cob\_display\_command\_line
  - common.h, [889](#)
- cob\_display\_env\_value
  - common.h, [889](#)
- cob\_display\_environment

- common.h, [889](#)
- COB\_DISPLAY\_SIGN\_ASCII
  - common.h, [880](#)
- COB\_DISPLAY\_SIGN\_EBCDIC
  - common.h, [880](#)
- cob\_div\_quotient
  - numeric.c, [1089](#)
  - numeric.h, [1101](#)
- cob\_div\_remainder
  - numeric.c, [1089](#)
  - numeric.h, [1102](#)
- COB\_EQ
  - fileio.h, [928](#)
- COB\_ERRBUF\_SIZE
  - common.c, [872](#)
- cob\_error\_file
  - fileio.c, [923](#)
  - fileio.h, [960](#)
- COB\_EXCEPTION
  - cobc.c, [144](#)
  - common.c, [872](#), [873](#)
  - common.h, [880](#)
- cob\_exception\_93
  - code, [94](#)
  - critical, [94](#)
  - name, [94](#)
- cob\_exception\_code
  - common.c, [873](#)
  - common.h, [891](#)
- cob\_exception\_id
  - common.h, [887](#)
- COB\_EXEEXT
  - config.h, [798](#)
- cob\_exit\_fileio
  - coblocal.h, [855](#)
  - fileio.c, [909](#)
- COB\_EXPORT\_DYN
  - config.h, [798](#)
- cob\_external\_94
  - ename, [95](#)
  - esize, [95](#)
  - ext\_alloc, [95](#)
  - next, [95](#)
- cob\_external\_addr
  - common.h, [889](#)
- COB\_EXTRA\_FLAGS
  - config.h, [799](#)
- cob\_fatal\_error
  - common.h, [889](#)
- COB\_FERROR\_CHAINING
  - common.h, [880](#)
- COB\_FERROR\_CODEGEN
  - common.h, [880](#)
- COB\_FERROR\_INITIALIZED
  - common.h, [880](#)
- COB\_FERROR\_STACK
  - common.h, [880](#)
- cob\_field\_95
  - attr, [96](#)
  - data, [96](#)
  - size, [96](#)
- cob\_field\_accept
  - screenio.c, [1105](#)
  - screenio.h, [1117](#)
- cob\_field\_attr\_97
  - digits, [97](#)
  - flags, [97](#)
  - pic, [97](#)
  - scale, [97](#)
  - type, [97](#)
- COB\_FIELD\_BINARY\_SWAP
  - common.h, [881](#)
- COB\_FIELD\_BLANK\_ZERO
  - common.h, [881](#)
- cob\_field\_cancel
  - call.c, [838](#)
  - call.h, [848](#)
- COB\_FIELD\_DATA
  - common.h, [881](#)
- COB\_FIELD\_DIGITS
  - common.h, [881](#)
- cob\_field\_display
  - screenio.c, [1108](#)
  - screenio.h, [1121](#)
- COB\_FIELD\_FLAGS
  - common.h, [881](#)
- COB\_FIELD\_HAVE\_SIGN
  - common.h, [881](#)
- COB\_FIELD\_INIT
  - intrinsic.c, [963](#)
- COB\_FIELD\_IS\_NUMERIC
  - common.h, [881](#)
- COB\_FIELD\_IS\_POINTER
  - common.h, [881](#)
- COB\_FIELD\_JUSTIFIED
  - common.h, [881](#)
- COB\_FIELD\_PIC
  - common.h, [881](#)
- COB\_FIELD\_REAL\_BINARY
  - common.h, [882](#)

- COB\_FIELD\_SCALE
  - common.h, [882](#)
- COB\_FIELD\_SIGN\_LEADING
  - common.h, [882](#)
- COB\_FIELD\_SIGN\_SEPARATE
  - common.h, [882](#)
- COB\_FIELD\_SIZE
  - common.h, [882](#)
- cob\_field\_to\_string
  - coblocal.h, [856](#)
- COB\_FIELD\_TYPE
  - common.h, [882](#)
- cob\_file, [98](#)
  - access\_mode, [99](#)
  - assign, [99](#)
  - extfh\_ptr, [99](#)
  - file, [99](#)
  - file\_status, [99](#)
  - file\_version, [100](#)
  - flag\_begin\_of\_file, [100](#)
  - flag\_end\_of\_file, [100](#)
  - flag\_first\_read, [100](#)
  - flag\_needs\_nl, [100](#)
  - flag\_needs\_top, [100](#)
  - flag\_nonexistent, [100](#)
  - flag\_optional, [100](#)
  - flag\_read\_done, [100](#)
  - flag\_select\_features, [100](#)
  - keys, [101](#)
  - last\_open\_mode, [101](#)
  - linorkeyptr, [101](#)
  - lock\_mode, [101](#)
  - nkeys, [101](#)
  - open\_mode, [101](#)
  - organization, [101](#)
  - record, [101](#)
  - record\_max, [101](#)
  - record\_min, [101](#)
  - record\_size, [102](#)
  - select\_name, [102](#)
  - sort\_collating, [102](#)
  - special, [102](#)
- cob\_file\_key, [102](#)
  - field, [103](#)
  - flag, [103](#)
  - offset, [103](#)
- COB\_FILE\_MODE
  - fileio.h, [928](#)
- cob\_file\_release
  - fileio.c, [910](#)
- fileio.h, [948](#)
- cob\_file\_return
  - fileio.c, [910](#)
  - fileio.h, [948](#)
- cob\_file\_sort\_close
  - fileio.c, [911](#)
  - fileio.h, [949](#)
- cob\_file\_sort\_giving
  - fileio.c, [911](#)
  - fileio.h, [949](#)
- cob\_file\_sort\_init
  - fileio.c, [912](#)
  - fileio.h, [950](#)
- cob\_file\_sort\_init\_key
  - fileio.c, [913](#)
  - fileio.h, [951](#)
- cob\_file\_sort\_using
  - fileio.c, [913](#)
  - fileio.h, [951](#)
- COB\_FILE\_VERSION
  - fileio.h, [928](#)
- cob\_fileio\_funcs, [104](#)
  - close, [104](#)
  - fdelete, [104](#)
  - open, [104](#)
  - read, [104](#)
  - read\_next, [105](#)
  - rewrite, [105](#)
  - start, [105](#)
  - write, [105](#)
- COB\_FLAG\_BINARY\_SWAP
  - common.h, [882](#)
- COB\_FLAG\_BLANK\_ZERO
  - common.h, [882](#)
- COB\_FLAG\_HAVE\_SIGN
  - common.h, [882](#)
- COB\_FLAG\_IS\_POINTER
  - common.h, [882](#)
- COB\_FLAG\_JUSTIFIED
  - common.h, [883](#)
- COB\_FLAG\_REAL\_BINARY
  - common.h, [883](#)
- COB\_FLAG\_SIGN\_LEADING
  - common.h, [883](#)
- COB\_FLAG\_SIGN\_SEPARATE
  - common.h, [883](#)
- cob\_free\_alloc
  - common.h, [889](#)
- COB\_GE
  - fileio.h, [928](#)

- COB\_GEN\_SCREENIO
  - screenio.c, [1105](#)
- cob\_get\_environment
  - common.h, [889](#)
- cob\_get\_exception\_name
  - common.h, [889](#)
- cob\_get\_int
  - move.c, [1071](#)
  - move.h, [1076](#)
- cob\_get\_long\_long
  - coblocal.h, [856](#)
  - move.c, [1072](#)
- cob\_get\_numdisp
  - codegen.h, [868](#)
- cob\_get\_packed\_int
  - codegen.h, [868](#)
- cob\_get\_pointer
  - common.h, [889](#)
- cob\_get\_prog\_pointer
  - common.h, [889](#)
- cob\_get\_sign
  - common.h, [883](#)
- cob\_get\_switch
  - common.h, [889](#)
- cob\_got\_exception
  - coblocal.h, [861](#)
  - common.c, [873](#)
- COB\_GT
  - fileio.h, [928](#)
- COB\_HAS\_INLINE
  - config.h, [799](#)
- COB\_HIDDEN
  - coblocal.h, [855](#)
- cob\_high
  - common.c, [873](#)
  - common.h, [891](#)
- cob\_i2d
  - common.h, [883](#)
- cob\_init
  - common.h, [889](#)
- cob\_init\_call
  - call.c, [838](#)
  - coblocal.h, [856](#)
- cob\_init\_fileio
  - coblocal.h, [858](#)
  - fileio.c, [913](#)
- cob\_init\_intrinsic
  - coblocal.h, [859](#)
  - intrinsic.c, [963](#)
- cob\_init\_move
  - coblocal.h, [859](#)
  - move.c, [1072](#)
- cob\_init\_numeric
  - coblocal.h, [859](#)
  - numeric.c, [1090](#)
- cob\_init\_strings
  - coblocal.h, [860](#)
  - strings.c, [1126](#)
- cob\_init\_termio
  - coblocal.h, [860](#)
  - termio.c, [1145](#)
- cob\_initial\_external
  - common.c, [873](#)
  - common.h, [891](#)
- cob\_initialized
  - common.c, [874](#)
  - common.h, [891](#)
- COB\_INLINE
  - common.h, [883](#)
- COB\_INP\_SIZE
  - screenio.c, [1105](#)
- cob\_inp\_struct, [106](#)
  - down\_index, [107](#)
  - scr, [107](#)
  - this\_x, [107](#)
  - this\_y, [107](#)
  - up\_index, [107](#)
- cob\_inspect\_after
  - strings.c, [1126](#)
  - strings.h, [1135](#)
- cob\_inspect\_all
  - strings.c, [1126](#)
  - strings.h, [1136](#)
- cob\_inspect\_before
  - strings.c, [1126](#)
  - strings.h, [1136](#)
- cob\_inspect\_characters
  - strings.c, [1127](#)
  - strings.h, [1136](#)
- cob\_inspect\_converting
  - strings.c, [1127](#)
  - strings.h, [1137](#)
- cob\_inspect\_finish
  - strings.c, [1128](#)
  - strings.h, [1137](#)
- cob\_inspect\_first
  - strings.c, [1128](#)
  - strings.h, [1137](#)
- cob\_inspect\_init
  - strings.c, [1128](#)

- strings.h, 1138
- cob\_inspect\_leading
  - strings.c, 1129
  - strings.h, 1138
- cob\_inspect\_start
  - strings.c, 1129
  - strings.h, 1138
- cob\_inspect\_trailing
  - strings.c, 1129
  - strings.h, 1138
- cob\_intr\_abs
  - intrinsic.c, 963
  - intrinsic.h, 1018
- cob\_intr\_acos
  - intrinsic.c, 964
  - intrinsic.h, 1019
- cob\_intr\_annuity
  - intrinsic.c, 964
  - intrinsic.h, 1019
- cob\_intr\_asin
  - intrinsic.c, 965
  - intrinsic.h, 1020
- cob\_intr\_atan
  - intrinsic.c, 965
  - intrinsic.h, 1020
- cob\_intr\_binop
  - intrinsic.c, 966
  - intrinsic.h, 1021
- cob\_intr\_char
  - intrinsic.c, 967
  - intrinsic.h, 1022
- cob\_intr\_combined\_datetime
  - intrinsic.c, 968
  - intrinsic.h, 1022
- cob\_intr\_concatenate
  - intrinsic.c, 968
  - intrinsic.h, 1023
- cob\_intr\_cos
  - intrinsic.c, 969
  - intrinsic.h, 1024
- cob\_intr\_current\_date
  - intrinsic.c, 970
  - intrinsic.h, 1024
- cob\_intr\_date\_of\_integer
  - intrinsic.c, 971
  - intrinsic.h, 1026
- cob\_intr\_date\_to\_yyyymmdd
  - intrinsic.c, 972
  - intrinsic.h, 1027
- cob\_intr\_day\_of\_integer
  - intrinsic.c, 973
  - intrinsic.h, 1028
- cob\_intr\_day\_to\_yyyyddd
  - intrinsic.c, 974
  - intrinsic.h, 1029
- cob\_intr\_exception\_file
  - intrinsic.c, 975
  - intrinsic.h, 1030
- cob\_intr\_exception\_location
  - intrinsic.c, 975
  - intrinsic.h, 1030
- cob\_intr\_exception\_statement
  - intrinsic.c, 976
  - intrinsic.h, 1031
- cob\_intr\_exception\_status
  - intrinsic.c, 977
  - intrinsic.h, 1031
- cob\_intr\_exp
  - intrinsic.c, 977
  - intrinsic.h, 1032
- cob\_intr\_exp10
  - intrinsic.c, 977
  - intrinsic.h, 1032
- cob\_intr\_factorial
  - intrinsic.c, 978
  - intrinsic.h, 1033
- cob\_intr\_fraction\_part
  - intrinsic.c, 978
  - intrinsic.h, 1033
- cob\_intr\_integer
  - intrinsic.c, 979
  - intrinsic.h, 1033
- cob\_intr\_integer\_of\_date
  - intrinsic.c, 979
  - intrinsic.h, 1034
- cob\_intr\_integer\_of\_day
  - intrinsic.c, 980
  - intrinsic.h, 1035
- cob\_intr\_integer\_part
  - intrinsic.c, 981
  - intrinsic.h, 1036
- cob\_intr\_lcl\_time\_from\_secs
  - intrinsic.c, 981
  - intrinsic.h, 1036
- cob\_intr\_length
  - intrinsic.c, 983
  - intrinsic.h, 1038
- cob\_intr\_locale\_date
  - intrinsic.c, 984
  - intrinsic.h, 1039

- cob\_intr\_locale\_time
  - intrinsic.c, [986](#)
  - intrinsic.h, [1041](#)
- cob\_intr\_log
  - intrinsic.c, [989](#)
  - intrinsic.h, [1043](#)
- cob\_intr\_log10
  - intrinsic.c, [989](#)
  - intrinsic.h, [1044](#)
- cob\_intr\_lower\_case
  - intrinsic.c, [989](#)
  - intrinsic.h, [1044](#)
- cob\_intr\_max
  - intrinsic.c, [990](#)
  - intrinsic.h, [1044](#)
- cob\_intr\_mean
  - intrinsic.c, [990](#)
  - intrinsic.h, [1045](#)
- cob\_intr\_median
  - intrinsic.c, [991](#)
  - intrinsic.h, [1046](#)
- cob\_intr\_midrange
  - intrinsic.c, [992](#)
  - intrinsic.h, [1046](#)
- cob\_intr\_min
  - intrinsic.c, [992](#)
  - intrinsic.h, [1047](#)
- cob\_intr\_mod
  - intrinsic.c, [993](#)
  - intrinsic.h, [1047](#)
- cob\_intr\_numval
  - intrinsic.c, [993](#)
  - intrinsic.h, [1048](#)
- cob\_intr\_numval\_c
  - intrinsic.c, [995](#)
  - intrinsic.h, [1049](#)
- cob\_intr\_ord
  - intrinsic.c, [996](#)
  - intrinsic.h, [1051](#)
- cob\_intr\_ord\_max
  - intrinsic.c, [997](#)
  - intrinsic.h, [1051](#)
- cob\_intr\_ord\_min
  - intrinsic.c, [997](#)
  - intrinsic.h, [1052](#)
- cob\_intr\_present\_value
  - intrinsic.c, [998](#)
  - intrinsic.h, [1053](#)
- cob\_intr\_random
  - intrinsic.c, [999](#)
  - intrinsic.h, [1053](#)
- cob\_intr\_range
  - intrinsic.c, [1000](#)
  - intrinsic.h, [1054](#)
- cob\_intr\_rem
  - intrinsic.c, [1000](#)
  - intrinsic.h, [1055](#)
- cob\_intr\_reverse
  - intrinsic.c, [1001](#)
  - intrinsic.h, [1056](#)
- cob\_intr\_seconds\_from\_formatted\_time
  - intrinsic.c, [1001](#)
  - intrinsic.h, [1056](#)
- cob\_intr\_seconds\_past\_midnight
  - intrinsic.c, [1002](#)
  - intrinsic.h, [1057](#)
- cob\_intr\_sign
  - intrinsic.c, [1003](#)
  - intrinsic.h, [1057](#)
- cob\_intr\_sin
  - intrinsic.c, [1003](#)
  - intrinsic.h, [1058](#)
- cob\_intr\_sqrt
  - intrinsic.c, [1004](#)
  - intrinsic.h, [1059](#)
- cob\_intr\_standard\_deviation
  - intrinsic.c, [1004](#)
  - intrinsic.h, [1059](#)
- cob\_intr\_stored\_char\_length
  - intrinsic.c, [1005](#)
  - intrinsic.h, [1060](#)
- cob\_intr\_substitute
  - intrinsic.c, [1006](#)
  - intrinsic.h, [1060](#)
- cob\_intr\_substitute\_case
  - intrinsic.c, [1008](#)
  - intrinsic.h, [1062](#)
- cob\_intr\_sum
  - intrinsic.c, [1009](#)
  - intrinsic.h, [1064](#)
- cob\_intr\_tan
  - intrinsic.c, [1010](#)
  - intrinsic.h, [1064](#)
- cob\_intr\_test\_date\_yyyymmdd
  - intrinsic.c, [1010](#)
  - intrinsic.h, [1065](#)
- cob\_intr\_test\_day\_yyyydd
  - intrinsic.c, [1011](#)
  - intrinsic.h, [1066](#)
- cob\_intr\_trim



- intrinsic.c, [1012](#)
- intrinsic.h, [1066](#)
- cob\_intr\_upper\_case
  - intrinsic.c, [1012](#)
  - intrinsic.h, [1067](#)
- cob\_intr\_variance
  - intrinsic.c, [1013](#)
  - intrinsic.h, [1067](#)
- cob\_intr\_when\_compiled
  - intrinsic.c, [1014](#)
  - intrinsic.h, [1069](#)
- cob\_intr\_year\_to\_yyyy
  - intrinsic.c, [1014](#)
  - intrinsic.h, [1069](#)
- cob\_is\_alpha
  - common.h, [889](#)
- cob\_is\_lower
  - common.h, [889](#)
- cob\_is\_numeric
  - common.h, [889](#)
- cob\_is\_omitted
  - common.h, [889](#)
- cob\_is\_upper
  - common.h, [889](#)
- COB\_LARGE\_BUFF
  - common.h, [883](#)
- COB\_LARGE\_MAX
  - common.h, [883](#)
- COB\_LDFLAGS
  - defaults.h, [808](#)
- COB\_LE
  - fileio.h, [928](#)
- COB\_LI\_IS\_LL
  - config.h, [799](#)
- COB\_LIB\_INCLUDE
  - numeric.c, [1081](#)
- COB\_LIBRARY\_PATH
  - defaults.h, [808](#)
- COB\_LIBS
  - defaults.h, [808](#)
- COB\_LINAGE\_INVALID
  - fileio.h, [928](#)
- COB\_LOCK\_AUTOMATIC
  - fileio.h, [928](#)
- COB\_LOCK\_EXCLUSIVE
  - fileio.h, [928](#)
- COB\_LOCK\_MANUAL
  - fileio.h, [929](#)
- COB\_LOCK\_MASK
  - fileio.h, [929](#)
- COB\_LOCK\_MULTIPLE
  - fileio.h, [929](#)
- cob\_low
  - common.c, [874](#)
  - common.h, [891](#)
- COB\_LT
  - fileio.h, [929](#)
- cob\_malloc
  - common.h, [889](#)
- COB\_MAX\_BINARY
  - numeric.c, [1081](#)
- COB\_MAX\_COBCALL\_PARAMS
  - call.c, [836](#)
- COB\_MAX\_FIELD\_PARAMS
  - common.h, [883](#)
- COB\_MAX\_SUBSCRIPTS
  - codegen.c, [173](#)
  - tree.h, [576](#)
- COB\_MEDIUM\_BUFF
  - common.h, [884](#)
- COB\_MEDIUM\_MAX
  - common.h, [884](#)
- cob\_memcpy
  - coblocal.h, [860](#)
- COB\_MINI\_BUFF
  - common.h, [884](#)
- COB\_MINI\_MAX
  - common.h, [884](#)
- cob\_module, [107](#)
  - cob\_procedure\_parameters, [109](#)
  - collating\_sequence, [109](#)
  - crt\_status, [109](#)
  - currency\_symbol, [109](#)
  - cursor\_pos, [109](#)
  - decimal\_point, [109](#)
  - display\_sign, [109](#)
  - flag\_binary\_truncate, [109](#)
  - flag\_filename\_mapping, [109](#)
  - flag\_pretty\_display, [109](#)
  - next, [109](#)
  - numeric\_separator, [110](#)
  - spare8, [110](#)
- cob\_module\_enter
  - common.h, [890](#)
- COB\_MODULE\_EXT
  - defaults.h, [808](#)
- cob\_module\_leave
  - common.h, [890](#)
- cob\_move
  - move.c, [1072](#)

- move.h, 1076
- COB\_NE
  - fileio.h, 929
- COB\_NOINLINE
  - common.h, 884
- COB\_NON\_ALIGNED
  - cobc.h, 159
- COB\_NORMAL\_BUFF
  - common.h, 884
- COB\_NORMAL\_MAX
  - common.h, 884
- COB\_NOT\_CONFIGURED
  - fileio.h, 929
- COB\_NUM\_CSYSNS
  - cobc.c, 144
- cob\_numeric\_cmp
  - common.h, 890
  - numeric.c, 1090
- cob\_one
  - common.c, 874
  - common.h, 891
- cob\_open
  - fileio.c, 914
  - fileio.h, 951
- COB\_OPEN\_CLOSED
  - fileio.h, 929
- COB\_OPEN\_EXTEND
  - fileio.h, 929
- COB\_OPEN\_I\_O
  - fileio.h, 929
- COB\_OPEN\_INPUT
  - fileio.h, 929
- COB\_OPEN\_LOCKED
  - fileio.h, 930
- COB\_OPEN\_OUTPUT
  - fileio.h, 930
- COB\_ORG\_INDEXED
  - fileio.h, 930
- COB\_ORG\_LINE\_SEQUENTIAL
  - fileio.h, 930
- COB\_ORG\_MAX
  - fileio.h, 930
- COB\_ORG\_RELATIVE
  - fileio.h, 930
- COB\_ORG\_SEQUENTIAL
  - fileio.h, 930
- COB\_ORG\_SORT
  - fileio.h, 930
- cob\_orig\_line
  - coblocal.h, 861
- common.c, 874
- cob\_orig\_paragraph
  - coblocal.h, 861
  - common.c, 874
- cob\_orig\_program\_id
  - coblocal.h, 861
  - common.c, 874
- cob\_orig\_section
  - coblocal.h, 861
  - common.c, 874
- cob\_orig\_statement
  - coblocal.h, 861
  - common.c, 874
- cob\_parameter\_size
  - common.h, 890
- COB\_PIC\_FLAGS
  - config.h, 799
- cob\_pointer
  - cob\_alloc\_cache, 92
- cob\_procedure\_parameters
  - cob\_module, 109
- cob\_put\_sign
  - common.h, 884
- cob\_quote
  - common.c, 874
  - common.h, 891
- cob\_read
  - fileio.c, 919
  - fileio.h, 956
- COB\_READ\_FIRST
  - fileio.h, 930
- COB\_READ\_IGNORE\_LOCK
  - fileio.h, 930
- COB\_READ\_KEPT\_LOCK
  - fileio.h, 931
- COB\_READ\_LAST
  - fileio.h, 931
- COB\_READ\_LOCK
  - fileio.h, 931
- COB\_READ\_NEXT
  - fileio.h, 931
- COB\_READ\_NO\_LOCK
  - fileio.h, 931
- COB\_READ\_PREVIOUS
  - fileio.h, 931
- COB\_READ\_WAIT\_LOCK
  - fileio.h, 931
- cob\_ready\_trace
  - common.h, 890
- cob\_real\_get\_sign

- coblocal.h, [860](#)
- cob\_real\_put\_sign
  - coblocal.h, [860](#)
- cob\_reset\_trace
  - common.h, [890](#)
- cob\_resolve
  - call.c, [840](#)
  - call.h, [848](#)
- cob\_resolve\_1
  - call.c, [842](#)
  - call.h, [850](#)
- cob\_resolve\_error
  - call.c, [842](#)
  - call.h, [850](#)
- cob\_return\_args
  - common.h, [890](#)
- cob\_rewrite
  - fileio.c, [920](#)
  - fileio.h, [957](#)
- cob\_rollback
  - fileio.c, [921](#)
  - fileio.h, [958](#)
- cob\_runtime\_error
  - common.h, [890](#)
- cob\_save\_call\_params
  - common.c, [874](#)
  - common.h, [892](#)
- cob\_screen
  - screenio.h, [1117](#)
- cob\_screen\_accept
  - screenio.c, [1109](#)
  - screenio.h, [1121](#)
- COB\_SCREEN\_AUTO
  - screenio.h, [1114](#)
- COB\_SCREEN\_BELL
  - screenio.h, [1114](#)
- COB\_SCREEN\_BLACK
  - screenio.h, [1114](#)
- COB\_SCREEN\_BLANK\_LINE
  - screenio.h, [1114](#)
- COB\_SCREEN\_BLANK\_SCREEN
  - screenio.h, [1114](#)
- COB\_SCREEN\_BLINK
  - screenio.h, [1114](#)
- COB\_SCREEN\_BLUE
  - screenio.h, [1114](#)
- COB\_SCREEN\_COLUMN\_MINUS
  - screenio.h, [1114](#)
- COB\_SCREEN\_COLUMN\_PLUS
  - screenio.h, [1115](#)
- COB\_SCREEN\_CYAN
  - screenio.h, [1115](#)
- cob\_screen\_display
  - screenio.c, [1110](#)
  - screenio.h, [1122](#)
- COB\_SCREEN\_ERASE\_EOL
  - screenio.h, [1115](#)
- COB\_SCREEN\_ERASE\_EOS
  - screenio.h, [1115](#)
- COB\_SCREEN\_FULL
  - screenio.h, [1115](#)
- COB\_SCREEN\_GREEN
  - screenio.h, [1115](#)
- COB\_SCREEN\_HIGHLIGHT
  - screenio.h, [1115](#)
- cob\_screen\_initialized
  - coblocal.h, [861](#)
  - screenio.c, [1111](#)
- COB\_SCREEN\_INPUT
  - screenio.h, [1115](#)
- cob\_screen\_line\_col
  - screenio.c, [1111](#)
  - screenio.h, [1123](#)
- COB\_SCREEN\_LINE\_MINUS
  - screenio.h, [1115](#)
- COB\_SCREEN\_LINE\_PLUS
  - screenio.h, [1115](#)
- COB\_SCREEN\_LOWLIGHT
  - screenio.h, [1116](#)
- COB\_SCREEN\_MAGENTA
  - screenio.h, [1116](#)
- cob\_screen\_mode
  - screenio.c, [1111](#)
  - screenio.h, [1123](#)
- COB\_SCREEN\_OVERLINE
  - screenio.h, [1116](#)
- COB\_SCREEN\_PROMPT
  - screenio.h, [1116](#)
- COB\_SCREEN\_RED
  - screenio.h, [1116](#)
- COB\_SCREEN\_REQUIRED
  - screenio.h, [1116](#)
- COB\_SCREEN\_REVERSE
  - screenio.h, [1116](#)
- COB\_SCREEN\_SCROLL\_DOWN
  - screenio.h, [1116](#)
- COB\_SCREEN\_SECURE
  - screenio.h, [1116](#)
- cob\_screen\_set\_mode
  - coblocal.h, [860](#)

- screenio.c, [1111](#)
- cob\_screen\_terminate
  - coblocal.h, [860](#)
  - screenio.c, [1111](#)
- COB\_SCREEN\_TYPE\_ATTRIBUTE
  - screenio.h, [1116](#)
- COB\_SCREEN\_TYPE\_FIELD
  - screenio.h, [1117](#)
- COB\_SCREEN\_TYPE\_GROUP
  - screenio.h, [1117](#)
- COB\_SCREEN\_TYPE\_VALUE
  - screenio.h, [1117](#)
- COB\_SCREEN\_UNDERLINE
  - screenio.h, [1117](#)
- COB\_SCREEN\_UPDATE
  - screenio.h, [1117](#)
- COB\_SCREEN\_WHITE
  - screenio.h, [1117](#)
- COB\_SCREEN\_YELLOW
  - screenio.h, [1117](#)
- COB\_SELECT\_EXTERNAL
  - fileio.h, [931](#)
- COB\_SELECT\_FILE\_STATUS
  - fileio.h, [931](#)
- COB\_SELECT\_LINAGE
  - fileio.h, [931](#)
- COB\_SELECT\_SPLITKEY
  - fileio.h, [932](#)
- cob\_set\_cancel
  - call.c, [842](#)
  - call.h, [851](#)
- cob\_set\_environment
  - common.h, [890](#)
- cob\_set\_exception
  - common.h, [890](#)
- cob\_set\_int
  - move.c, [1075](#)
  - move.h, [1079](#)
- cob\_set\_location
  - common.h, [890](#)
- cob\_set\_packed\_int
  - numeric.c, [1090](#)
  - numeric.h, [1102](#)
- cob\_set\_packed\_zero
  - numeric.c, [1091](#)
  - numeric.h, [1103](#)
- cob\_set\_switch
  - common.h, [890](#)
- cob\_setswp\_s16\_binary
  - codegen.h, [868](#)
- cob\_setswp\_s24\_binary
  - codegen.h, [868](#)
- cob\_setswp\_s32\_binary
  - codegen.h, [869](#)
- cob\_setswp\_s40\_binary
  - codegen.h, [869](#)
- cob\_setswp\_s48\_binary
  - codegen.h, [869](#)
- cob\_setswp\_s56\_binary
  - codegen.h, [869](#)
- cob\_setswp\_s64\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u16\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u24\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u32\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u40\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u48\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u56\_binary
  - codegen.h, [869](#)
- cob\_setswp\_u64\_binary
  - codegen.h, [869](#)
- COB\_SHARED\_OPT
  - config.h, [799](#)
- cob\_sighandler\_t
  - cobc.c, [144](#)
  - common.c, [873](#)
- COB\_SMALL\_BUFF
  - common.h, [884](#)
- COB\_SMALL\_MAX
  - common.h, [884](#)
- cob\_space
  - common.c, [875](#)
  - common.h, [892](#)
- COB\_STACK\_SIZE
  - common.h, [885](#)
- cob\_start
  - fileio.c, [921](#)
  - fileio.h, [958](#)
- COB\_STATIC
  - codegen.h, [865](#)
- COB\_STATUS\_00\_SUCCESS
  - fileio.h, [932](#)
- COB\_STATUS\_02\_SUCCESS\_DUPLICATE
  - fileio.h, [932](#)
- COB\_STATUS\_04\_SUCCESS\_INCOMPLETE

fileio.h, <a href="#">932</a>	fileio.h, <a href="#">934</a>
COB_STATUS_05_SUCCESS_OPTIONAL fileio.h, <a href="#">932</a>	COB_STATUS_61_FILE_SHARING fileio.h, <a href="#">934</a>
COB_STATUS_07_SUCCESS_NO_UNIT fileio.h, <a href="#">932</a>	COB_STATUS_91_NOT_AVAILABLE fileio.h, <a href="#">935</a>
COB_STATUS_10_END_OF_FILE fileio.h, <a href="#">932</a>	cob_stop_run common.h, <a href="#">890</a>
COB_STATUS_14_OUT_OF_KEY_RANGE fileio.h, <a href="#">932</a>	COB_STORE_KEEP_ON_OVERFLOW numeric.h, <a href="#">1093</a>
COB_STATUS_21_KEY_INVALID fileio.h, <a href="#">932</a>	COB_STORE_ROUND numeric.h, <a href="#">1093</a>
COB_STATUS_22_KEY_EXISTS fileio.h, <a href="#">932</a>	COB_STORE_TRUNC_ON_OVERFLOW numeric.h, <a href="#">1094</a>
COB_STATUS_23_KEY_NOT_EXISTS fileio.h, <a href="#">933</a>	COB_STRFTIME config.h, <a href="#">799</a>
COB_STATUS_30_PERMANENT_ERROR fileio.h, <a href="#">933</a>	cob_string_append strings.c, <a href="#">1129</a>
COB_STATUS_31_INCONSISTENT_FILENAME fileio.h, <a href="#">933</a>	strings.h, <a href="#">1139</a>
COB_STATUS_34_BOUNDARY_VIOLATION fileio.h, <a href="#">933</a>	cob_string_delimited strings.c, <a href="#">1130</a>
COB_STATUS_35_NOT_EXISTS fileio.h, <a href="#">933</a>	strings.h, <a href="#">1139</a>
COB_STATUS_37_PERMISSION_DENIED fileio.h, <a href="#">933</a>	cob_string_finish strings.c, <a href="#">1130</a>
COB_STATUS_38_CLOSED_WITH_LOCK fileio.h, <a href="#">933</a>	strings.h, <a href="#">1139</a>
COB_STATUS_39_CONFLICT_ATTRIBUTE fileio.h, <a href="#">933</a>	cob_string_init strings.c, <a href="#">1130</a>
COB_STATUS_41_ALREADY_OPEN fileio.h, <a href="#">933</a>	strings.h, <a href="#">1140</a>
COB_STATUS_42_NOT_OPEN fileio.h, <a href="#">933</a>	COB_STRIP_CMD config.h, <a href="#">799</a>
COB_STATUS_43_READ_NOT_DONE fileio.h, <a href="#">934</a>	cob_sub numeric.c, <a href="#">1091</a>
COB_STATUS_44_RECORD_OVERFLOW fileio.h, <a href="#">934</a>	numeric.h, <a href="#">1103</a>
COB_STATUS_46_READ_ERROR fileio.h, <a href="#">934</a>	cob_sub_align_s16_binary codegen.h, <a href="#">869</a>
COB_STATUS_47_INPUT_DENIED fileio.h, <a href="#">934</a>	cob_sub_align_s32_binary codegen.h, <a href="#">869</a>
COB_STATUS_48_OUTPUT_DENIED fileio.h, <a href="#">934</a>	cob_sub_align_s64_binary codegen.h, <a href="#">869</a>
COB_STATUS_49_I_O_DENIED fileio.h, <a href="#">934</a>	cob_sub_align_u16_binary codegen.h, <a href="#">869</a>
COB_STATUS_51_RECORD_LOCKED fileio.h, <a href="#">934</a>	cob_sub_align_u32_binary codegen.h, <a href="#">869</a>
COB_STATUS_52_EOP fileio.h, <a href="#">934</a>	cob_sub_align_u64_binary codegen.h, <a href="#">869</a>
COB_STATUS_57_I_O_LINAGE	cob_sub_int numeric.c, <a href="#">1091</a>
	numeric.h, <a href="#">1103</a>
	cob_sub_s16_binary codegen.h, <a href="#">869</a>
	cob_sub_s24_binary

- codegen.h, [869](#)
- cob\_sub\_s32\_binary
  - codegen.h, [869](#)
- cob\_sub\_s40\_binary
  - codegen.h, [869](#)
- cob\_sub\_s48\_binary
  - codegen.h, [869](#)
- cob\_sub\_s56\_binary
  - codegen.h, [869](#)
- cob\_sub\_s64\_binary
  - codegen.h, [870](#)
- cob\_sub\_s8\_binary
  - codegen.h, [870](#)
- cob\_sub\_u16\_binary
  - codegen.h, [870](#)
- cob\_sub\_u24\_binary
  - codegen.h, [870](#)
- cob\_sub\_u32\_binary
  - codegen.h, [870](#)
- cob\_sub\_u40\_binary
  - codegen.h, [870](#)
- cob\_sub\_u48\_binary
  - codegen.h, [870](#)
- cob\_sub\_u56\_binary
  - codegen.h, [870](#)
- cob\_sub\_u64\_binary
  - codegen.h, [870](#)
- cob\_sub\_u8\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s16\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s24\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s32\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s40\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s48\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s56\_binary
  - codegen.h, [870](#)
- cob\_subswp\_s64\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u16\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u24\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u32\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u40\_binary
  - codegen.h, [870](#)
- codegen.h, [870](#)
- cob\_subswp\_u48\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u56\_binary
  - codegen.h, [870](#)
- cob\_subswp\_u64\_binary
  - codegen.h, [870](#)
- COB\_SYSTEM\_GEN
  - call.c, [836](#)
  - codegen.c, [173](#)
  - typeck.c, [726](#)
- cob\_table\_sort
  - common.h, [890](#)
- cob\_table\_sort\_init
  - common.h, [890](#)
- cob\_table\_sort\_init\_key
  - common.h, [890](#)
- COB\_TYPE\_ALPHANUMERIC
  - common.h, [885](#)
- COB\_TYPE\_ALPHANUMERIC\_ALL
  - common.h, [885](#)
- COB\_TYPE\_ALPHANUMERIC\_EDITED
  - common.h, [885](#)
- COB\_TYPE\_BOOLEAN
  - common.h, [885](#)
- COB\_TYPE\_GROUP
  - common.h, [885](#)
- COB\_TYPE\_NATIONAL
  - common.h, [885](#)
- COB\_TYPE\_NATIONAL\_EDITED
  - common.h, [885](#)
- COB\_TYPE\_NUMERIC
  - common.h, [885](#)
- COB\_TYPE\_NUMERIC\_BINARY
  - common.h, [885](#)
- COB\_TYPE\_NUMERIC\_DISPLAY
  - common.h, [886](#)
- COB\_TYPE\_NUMERIC\_DOUBLE
  - common.h, [886](#)
- COB\_TYPE\_NUMERIC\_EDITED
  - common.h, [886](#)
- COB\_TYPE\_NUMERIC\_FLOAT
  - common.h, [886](#)
- COB\_TYPE\_NUMERIC\_PACKED
  - common.h, [886](#)
- COB\_TYPE\_UNKNOWN
  - common.h, [886](#)
- cob\_unlock\_file
  - fileio.c, [921](#)
  - fileio.h, [959](#)

- cob\_unstring\_delimited
  - strings.c, [1131](#)
  - strings.h, [1140](#)
- cob\_unstring\_finish
  - strings.c, [1131](#)
  - strings.h, [1140](#)
- cob\_unstring\_init
  - strings.c, [1131](#)
  - strings.h, [1141](#)
- cob\_unstring\_into
  - strings.c, [1132](#)
  - strings.h, [1141](#)
- cob\_unstring\_tallying
  - strings.c, [1134](#)
  - strings.h, [1143](#)
- COB\_USE\_SETJMP
  - codegen.c, [173](#)
- cob\_write
  - fileio.c, [922](#)
  - fileio.h, [959](#)
- COB\_WRITE\_AFTER
  - fileio.h, [935](#)
- COB\_WRITE\_BEFORE
  - fileio.h, [935](#)
- COB\_WRITE\_CHANNEL
  - fileio.h, [935](#)
- COB\_WRITE\_EOP
  - fileio.h, [935](#)
- COB\_WRITE\_LINES
  - fileio.h, [935](#)
- COB\_WRITE\_LOCK
  - fileio.h, [935](#)
- COB\_WRITE\_MASK
  - fileio.h, [935](#)
- COB\_WRITE\_PAGE
  - fileio.h, [935](#)
- cob\_zero
  - common.c, [875](#)
  - common.h, [892](#)
- cobc.c
  - alt\_ebcdic, [151](#)
  - CB\_LEVEL\_ASSEMBLE, [145](#)
  - CB\_LEVEL\_COMPILE, [145](#)
  - CB\_LEVEL\_EXECUTABLE, [145](#)
  - CB\_LEVEL\_LIBRARY, [145](#)
  - CB\_LEVEL\_MODULE, [145](#)
  - CB\_LEVEL\_PREPROCESS, [145](#)
  - CB\_LEVEL\_TRANSLATE, [145](#)
  - cb\_attr\_id, [151](#)
  - cb\_compile\_level, [144](#)
  - cb\_depend\_file, [152](#)
  - cb\_depend\_list, [152](#)
  - cb\_depend\_target, [152](#)
  - cb\_display\_sign, [152](#)
  - cb\_exception\_table, [152](#)
  - cb\_extension\_list, [152](#)
  - cb\_field\_id, [152](#)
  - CB\_FLAG, [143](#)
  - cb\_flag\_main, [152](#)
  - cb\_id, [152](#)
  - cb\_include\_list, [152](#)
  - cb\_listing\_file, [153](#)
  - cb\_literal\_id, [153](#)
  - cb\_oc\_build\_stamp, [153](#)
  - cb\_saveargc, [153](#)
  - cb\_saveargv, [153](#)
  - cb\_source\_file, [153](#)
  - cb\_source\_format, [153](#)
  - cb\_source\_line, [153](#)
  - cb\_storage\_file, [153](#)
  - cb\_storage\_file\_name, [153](#)
  - cb\_storage\_id, [154](#)
  - cb\_text\_list\_add, [145](#)
  - CB\_WARNDEF, [143](#), [144](#)
  - cob\_config\_dir, [154](#)
  - COB\_EXCEPTION, [144](#)
  - COB\_NUM\_CSYSNS, [144](#)
  - cob\_sighandler\_t, [144](#)
  - cobc\_abort, [145](#)
  - cobc\_check\_valid\_name, [146](#)
  - cobc\_malloc, [146](#)
  - cobc\_realloc, [146](#)
  - cobc\_tree\_cast\_error, [146](#)
  - demangle\_name, [154](#)
  - errorcount, [154](#)
  - main, [147](#)
  - optimize\_flag, [154](#)
  - PATHSEPS, [144](#)
  - source\_name, [154](#)
  - warningcount, [154](#)
- cobc.h
  - ABORT, [157](#)
  - alt\_ebcdic, [167](#)
  - CB\_ARCHAIC, [160](#)
  - CB\_ASSIGN\_COBOL2002, [159](#)
  - CB\_ASSIGN\_IBM, [159](#)
  - CB\_ASSIGN\_MF, [159](#)
  - CB\_BINARY\_SIZE\_1\_2\_4\_8, [159](#)
  - CB\_BINARY\_SIZE\_1\_8, [159](#)
  - CB\_BINARY\_SIZE\_2\_4\_8, [159](#)

- CB\_BYTEORDER\_BIG\_ENDIAN, 159
- CB\_BYTEORDER\_NATIVE, 159
- CB\_ERROR, 160
- CB\_IGNORE, 160
- CB\_OBSOLETE, 160
- CB\_OK, 160
- CB\_OPERATION\_ASSIGN, 160
- CB\_OPERATION\_READ, 160
- CB\_OPERATION\_WRITE, 160
- CB\_SKIP, 160
- CB\_UNCONFORMABLE, 160
- CB\_WARNING, 160
- cb\_assign\_clause, 159
- cb\_attr\_id, 168
- cb\_binary\_byteorder, 159
- cb\_binary\_size, 159
- CB\_CONFIG\_ANY, 157
- CB\_CONFIG\_BOOLEAN, 157
- CB\_CONFIG\_INT, 158
- CB\_CONFIG\_STRING, 158
- CB\_CONFIG\_SUPPORT, 158
- cb\_depend\_file, 168
- cb\_depend\_list, 168
- cb\_depend\_target, 168
- cb\_display\_sign, 168
- cb\_error, 161
- CB\_EXCEPTION\_CODE, 158
- CB\_EXCEPTION\_ENABLE, 158
- CB\_EXCEPTION\_NAME, 158
- cb\_exception\_table, 168
- cb\_extension\_list, 168
- cb\_field\_id, 168
- CB\_FLAG, 158
- cb\_flag\_main, 168
- CB\_FORMAT\_FIXED, 158
- CB\_FORMAT\_FREE, 158
- cb\_id, 168
- cb\_include\_list, 168
- cb\_listing\_file, 169
- cb\_literal\_id, 169
- cb\_load\_conf, 161
- cb\_load\_std, 164
- cb\_oc\_build\_stamp, 169
- cb\_operation\_type, 160
- cb\_saveargc, 169
- cb\_saveargv, 169
- cb\_source\_file, 169
- cb\_source\_format, 169
- cb\_source\_line, 169
- cb\_storage\_file, 169
- cb\_storage\_file\_name, 169
- cb\_storage\_id, 170
- cb\_support, 160
- cb\_text\_list\_add, 165
- cb\_verify, 165
- CB\_WARNDEF, 158
- cb\_warning, 166
- cob\_config\_dir, 170
- COB\_NON\_ALIGNED, 159
- cobc\_abort, 166
- cobc\_check\_valid\_name, 166
- cobc\_malloc, 166
- cobc\_realloc, 167
- current\_paragraph, 170
- current\_program, 170
- current\_section, 170
- current\_statement, 170
- demangle\_name, 170
- errorcount, 170
- functions\_are\_all, 170
- has\_external, 170
- norestab, 171
- optimize\_flag, 171
- pp\_set\_replace\_list, 167
- ppcopy, 167
- ppin, 171
- pplex, 167
- ppopen, 167
- ppout, 171
- ppparse, 167
- sending\_id, 171
- source\_name, 171
- suppress\_warn, 171
- warningcount, 171
- yyin, 171
- yylex, 167
- yyout, 171
- yyparse, 167
- cobc/ Directory Reference, 10
- cobc/cobc.c, 140
- cobc/cobc.h, 154
- cobc/codegen.c, 171
- cobc/config.c, 184
- cobc/error.c, 191
- cobc/field.c, 197
- cobc/parser.c, 204
- cobc/parser.h, 309
- cobc/pplex.c, 401
- cobc/ppparse.c, 414
- cobc/ppparse.h, 458



- cobc/reserved.c, [492](#)
- cobc/scanner.c, [496](#)
- cobc/tree.c, [508](#)
- cobc/tree.h, [551](#)
- cobc/typeck.c, [722](#)
- cobc\_abort
  - cobc.c, [145](#)
  - cobc.h, [166](#)
- cobc\_check\_valid\_name
  - cobc.c, [146](#)
  - cobc.h, [166](#)
- cobc\_malloc
  - cobc.c, [146](#)
  - cobc.h, [166](#)
- cobc\_realloc
  - cobc.c, [146](#)
  - cobc.h, [167](#)
- cobc\_tree\_cast\_error
  - cobc.c, [146](#)
  - tree.h, [695](#)
- cobcall
  - call.c, [843](#)
  - call.h, [851](#)
- cobcancel
  - call.c, [843](#)
  - call.h, [852](#)
- cobcommandline
  - common.h, [890](#)
- cobcrun.c
  - main, [139](#)
- cobexit
  - common.h, [890](#)
- cobfunc
  - call.c, [844](#)
  - call.h, [852](#)
- cobgetenv
  - common.h, [890](#)
- cobinit
  - common.h, [891](#)
- cobitem, [110](#)
  - block\_byte, [111](#)
  - end\_of\_block, [111](#)
  - item, [111](#)
  - next, [111](#)
  - unique, [111](#)
- cobjmp\_buf, [111](#)
  - cbj\_int, [112](#)
  - cbj\_jmp\_buf, [112](#)
  - cbj\_ptr, [112](#)
  - cbj\_ptr\_rest, [112](#)
- coblocal.h
  - COB\_ATTR\_INIT, [855](#)
  - COB\_CHK\_PARAMS, [855](#)
  - cob\_exit\_fileio, [855](#)
  - cob\_field\_to\_string, [856](#)
  - cob\_get\_long\_long, [856](#)
  - cob\_got\_exception, [861](#)
  - COB\_HIDDEN, [855](#)
  - cob\_init\_call, [856](#)
  - cob\_init\_fileio, [858](#)
  - cob\_init\_intrinsic, [859](#)
  - cob\_init\_move, [859](#)
  - cob\_init\_numeric, [859](#)
  - cob\_init\_strings, [860](#)
  - cob\_init\_termio, [860](#)
  - cob\_memcpy, [860](#)
  - cob\_orig\_line, [861](#)
  - cob\_orig\_paragraph, [861](#)
  - cob\_orig\_program\_id, [861](#)
  - cob\_orig\_section, [861](#)
  - cob\_orig\_statement, [861](#)
  - cob\_real\_get\_sign, [860](#)
  - cob\_real\_put\_sign, [860](#)
  - cob\_screen\_initialized, [861](#)
  - cob\_screen\_set\_mode, [860](#)
  - cob\_screen\_terminate, [860](#)
- coblongjmp
  - call.c, [844](#)
  - call.h, [853](#)
- cobputenv
  - common.h, [891](#)
- cobsavenv
  - call.c, [845](#)
  - call.h, [853](#)
- cobsavenv2
  - call.c, [845](#)
  - call.h, [853](#)
- cobsetjmp
  - call.h, [847](#)
- cobsort, [113](#)
  - destination\_file, [114](#)
  - empty, [114](#)
  - file, [114](#)
  - files\_used, [114](#)
  - fnstatus, [114](#)
  - memory, [114](#)
  - pointer, [114](#)
  - queue, [114](#)
  - r\_size, [114](#)
  - retrieval\_queue, [114](#)

- retrieving, [115](#)
- size, [115](#)
- sort\_return, [115](#)
- unique, [115](#)
- w\_size, [115](#)
- COBSORTABORT
  - fileio.c, [895](#)
- COBSORTEND
  - fileio.c, [895](#)
- COBSORTFILEERR
  - fileio.c, [895](#)
- COBSORTNOTOPEN
  - fileio.c, [895](#)
- cobtidy
  - common.h, [891](#)
- CODE
  - parser.c, [229](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- code
  - cb\_exception, [29](#)
  - cob\_exception, [94](#)
- CODE\_SET
  - parser.c, [278](#), [289](#)
  - parser.h, [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- CODE\_SET
  - parser.c, [229](#)
  - parser.h, [332](#)
- codegen
  - codegen.c, [174](#)
  - tree.h, [695](#)
- codegen.c
  - COB\_MAX\_SUBSCRIPTS, [173](#)
  - COB\_SYSTEM\_GEN, [173](#)
  - COB\_USE\_SETJMP, [173](#)
  - codegen, [174](#)
  - has\_external, [184](#)
  - INITIALIZE\_COMPOUND, [173](#)
  - INITIALIZE\_DEFAULT, [173](#)
  - INITIALIZE\_EXTERNAL, [173](#)
  - INITIALIZE\_NONE, [173](#)
  - INITIALIZE\_ONE, [173](#)
- codegen.h
  - cob\_add\_align\_s16\_binary, [865](#)
  - cob\_add\_align\_s32\_binary, [865](#)
  - cob\_add\_align\_s64\_binary, [865](#)
  - cob\_add\_align\_u16\_binary, [865](#)
  - cob\_add\_align\_u32\_binary, [865](#)
  - cob\_add\_align\_u64\_binary, [865](#)
  - cob\_add\_packed\_int, [865](#)
  - cob\_add\_s16\_binary, [865](#)
  - cob\_add\_s24\_binary, [865](#)
  - cob\_add\_s32\_binary, [865](#)
  - cob\_add\_s40\_binary, [865](#)
  - cob\_add\_s48\_binary, [865](#)
  - cob\_add\_s56\_binary, [866](#)
  - cob\_add\_s64\_binary, [866](#)
  - cob\_add\_s8\_binary, [866](#)
  - cob\_add\_u16\_binary, [866](#)
  - cob\_add\_u24\_binary, [866](#)
  - cob\_add\_u32\_binary, [866](#)
  - cob\_add\_u40\_binary, [866](#)
  - cob\_add\_u48\_binary, [866](#)
  - cob\_add\_u56\_binary, [866](#)
  - cob\_add\_u64\_binary, [866](#)
  - cob\_add\_u8\_binary, [866](#)
  - cob\_addswp\_s16\_binary, [866](#)
  - cob\_addswp\_s24\_binary, [866](#)
  - cob\_addswp\_s32\_binary, [866](#)
  - cob\_addswp\_s40\_binary, [866](#)
  - cob\_addswp\_s48\_binary, [866](#)
  - cob\_addswp\_s56\_binary, [866](#)
  - cob\_addswp\_s64\_binary, [866](#)
  - cob\_addswp\_u16\_binary, [866](#)
  - cob\_addswp\_u24\_binary, [866](#)
  - cob\_addswp\_u32\_binary, [866](#)
  - cob\_addswp\_u40\_binary, [866](#)
  - cob\_addswp\_u48\_binary, [866](#)
  - cob\_addswp\_u56\_binary, [866](#)
  - cob\_addswp\_u64\_binary, [867](#)
  - cob\_cmp\_align\_s16\_binary, [867](#)
  - cob\_cmp\_align\_s32\_binary, [867](#)
  - cob\_cmp\_align\_s64\_binary, [867](#)
  - cob\_cmp\_align\_u16\_binary, [867](#)
  - cob\_cmp\_align\_u32\_binary, [867](#)
  - cob\_cmp\_align\_u64\_binary, [867](#)
  - cob\_cmp\_packed\_int, [867](#)
  - cob\_cmp\_s16\_binary, [867](#)
  - cob\_cmp\_s24\_binary, [867](#)
  - cob\_cmp\_s32\_binary, [867](#)
  - cob\_cmp\_s40\_binary, [867](#)
  - cob\_cmp\_s48\_binary, [867](#)
  - cob\_cmp\_s56\_binary, [867](#)
  - cob\_cmp\_s64\_binary, [867](#)
  - cob\_cmp\_s8\_binary, [867](#)
  - cob\_cmp\_u16\_binary, [867](#)
  - cob\_cmp\_u24\_binary, [867](#)

- cob\_cmp\_u32\_binary, [867](#)
  - cob\_cmp\_u40\_binary, [867](#)
  - cob\_cmp\_u48\_binary, [867](#)
  - cob\_cmp\_u56\_binary, [867](#)
  - cob\_cmp\_u64\_binary, [867](#)
  - cob\_cmp\_u8\_binary, [867](#)
  - cob\_cmpswp\_align\_s16\_binary, [868](#)
  - cob\_cmpswp\_align\_s32\_binary, [868](#)
  - cob\_cmpswp\_align\_s64\_binary, [868](#)
  - cob\_cmpswp\_align\_u16\_binary, [868](#)
  - cob\_cmpswp\_align\_u32\_binary, [868](#)
  - cob\_cmpswp\_align\_u64\_binary, [868](#)
  - cob\_cmpswp\_s16\_binary, [868](#)
  - cob\_cmpswp\_s24\_binary, [868](#)
  - cob\_cmpswp\_s32\_binary, [868](#)
  - cob\_cmpswp\_s40\_binary, [868](#)
  - cob\_cmpswp\_s48\_binary, [868](#)
  - cob\_cmpswp\_s56\_binary, [868](#)
  - cob\_cmpswp\_s64\_binary, [868](#)
  - cob\_cmpswp\_u16\_binary, [868](#)
  - cob\_cmpswp\_u24\_binary, [868](#)
  - cob\_cmpswp\_u32\_binary, [868](#)
  - cob\_cmpswp\_u40\_binary, [868](#)
  - cob\_cmpswp\_u48\_binary, [868](#)
  - cob\_cmpswp\_u56\_binary, [868](#)
  - cob\_cmpswp\_u64\_binary, [868](#)
  - cob\_get\_numdisp, [868](#)
  - cob\_get\_packed\_int, [868](#)
  - cob\_setswp\_s16\_binary, [868](#)
  - cob\_setswp\_s24\_binary, [868](#)
  - cob\_setswp\_s32\_binary, [869](#)
  - cob\_setswp\_s40\_binary, [869](#)
  - cob\_setswp\_s48\_binary, [869](#)
  - cob\_setswp\_s56\_binary, [869](#)
  - cob\_setswp\_s64\_binary, [869](#)
  - cob\_setswp\_u16\_binary, [869](#)
  - cob\_setswp\_u24\_binary, [869](#)
  - cob\_setswp\_u32\_binary, [869](#)
  - cob\_setswp\_u40\_binary, [869](#)
  - cob\_setswp\_u48\_binary, [869](#)
  - cob\_setswp\_u56\_binary, [869](#)
  - cob\_setswp\_u64\_binary, [869](#)
  - COB\_STATIC, [865](#)
  - cob\_sub\_align\_s16\_binary, [869](#)
  - cob\_sub\_align\_s32\_binary, [869](#)
  - cob\_sub\_align\_s64\_binary, [869](#)
  - cob\_sub\_align\_u16\_binary, [869](#)
  - cob\_sub\_align\_u32\_binary, [869](#)
  - cob\_sub\_align\_u64\_binary, [869](#)
  - cob\_sub\_s16\_binary, [869](#)
  - cob\_sub\_s24\_binary, [869](#)
  - cob\_sub\_s32\_binary, [869](#)
  - cob\_sub\_s40\_binary, [869](#)
  - cob\_sub\_s48\_binary, [869](#)
  - cob\_sub\_s56\_binary, [869](#)
  - cob\_sub\_s64\_binary, [870](#)
  - cob\_sub\_s8\_binary, [870](#)
  - cob\_sub\_u16\_binary, [870](#)
  - cob\_sub\_u24\_binary, [870](#)
  - cob\_sub\_u32\_binary, [870](#)
  - cob\_sub\_u40\_binary, [870](#)
  - cob\_sub\_u48\_binary, [870](#)
  - cob\_sub\_u56\_binary, [870](#)
  - cob\_sub\_u64\_binary, [870](#)
  - cob\_sub\_u8\_binary, [870](#)
  - cob\_subswp\_s16\_binary, [870](#)
  - cob\_subswp\_s24\_binary, [870](#)
  - cob\_subswp\_s32\_binary, [870](#)
  - cob\_subswp\_s40\_binary, [870](#)
  - cob\_subswp\_s48\_binary, [870](#)
  - cob\_subswp\_s56\_binary, [870](#)
  - cob\_subswp\_s64\_binary, [870](#)
  - cob\_subswp\_u16\_binary, [870](#)
  - cob\_subswp\_u24\_binary, [870](#)
  - cob\_subswp\_u32\_binary, [870](#)
  - cob\_subswp\_u40\_binary, [870](#)
  - cob\_subswp\_u48\_binary, [870](#)
  - cob\_subswp\_u56\_binary, [870](#)
  - cob\_subswp\_u64\_binary, [870](#)
- COL
- parser.c, [229](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COLLATING
- parser.c, [229](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- collating\_sequence
- cb\_program, [73](#)
  - cob\_module, [109](#)
- COLS
- parser.c, [229](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COLUMN
- parser.c, [230](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)

- ppparse.c, [434](#), [446](#)
- ppparse.h, [469](#), [481](#)
- column
  - \_\_cob\_screen, [14](#)
- COLUMNS
  - parser.c, [230](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COMMA
  - parser.c, [230](#), [278](#), [289](#)
  - parser.h, [332](#), [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COMMA\_DELIM
  - parser.c, [278](#), [290](#)
  - parser.h, [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COMMA\_DELIM
  - parser.c, [230](#)
  - parser.h, [332](#)
- COMMAND\_LINE
  - parser.c, [278](#), [289](#)
  - parser.h, [372](#), [383](#)
  - ppparse.c, [434](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COMMAND\_LINE
  - parser.c, [230](#)
  - parser.h, [332](#)
- COMMIT
  - parser.c, [230](#), [278](#), [290](#)
  - parser.h, [333](#), [372](#), [383](#)
  - ppparse.c, [435](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- COMMON
  - parser.c, [230](#), [278](#), [290](#)
  - parser.h, [333](#), [372](#), [383](#)
  - ppparse.c, [435](#), [446](#)
  - ppparse.h, [469](#), [481](#)
- common
  - cb\_alphabet\_name, [17](#)
  - cb\_assign, [20](#)
  - cb\_binary\_op, [21](#)
  - cb\_call, [23](#)
  - cb\_cast, [24](#)
  - cb\_class\_name, [26](#)
  - cb\_const, [27](#)
  - cb\_continue, [28](#)
  - cb\_decimal, [29](#)
  - cb\_field, [32](#)
  - cb\_file, [42](#)
  - cb\_funcall, [45](#)
  - cb\_goto, [47](#)
  - cb\_if, [48](#)
  - cb\_initialize, [49](#)
  - cb\_integer, [51](#)
  - cb\_intrinsic, [52](#)
  - cb\_label, [57](#)
  - cb\_list, [61](#)
  - cb\_literal, [62](#)
  - cb\_locale\_name, [64](#)
  - cb\_perform, [65](#)
  - cb\_perform\_varying, [67](#)
  - cb\_picture, [69](#)
  - cb\_reference, [80](#)
  - cb\_search, [83](#)
  - cb\_statement, [85](#)
  - cb\_string, [87](#)
  - cb\_system\_name, [88](#)
- common.c
  - cob\_call\_params, [873](#)
  - cob\_current\_module, [873](#)
  - COB\_ERRBUF\_SIZE, [872](#)
  - COB\_EXCEPTION, [872](#), [873](#)
  - cob\_exception\_code, [873](#)
  - cob\_got\_exception, [873](#)
  - cob\_high, [873](#)
  - cob\_initial\_external, [873](#)
  - cob\_initialized, [874](#)
  - cob\_low, [874](#)
  - cob\_one, [874](#)
  - cob\_orig\_line, [874](#)
  - cob\_orig\_paragraph, [874](#)
  - cob\_orig\_program\_id, [874](#)
  - cob\_orig\_section, [874](#)
  - cob\_orig\_statement, [874](#)
  - cob\_quote, [874](#)
  - cob\_save\_call\_params, [874](#)
  - cob\_sighandler\_t, [873](#)
  - cob\_space, [875](#)
  - cob\_zero, [875](#)
  - EXCEPTION\_TAB\_SIZE, [873](#)
- common.h
  - CBL\_AND, [887](#)
  - CBL\_EQ, [887](#)
  - CBL\_ERROR\_PROC, [887](#)
  - CBL\_EXIT\_PROC, [887](#)
  - CBL\_IMP, [887](#)
  - CBL\_NIMP, [887](#)

- CBL\_NOR, 887
- CBL\_NOT, 887
- CBL\_OC\_NANOSLEEP, 888
- CBL\_OR, 888
- CBL\_TOLOWER, 888
- CBL\_TOUPPER, 888
- CBL\_X91, 888
- CBL\_XF4, 888
- CBL\_XF5, 888
- CBL\_XOR, 888
- COB\_EC\_MAX, 887
- COB\_EC\_ZERO, 887
- cob\_accept\_arg\_number, 888
- cob\_accept\_arg\_value, 888
- cob\_accept\_command\_line, 888
- cob\_accept\_date, 888
- cob\_accept\_date\_yyyymmdd, 888
- cob\_accept\_day, 888
- cob\_accept\_day\_of\_week, 888
- cob\_accept\_day\_yyyddd, 888
- cob\_accept\_environment, 888
- cob\_accept\_time, 888
- cob\_acuw\_justify, 888
- cob\_acuw\_sleep, 888
- cob\_allocate, 888
- cob\_call\_params, 891
- cob\_chain\_setup, 888
- cob\_check\_based, 888
- cob\_check\_numeric, 888
- cob\_check\_odo, 889
- cob\_check\_ref\_mod, 889
- cob\_check\_subscript, 889
- cob\_check\_version, 889
- cob\_cmp, 889
- cob\_current\_module, 891
- cob\_d2i, 880
- cob\_display\_arg\_number, 889
- cob\_display\_command\_line, 889
- cob\_display\_env\_value, 889
- cob\_display\_environment, 889
- COB\_DISPLAY\_SIGN\_ASCII, 880
- COB\_DISPLAY\_SIGN\_EBCDIC, 880
- COB\_EXCEPTION, 880
- cob\_exception\_code, 891
- cob\_exception\_id, 887
- cob\_external\_addr, 889
- cob\_fatal\_error, 889
- COB\_FERROR\_CHAINING, 880
- COB\_FERROR\_CODEGEN, 880
- COB\_FERROR\_INITIALIZED, 880
- COB\_FERROR\_STACK, 880
- COB\_FIELD\_BINARY\_SWAP, 881
- COB\_FIELD\_BLANK\_ZERO, 881
- COB\_FIELD\_DATA, 881
- COB\_FIELD\_DIGITS, 881
- COB\_FIELD\_FLAGS, 881
- COB\_FIELD\_HAVE\_SIGN, 881
- COB\_FIELD\_IS\_NUMERIC, 881
- COB\_FIELD\_IS\_POINTER, 881
- COB\_FIELD\_JUSTIFIED, 881
- COB\_FIELD\_PIC, 881
- COB\_FIELD\_REAL\_BINARY, 882
- COB\_FIELD\_SCALE, 882
- COB\_FIELD\_SIGN\_LEADING, 882
- COB\_FIELD\_SIGN\_SEPARATE, 882
- COB\_FIELD\_SIZE, 882
- COB\_FIELD\_TYPE, 882
- COB\_FLAG\_BINARY\_SWAP, 882
- COB\_FLAG\_BLANK\_ZERO, 882
- COB\_FLAG\_HAVE\_SIGN, 882
- COB\_FLAG\_IS\_POINTER, 882
- COB\_FLAG\_JUSTIFIED, 883
- COB\_FLAG\_REAL\_BINARY, 883
- COB\_FLAG\_SIGN\_LEADING, 883
- COB\_FLAG\_SIGN\_SEPARATE, 883
- cob\_free\_alloc, 889
- cob\_get\_environment, 889
- cob\_get\_exception\_name, 889
- cob\_get\_pointer, 889
- cob\_get\_prog\_pointer, 889
- cob\_get\_sign, 883
- cob\_get\_switch, 889
- cob\_high, 891
- cob\_i2d, 883
- cob\_init, 889
- cob\_initial\_external, 891
- cob\_initialized, 891
- COB\_INLINE, 883
- cob\_is\_alpha, 889
- cob\_is\_lower, 889
- cob\_is\_numeric, 889
- cob\_is\_omitted, 889
- cob\_is\_upper, 889
- COB\_LARGE\_BUFF, 883
- COB\_LARGE\_MAX, 883
- cob\_low, 891
- cob\_malloc, 889
- COB\_MAX\_FIELD\_PARAMS, 883
- COB\_MEDIUM\_BUFF, 884
- COB\_MEDIUM\_MAX, 884

- COB\_MINI\_BUFF, 884
- COB\_MINI\_MAX, 884
- cob\_module\_enter, 890
- cob\_module\_leave, 890
- COB\_NOINLINE, 884
- COB\_NORMAL\_BUFF, 884
- COB\_NORMAL\_MAX, 884
- cob\_numeric\_cmp, 890
- cob\_one, 891
- cob\_parameter\_size, 890
- cob\_put\_sign, 884
- cob\_quote, 891
- cob\_ready\_trace, 890
- cob\_reset\_trace, 890
- cob\_return\_args, 890
- cob\_runtime\_error, 890
- cob\_save\_call\_params, 892
- cob\_set\_environment, 890
- cob\_set\_exception, 890
- cob\_set\_location, 890
- cob\_set\_switch, 890
- COB\_SMALL\_BUFF, 884
- COB\_SMALL\_MAX, 884
- cob\_space, 892
- COB\_STACK\_SIZE, 885
- cob\_stop\_run, 890
- cob\_table\_sort, 890
- cob\_table\_sort\_init, 890
- cob\_table\_sort\_init\_key, 890
- COB\_TYPE\_ALPHANUMERIC, 885
- COB\_TYPE\_ALPHANUMERIC\_ALL, 885
- COB\_TYPE\_ALPHANUMERIC\_EDITED, 885
- COB\_TYPE\_BOOLEAN, 885
- COB\_TYPE\_GROUP, 885
- COB\_TYPE\_NATIONAL, 885
- COB\_TYPE\_NATIONAL\_EDITED, 885
- COB\_TYPE\_NUMERIC, 885
- COB\_TYPE\_NUMERIC\_BINARY, 885
- COB\_TYPE\_NUMERIC\_DISPLAY, 886
- COB\_TYPE\_NUMERIC\_DOUBLE, 886
- COB\_TYPE\_NUMERIC\_EDITED, 886
- COB\_TYPE\_NUMERIC\_FLOAT, 886
- COB\_TYPE\_NUMERIC\_PACKED, 886
- COB\_TYPE\_UNKNOWN, 886
- cob\_zero, 892
- cobcommandline, 890
- cobexit, 890
- cobgetenv, 890
- cobinit, 891
- cobputenv, 891
- cobtidy, 891
- DLL\_EXPIMP, 886
- GET\_SIGN\_ASCII, 886
- likely, 886
- PUT\_SIGN\_ASCII, 886
- SYSTEM, 891
- ucharptr, 887
- unlikely, 887
- COMP
  - parser.c, 230, 278, 290
  - parser.h, 333, 372, 383
  - ppparse.c, 435, 446
  - ppparse.h, 469, 481
- COMP\_1
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_2
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_3
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_4
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_5
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_X
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- COMP\_1
  - parser.c, 230
  - parser.h, 333
- COMP\_2
  - parser.c, 230

- parser.h, 333
- COMP\_3
  - parser.c, 231
  - parser.h, 333
- COMP\_4
  - parser.c, 231
  - parser.h, 333
- COMP\_5
  - parser.c, 231
  - parser.h, 333
- COMP\_X
  - parser.c, 231
  - parser.h, 333
- COMPUTE
  - parser.c, 231, 278, 290
  - parser.h, 333, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 469, 481
- CONCATENATE\_FUNC
  - parser.c, 278, 290
  - parser.h, 372, 384
  - ppparse.c, 435, 446
  - ppparse.h, 470, 481
- CONCATENATE\_FUNC
  - parser.c, 231
  - parser.h, 334
- config.c
  - ANY, 186
  - BOOLEAN, 186
  - CB\_CONFIG\_ANY, 185
  - CB\_CONFIG\_BOOLEAN, 185
  - CB\_CONFIG\_INT, 185
  - CB\_CONFIG\_STRING, 185, 186
  - CB\_CONFIG\_SUPPORT, 186
  - cb\_config\_type, 186
  - cb\_load\_conf, 186
  - cb\_load\_std, 190
  - INT, 186
  - name, 190
  - norestab, 190
  - STRING, 186
  - SUPPORT, 186
  - type, 190
  - val, 191
  - var, 191
- config.h, 797
  - \_\_USE\_STRING\_INLINES, 798
  - COB\_EXEEXT, 798
  - COB\_EXPORT\_DYN, 798
  - COB\_EXTRA\_FLAGS, 799
  - COB\_HAS\_INLINE, 799
  - COB\_LI\_IS\_LL, 799
  - COB\_PIC\_FLAGS, 799
  - COB\_SHARED\_OPT, 799
  - COB\_STRFTIME, 799
  - COB\_STRIP\_CMD, 799
  - ENABLE\_NLS, 799
  - HAVE\_ALLOCA, 799
  - HAVE\_ALLOCA\_H, 799
  - HAVE\_ATTRIBUTE\_ALIGNED, 800
  - HAVE\_COLOR\_SET, 800
  - HAVE\_DCGETTEXT, 800
  - HAVE\_DLFCN\_H, 800
  - HAVE\_FCNTL, 800
  - HAVE\_FCNTL\_H, 800
  - HAVE\_GETOPT\_H, 800
  - HAVE\_GETTEXT, 800
  - HAVE\_GETTIMEOFDAY, 800
  - HAVE\_GMP\_H, 800
  - HAVE\_ICONV, 801
  - HAVE\_INTTYPES\_H, 801
  - HAVE\_LANGINFO\_CODESET, 801
  - HAVE\_LIBGMP, 801
  - HAVE\_LIBNCURSES, 801
  - HAVE\_LOCALE\_H, 801
  - HAVE\_MALLOC\_H, 801
  - HAVE\_MEMMOVE, 801
  - HAVE\_MEMORY\_H, 801
  - HAVE\_MEMSET, 801
  - HAVE\_NCURSES\_H, 802
  - HAVE\_PSIGN\_OPT, 802
  - HAVE\_SETLOCALE, 802
  - HAVE\_SIGNAL\_H, 802
  - HAVE\_STDDEF\_H, 802
  - HAVE\_STDINT\_H, 802
  - HAVE\_STDLIB\_H, 802
  - HAVE\_STRCASECMP, 802
  - HAVE\_STRCHR, 802
  - HAVE\_STRDUP, 802
  - HAVE\_STRERROR, 803
  - HAVE\_STRING\_H, 803
  - HAVE\_STRINGS\_H, 803
  - HAVE\_STRRCHR, 803
  - HAVE\_STRSTR, 803
  - HAVE\_STRTOL, 803
  - HAVE\_SYS\_STAT\_H, 803
  - HAVE\_SYS\_TIME\_H, 803
  - HAVE\_SYS\_TYPES\_H, 803
  - HAVE\_TIMEZONE, 803
  - HAVE\_UNISTD\_H, 804

- HAVE\_VPRINTF, [804](#)
- HAVE\_WCHAR\_H, [804](#)
- ICONV\_CONST, [804](#)
- PACKAGE, [804](#)
- PACKAGE\_BUGREPORT, [804](#)
- PACKAGE\_NAME, [804](#)
- PACKAGE\_STRING, [804](#)
- PACKAGE\_TARNAME, [804](#)
- PACKAGE\_VERSION, [804](#)
- PATCH\_LEVEL, [805](#)
- STDC\_HEADERS, [805](#)
- USE\_DB41, [805](#)
- USE\_LIBDL, [805](#)
- VERSION, [805](#)
- WITH\_DB, [805](#)
- WITH\_VARSEQ, [805](#)
- CONFIGURATION
  - [parser.c](#), [231](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- const
  - [getopt.c](#), [811](#)
  - [getopt1.c](#), [827](#)
- CONSTANT
  - [parser.c](#), [231](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTAINS
  - [parser.c](#), [231](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTENT
  - [parser.c](#), [231](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTINUE
  - [parser.c](#), [232](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTROL
  - [parser.c](#), [232](#), [278](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTROL\_FOOTING
  - [parser.c](#), [279](#), [290](#)
  - [parser.h](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTROL\_HEADING
  - [parser.c](#), [279](#), [290](#)
  - [parser.h](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONTROL\_FOOTING
  - [parser.c](#), [232](#)
  - [parser.h](#), [334](#)
- CONTROL\_HEADING
  - [parser.c](#), [232](#)
  - [parser.h](#), [334](#)
- CONTROLS
  - [parser.c](#), [232](#), [279](#), [290](#)
  - [parser.h](#), [334](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [446](#)
  - [ppparse.h](#), [470](#), [481](#)
- CONVERTING
  - [parser.c](#), [232](#), [279](#), [290](#)
  - [parser.h](#), [335](#), [372](#), [384](#)
  - [ppparse.c](#), [435](#), [447](#)
  - [ppparse.h](#), [470](#), [481](#)
- COPY
  - [parser.c](#), [299](#), [300](#)
  - [parser.h](#), [393](#), [394](#)
  - [ppparse.c](#), [424](#), [456](#)
  - [ppparse.h](#), [466](#), [491](#)
- COPY\_STATE
  - [plex.c](#), [404](#)
- CORRESPONDING
  - [parser.c](#), [232](#), [279](#), [290](#)
  - [parser.h](#), [335](#), [373](#), [384](#)
  - [ppparse.c](#), [435](#), [447](#)
  - [ppparse.h](#), [470](#), [481](#)
- COUNT
  - [parser.c](#), [232](#), [279](#), [290](#)
  - [parser.h](#), [335](#), [373](#), [384](#)
  - [ppparse.c](#), [435](#), [447](#)
  - [ppparse.h](#), [470](#), [481](#)
- count
  - [cb\\_field](#), [32](#)
  - [cb\\_word](#), [91](#)
  - [file\\_struct](#), [118](#)
  - [memory\\_struct](#), [128](#)
- [cpucheck.c](#), [805](#)
- [main](#), [806](#)
- critical



- cob\_exception, 94
- CRT
  - parser.c, 232, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 481
- crt\_status
  - cb\_program, 73
  - cb\_module, 109
- CURRENCY
  - parser.c, 232, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- currency\_symbol
  - cb\_program, 74
  - cb\_module, 109
- CURRENT\_DATE\_FUNC
  - parser.c, 279, 290
  - parser.h, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- CURRENT\_DATE\_FUNC
  - parser.c, 233
  - parser.h, 335
- current\_paragraph
  - cobc.h, 170
  - parser.c, 307
- current\_program
  - cobc.h, 170
  - parser.c, 307
- current\_section
  - cobc.h, 170
  - parser.c, 307
- current\_statement
  - cobc.h, 170
  - parser.c, 307
- CURSOR
  - parser.c, 233, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- cursor
  - indexed\_file, 123
- cursor\_pos
  - cb\_program, 74
  - cb\_module, 109
- custom\_list
  - cb\_alphabet\_name, 17
- CYCLE
  - parser.c, 233, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- cycle\_label
  - cb\_perform, 66
- d2
  - intrinsic.c, 1015
- d3
  - intrinsic.c, 1016
- d4
  - intrinsic.c, 1016
- d5
  - intrinsic.c, 1016
- DATA
  - parser.c, 233, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- data
  - cb\_literal, 62
  - cb\_perform, 66
  - cb\_string, 87
  - cob\_field, 96
  - indexed\_file, 123
- DATE
  - parser.c, 233, 279, 290
  - parser.h, 335, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DAY
  - parser.c, 233, 279, 290
  - parser.h, 336, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DAY\_OF\_WEEK
  - parser.c, 279, 290
  - parser.h, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DAY\_OF\_WEEK
  - parser.c, 233
  - parser.h, 336
- db
  - indexed\_file, 123
- DB\_CLOSE
  - fileio.c, 895
- DB\_DEL
  - fileio.c, 896

- DB\_GET
  - fileio.c, 896
- DB\_PUT
  - fileio.c, 896
- DB\_SEQ
  - fileio.c, 896
- DB\_SYNC
  - fileio.c, 896
- DBT\_SET
  - fileio.c, 896
- dcgettext
  - gettext.h, 830
- dcngettext
  - gettext.h, 830
- DE
  - parser.c, 233, 279, 290
  - parser.h, 336, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DEBUGGING
  - parser.c, 233, 279, 290
  - parser.h, 336, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DECIMAL\_POINT
  - parser.c, 279, 290
  - parser.h, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DECIMAL\_CHECK
  - numeric.c, 1081
- decimal\_index
  - cb\_program, 74
- decimal\_index\_max
  - cb\_program, 74
- DECIMAL\_IS\_COMMA
  - scanner.c, 499
- DECIMAL\_IS\_PERIOD
  - scanner.c, 499
- DECIMAL\_NAN
  - numeric.c, 1081
- DECIMAL\_POINT
  - parser.c, 233
  - parser.h, 336
- decimal\_point
  - cb\_program, 74
  - cob\_module, 109
- DECLARATIVES
  - parser.c, 234, 279, 290
  - parser.h, 336, 373, 384
- ppparse.c, 435, 447
- ppparse.h, 470, 482
- def
  - cb\_initialize, 49
- DEFAULT
  - parser.c, 234, 279, 290
  - parser.h, 336, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- defaults.h, 807
  - COB\_CC, 808
  - COB\_CFLAGS, 808
  - COB\_CONFIG\_DIR, 808
  - COB\_COPY\_DIR, 808
  - COB\_LDFLAGS, 808
  - COB\_LIBRARY\_PATH, 808
  - COB\_LIBS, 808
  - COB\_MODULE\_EXT, 808
  - LOCALEDIR, 808
- DELETE
  - parser.c, 234, 279, 291
  - parser.h, 336, 373, 384
  - ppparse.c, 435, 447
  - ppparse.h, 470, 482
- DELIMITED
  - parser.c, 234, 279, 291
  - parser.h, 336, 373, 384
  - ppparse.c, 436, 447
  - ppparse.h, 470, 482
- DELIMITER
  - parser.c, 234, 279, 291
  - parser.h, 336, 373, 384
  - ppparse.c, 436, 447
  - ppparse.h, 470, 482
- demangle\_name
  - cobc.c, 154
  - cobc.h, 170
- demangle\_source
  - filename, 119
- DEPENDING
  - parser.c, 234, 279, 291
  - parser.h, 337, 373, 384
  - ppparse.c, 436, 447
  - ppparse.h, 470, 482
- depending
  - cb\_goto, 47
- DEPTH\_LEVEL
  - intrinsic.c, 963
- DESCENDING
  - parser.c, 234, 279, 291

- parser.h, [337](#), [373](#), [385](#)
- ppparse.c, [436](#), [447](#)
- ppparse.h, [470](#), [482](#)
- destination\_file
  - cobsort, [114](#)
- DETAIL
  - parser.c, [234](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- dgettext
  - gettext.h, [830](#)
- digits
  - cb\_picture, [69](#)
  - cob\_field\_attr, [97](#)
- dir
  - cb\_field::cb\_key, [55](#)
- DISK
  - parser.c, [234](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- DISPLAY
  - parser.c, [234](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- display\_sign
  - cob\_module, [109](#)
- DIVIDE
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- DIVISION
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- DLL\_EXPIMP
  - common.h, [886](#)
- DLM\_DEFAULT\_NUM
  - strings.c, [1125](#)
- dlim\_struct, [116](#)
  - uns\_all, [116](#)
  - uns\_dlm, [116](#)
- dngettext
  - gettext.h, [830](#)
- DOWN
  - parser.c, [235](#), [279](#), [291](#)
- parser.h, [337](#), [373](#), [385](#)
- ppparse.c, [436](#), [447](#)
- ppparse.h, [471](#), [482](#)
- down\_index
  - cob\_inp\_struct, [107](#)
- dpush
  - typeck.c, [726](#)
- dummymac.c
  - dummymacfix, [809](#)
- dummymacfix
  - dummymac.c, [809](#)
- DUPLICATES
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- duplicates
  - cb\_alt\_key, [19](#)
- DYNAMIC
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [337](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- EBCDIC
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [338](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- ECHO
  - pplex.c, [404](#)
  - scanner.c, [499](#)
- ELSE
  - parser.c, [235](#), [279](#), [291](#)
  - parser.h, [338](#), [373](#), [385](#)
  - ppparse.c, [436](#), [447](#)
  - ppparse.h, [471](#), [482](#)
- emit\_statement
  - parser.c, [235](#)
- empty
  - cobsort, [114](#)
- enable
  - cb\_exception, [29](#)
- ENABLE\_NLS
  - config.h, [799](#)
- ename
  - cb\_field, [32](#)
  - cob\_external, [95](#)
- END
  - parser.c, [235](#), [279](#), [291](#)

- parser.h, 338, 373, 385
- ppparse.c, 436, 447
- ppparse.h, 471, 482
- END\_ACCEPT
  - parser.c, 279, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 447
  - ppparse.h, 471, 482
- END\_ADD
  - parser.c, 279, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 447
  - ppparse.h, 471, 482
- END\_CALL
  - parser.c, 280, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 447
  - ppparse.h, 471, 482
- END\_COMPUTE
  - parser.c, 280, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 447
  - ppparse.h, 471, 482
- END\_DELETE
  - parser.c, 280, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 447
  - ppparse.h, 471, 482
- END\_DISPLAY
  - parser.c, 280, 291
  - parser.h, 373, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 482
- END\_DIVIDE
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 482
- END\_EVALUATE
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 482
- END\_FUNCTION
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 482
- END\_IF
  - parser.c, 280, 291
- parser.h, 374, 385
- ppparse.c, 436, 448
- ppparse.h, 471, 483
- END\_MULTIPLY
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_PERFORM
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_PROGRAM
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_READ
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_RETURN
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_REWRITE
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_SEARCH
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_START
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_STRING
  - parser.c, 280, 291
  - parser.h, 374, 385
  - ppparse.c, 436, 448
  - ppparse.h, 471, 483
- END\_SUBTRACT
  - parser.c, 280, 291

- parser.h, [374](#), [385](#)
- ppparse.c, [436](#), [448](#)
- ppparse.h, [471](#), [483](#)
- END\_UNSTRING
  - parser.c, [280](#), [291](#)
  - parser.h, [374](#), [385](#)
  - ppparse.c, [436](#), [448](#)
  - ppparse.h, [471](#), [483](#)
- END\_WRITE
  - parser.c, [280](#), [291](#)
  - parser.h, [374](#), [385](#)
  - ppparse.c, [436](#), [448](#)
  - ppparse.h, [471](#), [483](#)
- END\_ACCEPT
  - parser.c, [235](#)
  - parser.h, [338](#)
- END\_ADD
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_CALL
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_COMPUTE
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_DELETE
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_DISPLAY
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_DIVIDE
  - parser.c, [236](#)
  - parser.h, [338](#)
- END\_EVALUATE
  - parser.c, [236](#)
  - parser.h, [339](#)
- END\_FUNCTION
  - parser.c, [236](#)
  - parser.h, [339](#)
- END\_IF
  - parser.c, [236](#)
  - parser.h, [339](#)
- END\_MULTIPLY
  - parser.c, [236](#)
  - parser.h, [339](#)
- end\_of\_block
  - cobitem, [111](#)
- END\_PERFORM
  - parser.c, [237](#)
- parser.h, [339](#)
- END\_PROGRAM
  - parser.c, [237](#)
  - parser.h, [339](#)
- END\_READ
  - parser.c, [237](#)
  - parser.h, [339](#)
- END\_RETURN
  - parser.c, [237](#)
  - parser.h, [339](#)
- END\_REWRITE
  - parser.c, [237](#)
  - parser.h, [339](#)
- END\_SEARCH
  - parser.c, [237](#)
  - parser.h, [339](#)
- END\_START
  - parser.c, [237](#)
  - parser.h, [340](#)
- end\_stmt
  - cb\_search, [83](#)
- END\_STRING
  - parser.c, [237](#)
  - parser.h, [340](#)
- END\_SUBTRACT
  - parser.c, [237](#)
  - parser.h, [340](#)
- END\_UNSTRING
  - parser.c, [237](#)
  - parser.h, [340](#)
- END\_WRITE
  - parser.c, [238](#)
  - parser.h, [340](#)
- ENTRY
  - parser.c, [238](#), [280](#), [292](#)
  - parser.h, [340](#), [374](#), [385](#)
  - ppparse.c, [436](#), [448](#)
  - ppparse.h, [471](#), [483](#)
- entry\_list
  - cb\_program, [74](#)
- ENVIRONMENT
  - parser.c, [238](#), [280](#), [292](#)
  - parser.h, [340](#), [374](#), [385](#)
  - ppparse.c, [437](#), [448](#)
  - ppparse.h, [471](#), [483](#)
- ENVIRONMENT\_NAME
  - parser.c, [280](#), [292](#)
  - parser.h, [374](#), [385](#)
  - ppparse.c, [437](#), [448](#)
  - ppparse.h, [471](#), [483](#)

- ENVIRONMENT\_VALUE  
 parser.c, 280, 292  
 parser.h, 374, 385  
 ppparse.c, 437, 448  
 ppparse.h, 471, 483
- ENVIRONMENT\_NAME  
 parser.c, 238  
 parser.h, 340
- ENVIRONMENT\_VALUE  
 parser.c, 238  
 parser.h, 340
- EOB\_ACT\_CONTINUE\_SCAN  
 pplex.c, 404  
 scanner.c, 499
- EOB\_ACT\_END\_OF\_FILE  
 pplex.c, 405  
 scanner.c, 499
- EOB\_ACT\_LAST\_MATCH  
 pplex.c, 405  
 scanner.c, 500
- EOL  
 parser.c, 238, 280, 292  
 parser.h, 340, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 471, 483
- EOP  
 parser.c, 238, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- EOS  
 parser.c, 238, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- EQEQ  
 parser.c, 300  
 parser.h, 393, 394  
 ppparse.c, 424, 456  
 ppparse.h, 466, 491
- EQUAL  
 parser.c, 238, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- EQUALS  
 parser.c, 238, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- ERASE  
 parser.c, 239, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- ERROR  
 parser.c, 239, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- error  
 cb\_word, 91
- error.c  
 ambiguous\_error, 192  
 cb\_error, 193  
 cb\_error\_x, 193  
 cb\_verify, 193  
 cb\_warning, 194  
 cb\_warning\_x, 194  
 check\_filler\_name, 195  
 group\_error, 195  
 level\_except\_error, 195  
 level\_redundant\_error, 195  
 level\_require\_error, 195  
 redefinition\_error, 196  
 redefinition\_warning, 196  
 undefined\_error, 196
- errorcount  
 cobc.c, 154  
 cobc.h, 170
- ESCAPE  
 parser.c, 239, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- esize  
 cob\_external, 95
- EVALUATE  
 parser.c, 239, 280, 292  
 parser.h, 341, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- EVENT\_STATUS  
 parser.c, 280, 292  
 parser.h, 374, 386  
 ppparse.c, 437, 448  
 ppparse.h, 472, 483
- EVENT\_STATUS  
 parser.c, 239  
 parser.h, 341

- EXCEPTION
  - parser.c, [239, 280, 292](#)
  - parser.h, [341, 374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- EXCEPTION\_TAB\_SIZE
  - common.c, [873](#)
- EXCLUSIVE
  - parser.c, [239, 280, 292](#)
  - parser.h, [342, 374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- exec\_list
  - cb\_program, [74](#)
- EXIT
  - parser.c, [239, 280, 292](#)
  - parser.h, [342, 374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- exit\_label
  - cb\_label, [57](#)
  - cb\_perform, [66](#)
- exit\_label\_ref
  - cb\_label, [57](#)
- expr\_node, [117](#)
  - token, [117](#)
  - value, [117](#)
- ext\_alloc
  - cob\_external, [95](#)
- EXTEND
  - parser.c, [239, 280, 292](#)
  - parser.h, [342, 374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- EXTERNAL
  - parser.c, [239, 281, 292](#)
  - parser.h, [342, 374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- external
  - cb\_file, [42](#)
- external\_assign
  - cb\_file, [42](#)
- extfh\_ptr
  - cob\_file, [99](#)
- false\_88
  - cb\_field, [33](#)
- FD
  - parser.c, [240, 281, 292](#)
- parser.h, [342, 374, 386](#)
- ppparse.c, [437, 448](#)
- ppparse.h, [472, 483](#)
- fdelete
  - cob\_fileio\_funcs, [104](#)
- field
  - \_\_cob\_screen, [14](#)
  - cob\_file\_key, [103](#)
- field.c
  - cb\_build\_field\_tree, [198](#)
  - cb\_clear\_real\_field, [200](#)
  - cb\_get\_level, [200](#)
  - cb\_needs\_01, [204](#)
  - cb\_resolve\_redefines, [201](#)
  - cb\_validate\_78\_item, [202](#)
  - cb\_validate\_88\_item, [203](#)
  - cb\_validate\_field, [203](#)
- file
  - cb\_field, [33](#)
  - cb\_statement, [86](#)
  - cob\_file, [99](#)
  - cobsort, [114](#)
  - pplex.c, [413](#)
  - scanner.c, [507](#)
- FILE\_CONTROL
  - parser.c, [281, 292](#)
  - parser.h, [374, 386](#)
  - ppparse.c, [437, 448](#)
  - ppparse.h, [472, 483](#)
- FILE\_ID
  - parser.c, [281, 292](#)
  - parser.h, [374, 386](#)
  - ppparse.c, [437, 449](#)
  - ppparse.h, [472, 483](#)
- FILE\_CONTROL
  - parser.c, [240](#)
  - parser.h, [342](#)
- FILE\_ID
  - parser.c, [240](#)
  - parser.h, [342](#)
- file\_list
  - cb\_program, [74](#)
- file\_status
  - cb\_file, [42](#)
  - cob\_file, [99](#)
- file\_struct, [118](#)
  - count, [118](#)
  - fp, [118](#)
- file\_version
  - cob\_file, [100](#)

- fileid\_assign
  - cb\_file, 42
- fileio.c
  - \_FILE\_OFFSET\_BITS, 895
  - \_LARGEFILE64\_SOURCE, 895
  - \_LFS64\_LARGEFILE, 895
  - \_LFS64\_STDIO, 895
  - CBL\_CHANGE\_DIR, 897
  - CBL\_CHECK\_FILE\_EXIST, 897
  - CBL\_CLOSE\_FILE, 898
  - CBL\_COPY\_FILE, 898
  - CBL\_CREATE\_DIR, 899
  - CBL\_CREATE\_FILE, 900
  - CBL\_DELETE\_DIR, 900
  - CBL\_DELETE\_FILE, 900
  - CBL\_FLUSH\_FILE, 901
  - CBL\_GET\_CURRENT\_DIR, 901
  - CBL\_OPEN\_FILE, 902
  - CBL\_READ\_FILE, 902
  - CBL\_RENAME\_FILE, 903
  - CBL\_WRITE\_FILE, 903
  - cob\_acuw\_chdir, 904
  - cob\_acuw\_copyfile, 904
  - cob\_acuw\_file\_delete, 905
  - cob\_acuw\_file\_info, 905
  - cob\_acuw\_mkdir, 906
  - cob\_close, 906
  - cob\_commit, 907
  - cob\_dbtsize\_t, 895
  - cob\_default\_error\_handle, 907
  - cob\_delete, 909
  - cob\_error\_file, 923
  - cob\_exit\_fileio, 909
  - cob\_file\_release, 910
  - cob\_file\_return, 910
  - cob\_file\_sort\_close, 911
  - cob\_file\_sort\_giving, 911
  - cob\_file\_sort\_init, 912
  - cob\_file\_sort\_init\_key, 913
  - cob\_file\_sort\_using, 913
  - cob\_init\_fileio, 913
  - cob\_open, 914
  - cob\_read, 919
  - cob\_rewrite, 920
  - cob\_rollback, 921
  - cob\_start, 921
  - cob\_unlock\_file, 921
  - cob\_write, 922
  - COBSORTABORT, 895
  - COBSORTEND, 895
  - COBSORTFILEERR, 895
  - COBSORTNOTOPEN, 895
  - DB\_CLOSE, 895
  - DB\_DEL, 896
  - DB\_GET, 896
  - DB\_PUT, 896
  - DB\_SEQ, 896
  - DB\_SYNC, 896
  - DBT\_SET, 896
  - INITIAL\_FLAGS, 896
  - NUM\_PREFIX, 896
  - O\_BINARY, 896
  - O\_LARGEFILE, 896
  - RETURN\_STATUS, 897
  - SEEK\_INIT, 897
- fileio.h
  - CBL\_CHANGE\_DIR, 936
  - CBL\_CHECK\_FILE\_EXIST, 936
  - CBL\_CLOSE\_FILE, 937
  - CBL\_COPY\_FILE, 937
  - CBL\_CREATE\_DIR, 938
  - CBL\_CREATE\_FILE, 938
  - CBL\_DELETE\_DIR, 939
  - CBL\_DELETE\_FILE, 939
  - CBL\_FLUSH\_FILE, 939
  - CBL\_GET\_CURRENT\_DIR, 940
  - CBL\_OPEN\_FILE, 940
  - CBL\_READ\_FILE, 941
  - CBL\_RENAME\_FILE, 941
  - CBL\_WRITE\_FILE, 942
  - COB\_ACCESS\_DYNAMIC, 927
  - COB\_ACCESS\_RANDOM, 927
  - COB\_ACCESS\_SEQUENTIAL, 927
  - cob\_acuw\_chdir, 942
  - cob\_acuw\_copyfile, 943
  - cob\_acuw\_file\_delete, 943
  - cob\_acuw\_file\_info, 944
  - cob\_acuw\_mkdir, 944
  - COB\_ASCENDING, 927
  - cob\_close, 945
  - COB\_CLOSE\_LOCK, 927
  - COB\_CLOSE\_NO\_REWIND, 927
  - COB\_CLOSE\_NORMAL, 927
  - COB\_CLOSE\_UNIT, 927
  - COB\_CLOSE\_UNIT\_REMOVAL, 927
  - cob\_commit, 946
  - cob\_default\_error\_handle, 946
  - cob\_delete, 947
  - COB\_DESCENDING, 928
  - COB\_EQ, 928



- cob\_error\_file, 960
- COB\_FILE\_MODE, 928
- cob\_file\_release, 948
- cob\_file\_return, 948
- cob\_file\_sort\_close, 949
- cob\_file\_sort\_giving, 949
- cob\_file\_sort\_init, 950
- cob\_file\_sort\_init\_key, 951
- cob\_file\_sort\_using, 951
- COB\_FILE\_VERSION, 928
- COB\_GE, 928
- COB\_GT, 928
- COB\_LE, 928
- COB\_LINAGE\_INVALID, 928
- COB\_LOCK\_AUTOMATIC, 928
- COB\_LOCK\_EXCLUSIVE, 928
- COB\_LOCK\_MANUAL, 929
- COB\_LOCK\_MASK, 929
- COB\_LOCK\_MULTIPLE, 929
- COB\_LT, 929
- COB\_NE, 929
- COB\_NOT\_CONFIGURED, 929
- cob\_open, 951
- COB\_OPEN\_CLOSED, 929
- COB\_OPEN\_EXTEND, 929
- COB\_OPEN\_I\_O, 929
- COB\_OPEN\_INPUT, 929
- COB\_OPEN\_LOCKED, 930
- COB\_OPEN\_OUTPUT, 930
- COB\_ORG\_INDEXED, 930
- COB\_ORG\_LINE\_SEQUENTIAL, 930
- COB\_ORG\_MAX, 930
- COB\_ORG\_RELATIVE, 930
- COB\_ORG\_SEQUENTIAL, 930
- COB\_ORG\_SORT, 930
- cob\_read, 956
- COB\_READ\_FIRST, 930
- COB\_READ\_IGNORE\_LOCK, 930
- COB\_READ\_KEPT\_LOCK, 931
- COB\_READ\_LAST, 931
- COB\_READ\_LOCK, 931
- COB\_READ\_NEXT, 931
- COB\_READ\_NO\_LOCK, 931
- COB\_READ\_PREVIOUS, 931
- COB\_READ\_WAIT\_LOCK, 931
- cob\_rewrite, 957
- cob\_rollback, 958
- COB\_SELECT\_EXTERNAL, 931
- COB\_SELECT\_FILE\_STATUS, 931
- COB\_SELECT\_LINAGE, 931
- COB\_SELECT\_SPLITKEY, 932
- cob\_start, 958
- COB\_STATUS\_00\_SUCCESS, 932
- COB\_STATUS\_02\_SUCCESS\_DUPLICATE, 932
- COB\_STATUS\_04\_SUCCESS\_INCOMPLETE, 932
- COB\_STATUS\_05\_SUCCESS\_OPTIONAL, 932
- COB\_STATUS\_07\_SUCCESS\_NO\_-UNIT, 932
- COB\_STATUS\_10\_END\_OF\_FILE, 932
- COB\_STATUS\_14\_OUT\_OF\_KEY\_RANGE, 932
- COB\_STATUS\_21\_KEY\_INVALID, 932
- COB\_STATUS\_22\_KEY\_EXISTS, 932
- COB\_STATUS\_23\_KEY\_NOT\_EXISTS, 933
- COB\_STATUS\_30\_PERMANENT\_ERROR, 933
- COB\_STATUS\_31\_INCONSISTENT\_-FILENAME, 933
- COB\_STATUS\_34\_BOUNDARY\_VIOLATION, 933
- COB\_STATUS\_35\_NOT\_EXISTS, 933
- COB\_STATUS\_37\_PERMISSION\_DENIED, 933
- COB\_STATUS\_38\_CLOSED\_WITH\_-LOCK, 933
- COB\_STATUS\_39\_CONFLICT\_ATTRIBUTE, 933
- COB\_STATUS\_41\_ALREADY\_OPEN, 933
- COB\_STATUS\_42\_NOT\_OPEN, 933
- COB\_STATUS\_43\_READ\_NOT\_DONE, 934
- COB\_STATUS\_44\_RECORD\_OVERFLOW, 934
- COB\_STATUS\_46\_READ\_ERROR, 934
- COB\_STATUS\_47\_INPUT\_DENIED, 934
- COB\_STATUS\_48\_OUTPUT\_DENIED, 934
- COB\_STATUS\_49\_I\_O\_DENIED, 934
- COB\_STATUS\_51\_RECORD\_LOCKED, 934
- COB\_STATUS\_52\_EOP, 934
- COB\_STATUS\_57\_I\_O\_LINAGE, 934
- COB\_STATUS\_61\_FILE\_SHARING, 934
- COB\_STATUS\_91\_NOT\_AVAILABLE, 935

- cob\_unlock\_file, [959](#)
- cob\_write, [959](#)
- COB\_WRITE\_AFTER, [935](#)
- COB\_WRITE\_BEFORE, [935](#)
- COB\_WRITE\_CHANNEL, [935](#)
- COB\_WRITE\_EOP, [935](#)
- COB\_WRITE\_LINES, [935](#)
- COB\_WRITE\_LOCK, [935](#)
- COB\_WRITE\_MASK, [935](#)
- COB\_WRITE\_PAGE, [935](#)
- filename, [118](#)
  - demangle\_source, [119](#)
  - indexed\_file, [123](#)
  - localfile, [119](#)
  - need\_assemble, [119](#)
  - need\_preprocess, [119](#)
  - need\_translate, [119](#)
  - next, [119](#)
  - object, [119](#)
  - preprocess, [120](#)
  - source, [120](#)
  - translate, [120](#)
  - trstorage, [120](#)
- filenamelen
  - indexed\_file, [123](#)
- files\_used
  - cobsort, [114](#)
- FILLER
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [342](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [483](#)
- FINAL
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [342](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [483](#)
- finalize\_file
  - tree.c, [546](#)
  - tree.h, [705](#)
- finalized
  - cb\_file, [42](#)
- FIRST
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [342](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [483](#)
- first
  - memory\_struct, [128](#)
- flag
  - cob\_file\_key, [103](#)
  - option, [130](#)
  - flag\_all
    - cb\_search, [83](#)
  - flag\_any\_length
    - cb\_field, [33](#)
  - flag\_anylen\_done
    - cb\_field, [33](#)
  - flag\_base
    - cb\_field, [33](#)
  - flag\_begin\_of\_file
    - cob\_file, [100](#)
  - flag\_binary\_swap
    - cb\_field, [33](#)
  - flag\_binary\_truncate
    - cob\_module, [109](#)
  - flag\_blank\_zero
    - cb\_field, [33](#)
  - flag\_chained
    - cb\_field, [33](#)
    - cb\_program, [74](#)
  - flag\_common
    - cb\_program, [74](#)
  - flag\_end\_of\_file
    - cob\_file, [100](#)
  - flag\_external
    - cb\_field, [33](#)
  - flag\_field
    - cb\_field, [33](#)
  - flag\_filename\_mapping
    - cob\_module, [109](#)
  - flag\_first\_read
    - cob\_file, [100](#)
  - flag\_global\_use
    - cb\_program, [75](#)
  - flag\_indexed\_by
    - cb\_field, [34](#)
  - flag\_initial
    - cb\_program, [75](#)
  - flag\_invalid
    - cb\_field, [34](#)
  - flag\_is\_active
    - call\_hash, [16](#)
  - flag\_is\_c\_long
    - cb\_field, [34](#)
  - flag\_is\_global
    - cb\_field, [34](#)
  - flag\_is\_pdiv\_parm
    - cb\_field, [34](#)
  - flag\_is\_pointer

- cb\_field, [34](#)
- flag\_is\_verified
  - cb\_field, [34](#)
- flag\_item\_78
  - cb\_field, [34](#)
- flag\_item\_based
  - cb\_field, [34](#)
- flag\_item\_external
  - cb\_field, [34](#)
- flag\_justified
  - cb\_field, [35](#)
- flag\_local
  - cb\_field, [35](#)
- flag\_local\_allocated
  - cb\_field, [35](#)
- flag\_main
  - cb\_program, [75](#)
- flag\_needs\_nl
  - cob\_file, [100](#)
- flag\_needs\_top
  - cob\_file, [100](#)
- flag\_no\_init
  - cb\_field, [35](#)
- flag\_nonexistent
  - cob\_file, [100](#)
- flag\_occurs
  - cb\_field, [35](#)
- flag\_optional
  - cob\_file, [100](#)
- flag\_pretty\_display
  - cob\_module, [109](#)
- flag\_read\_done
  - cob\_file, [100](#)
- flag\_real\_binary
  - cb\_field, [35](#)
- flag\_recursive
  - cb\_program, [75](#)
- flag\_screen
  - cb\_program, [75](#)
- flag\_select\_features
  - cob\_file, [100](#)
- flag\_sign\_leading
  - cb\_field, [35](#)
- flag\_sign\_separate
  - cb\_field, [35](#)
- flag\_spare
  - cb\_field, [35](#)
- flag\_statement
  - cb\_initialize, [49](#)
- flag\_synchronized
  - cb\_field, [35](#)
- flag\_validated
  - cb\_program, [75](#)
- flags
  - cob\_field\_attr, [97](#)
- fld78
  - cb\_level\_78, [60](#)
- FLEX\_SCANNER
  - pplex.c, [405](#)
  - scanner.c, [500](#)
- fnstatus
  - cobsort, [114](#)
- FOOTING
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- FOR
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- foreg
  - \_\_cob\_screen, [14](#)
- BACKGROUND\_COLOR
  - parser.c, [281](#), [292](#)
  - parser.h, [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- BACKGROUND\_COLOR
  - parser.c, [240](#)
  - parser.h, [343](#)
- FOREVER
  - parser.c, [240](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- fp
  - file\_struct, [118](#)
- FREE
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- FROM
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- from

- cb\_perform\_varying, [67](#)
- FULL
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- func
  - call\_hash, [16](#)
- FUNCTION
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [343](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- FUNCTION\_ID
  - parser.c, [281](#), [292](#)
  - parser.h, [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- FUNCTION\_NAME
  - parser.c, [281](#), [292](#)
  - parser.h, [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- FUNCTION\_ID
  - parser.c, [241](#)
  - parser.h, [343](#)
- FUNCTION\_NAME
  - parser.c, [241](#)
  - parser.h, [343](#)
- function\_spec\_list
  - cb\_program, [75](#)
- FUNCTION\_STATE
  - scanner.c, [500](#)
- functions\_are\_all
  - cobc.h, [170](#)
  - parser.c, [308](#)
- GE
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [344](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- gen\_decset
  - cb\_program, [75](#)
- gen\_file\_error
  - cb\_program, [75](#)
- gen\_ptrmanip
  - cb\_program, [75](#)
- gen\_screen\_ptr
  - tree.c, [551](#)
- tree.h, [722](#)
- gen\_udcset
  - cb\_program, [76](#)
- GENERATE
  - parser.c, [241](#), [281](#), [292](#)
  - parser.h, [344](#), [375](#), [386](#)
  - ppparse.c, [437](#), [449](#)
  - ppparse.h, [472](#), [484](#)
- GET\_SIGN\_ASCII
  - common.h, [886](#)
- getenv
  - getopt.c, [823](#)
- getopt
  - getopt.c, [823](#)
  - getopt.h, [825](#)
- getopt.c
  - \_, [811](#)
  - \_NO\_PROTO, [811](#)
  - \_getopt\_internal, [812](#)
  - attribute\_hidden, [811](#), [823](#)
  - const, [811](#)
  - getenv, [823](#)
  - getopt, [823](#)
  - GETOPT\_INTERFACE\_VERSION, [811](#)
  - NONOPTION\_P, [811](#)
  - optarg, [823](#)
  - opterr, [823](#)
  - optind, [823](#)
  - optopt, [823](#)
  - PERMUTE, [811](#)
  - REQUIRE\_ORDER, [811](#)
  - RETURN\_IN\_ORDER, [811](#)
  - SWAP\_FLAGS, [811](#)
- getopt.h
  - \_GETOPT\_H, [825](#)
  - \_getopt\_internal, [825](#)
  - getopt, [825](#)
  - getopt\_long, [825](#)
  - getopt\_long\_only, [825](#)
  - no\_argument, [825](#)
  - optarg, [826](#)
  - opterr, [826](#)
  - optind, [826](#)
  - optional\_argument, [825](#)
  - optopt, [826](#)
  - required\_argument, [825](#)
- getopt1.c
  - const, [827](#)
  - GETOPT\_INTERFACE\_VERSION, [827](#)
  - getopt\_long, [828](#)

- getopt\_long\_only, 828
  - NULL, 827
- GETOPT\_INTERFACE\_VERSION
  - getopt.c, 811
  - getopt1.c, 827
- getopt\_long
  - getopt.h, 825
  - getopt1.c, 828
- getopt\_long\_only
  - getopt.h, 825
  - getopt1.c, 828
- gettext
  - gettext.h, 830
- gettext.h
  - \_, 830
  - bind\_textdomain\_codeset, 830
  - bindtextdomain, 830
  - dcgettext, 830
  - dcngettext, 830
  - dgettext, 830
  - dngettext, 830
  - gettext, 830
  - gettext\_noop, 831
  - N\_, 831
  - ngettext, 831
  - textdomain, 831
- gettext\_noop
  - gettext.h, 831
- GIVING
  - parser.c, 241, 281, 292
  - parser.h, 344, 375, 386
  - ppparse.c, 437, 449
  - ppparse.h, 472, 484
- GLOBAL
  - parser.c, 241, 281, 293
  - parser.h, 344, 375, 386
  - ppparse.c, 437, 449
  - ppparse.h, 472, 484
- global
  - cb\_file, 42
- global\_file\_list
  - cb\_program, 76
- global\_handler
  - cb\_program, 76
- global\_list
  - cb\_program, 76
- GO
  - parser.c, 242, 281, 293
  - parser.h, 344, 375, 386
  - ppparse.c, 438, 449
  - ppparse.h, 472, 484
- GOBACK
  - parser.c, 242, 281, 293
  - parser.h, 344, 375, 386
  - ppparse.c, 438, 449
  - ppparse.h, 472, 484
- GREATER
  - parser.c, 242, 281, 293
  - parser.h, 344, 375, 386
  - ppparse.c, 438, 449
  - ppparse.h, 472, 484
- GROUP
  - parser.c, 242, 281, 293
  - parser.h, 344, 375, 387
  - ppparse.c, 438, 449
  - ppparse.h, 472, 484
- group\_error
  - error.c, 195
  - tree.h, 707
- handler
  - cb\_file, 42
- handler1
  - cb\_statement, 86
- handler2
  - cb\_statement, 86
- handler3
  - cb\_statement, 86
- handler\_id
  - cb\_statement, 86
- handler\_label
  - handler\_struct, 122
- handler\_prog
  - cb\_file, 42
  - handler\_struct, 122
- handler\_struct, 120
  - handler\_label, 122
  - handler\_prog, 122
- has\_arg
  - option, 130
- has\_external
  - cobc.h, 170
  - codegen.c, 184
- HASH\_SIZE
  - call.c, 836
- HAVE\_ALLOCA
  - config.h, 799
- HAVE\_ALLOCA\_H
  - config.h, 799
- HAVE\_ATTRIBUTE\_ALIGNED

- config.h, [800](#)
- HAVE\_COLOR\_SET
  - config.h, [800](#)
- HAVE\_DCGETTEXT
  - config.h, [800](#)
- HAVE\_DLFCN\_H
  - config.h, [800](#)
- HAVE\_FCNTL
  - config.h, [800](#)
- HAVE\_FCNTL\_H
  - config.h, [800](#)
- HAVE\_GETOPT\_H
  - config.h, [800](#)
- HAVE\_GETTEXT
  - config.h, [800](#)
- HAVE\_GETTIMEOFDAY
  - config.h, [800](#)
- HAVE\_GMP\_H
  - config.h, [800](#)
- HAVE\_ICONV
  - config.h, [801](#)
- HAVE\_INTTYPES\_H
  - config.h, [801](#)
- HAVE\_LANGINFO\_CODESET
  - config.h, [801](#)
- HAVE\_LIBGMP
  - config.h, [801](#)
- HAVE\_LIBNCURSES
  - config.h, [801](#)
- HAVE\_LOCALE\_H
  - config.h, [801](#)
- HAVE\_MALLOC\_H
  - config.h, [801](#)
- HAVE\_MEMMOVE
  - config.h, [801](#)
- HAVE\_MEMORY\_H
  - config.h, [801](#)
- HAVE\_MEMSET
  - config.h, [801](#)
- HAVE\_NCURSES\_H
  - config.h, [802](#)
- HAVE\_PSIGN\_OPT
  - config.h, [802](#)
- HAVE\_SETLOCALE
  - config.h, [802](#)
- have\_sign
  - cb\_picture, [69](#)
- HAVE\_SIGNAL\_H
  - config.h, [802](#)
- HAVE\_STDDEF\_H
  - config.h, [802](#)
- HAVE\_STDINT\_H
  - config.h, [802](#)
- HAVE\_STDLIB\_H
  - config.h, [802](#)
- HAVE\_STRCASECMP
  - config.h, [802](#)
- HAVE\_STRCHR
  - config.h, [802](#)
- HAVE\_STRDUP
  - config.h, [802](#)
- HAVE\_STRERROR
  - config.h, [803](#)
- HAVE\_STRING\_H
  - config.h, [803](#)
- HAVE\_STRINGS\_H
  - config.h, [803](#)
- HAVE\_STRRCHR
  - config.h, [803](#)
- HAVE\_STRSTR
  - config.h, [803](#)
- HAVE\_STRTOL
  - config.h, [803](#)
- HAVE\_SYS\_STAT\_H
  - config.h, [803](#)
- HAVE\_SYS\_TIME\_H
  - config.h, [803](#)
- HAVE\_SYS\_TYPES\_H
  - config.h, [803](#)
- HAVE\_TIMEZONE
  - config.h, [803](#)
- HAVE\_UNISTD\_H
  - config.h, [804](#)
- HAVE\_VPRINTF
  - config.h, [804](#)
- HAVE\_WCHAR\_H
  - config.h, [804](#)
- HEADING
  - parser.c, [242](#), [281](#), [293](#)
  - parser.h, [344](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- HIGH\_VALUE
  - parser.c, [281](#), [293](#)
  - parser.h, [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- high\_val\_char
  - cb\_alphabet\_name, [18](#)
- HIGH\_VALUE

- parser.c, [242](#)
- parser.h, [344](#)
- HIGHLIGHT
  - parser.c, [242](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- I\_O
  - parser.c, [282](#), [293](#)
  - parser.h, [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- I\_O\_CONTROL
  - parser.c, [282](#), [293](#)
  - parser.h, [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- I\_O
  - parser.c, [242](#)
  - parser.h, [345](#)
- I\_O\_CONTROL
  - parser.c, [242](#)
  - parser.h, [345](#)
- ICONV\_CONST
  - config.h, [804](#)
- id
  - cb\_decimal, [29](#)
  - cb\_field, [36](#)
  - cb\_label, [57](#)
- IDENTIFICATION
  - parser.c, [242](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- IF
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- IGNORE
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- IGNORING
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- implemented
  - cb\_intrinsic\_table, [53](#)
- IN
  - parser.c, [243](#), [281](#), [293](#), [299](#), [300](#)
  - parser.h, [345](#), [375](#), [387](#), [393](#), [394](#)
  - ppparse.c, [424](#), [438](#), [449](#), [456](#)
  - ppparse.h, [466](#), [473](#), [484](#), [491](#)
- INDEX
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- index\_list
  - cb\_field, [36](#)
- index\_qual
  - cb\_field, [36](#)
- INDEXED
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [345](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- indexed\_file, [122](#)
  - bdb\_file\_lock, [123](#)
  - bdb\_lock\_id, [123](#)
  - bdb\_record\_lock, [123](#)
  - cursor, [123](#)
  - data, [123](#)
  - db, [123](#)
  - filename, [123](#)
  - filenamelen, [123](#)
  - key, [123](#)
  - key\_index, [123](#)
  - last\_dupno, [124](#)
  - last\_key, [124](#)
  - last\_readkey, [124](#)
  - record\_locked, [124](#)
  - rewrite\_sec\_key, [124](#)
  - temp\_key, [124](#)
  - write\_cursor\_open, [124](#)
- indexes
  - cb\_field, [36](#)
- INDICATE
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INITIAL
  - pplex.c, [405](#)
  - scanner.c, [500](#)
- INITIAL\_FLAGS

- fileio.c, [896](#)
- INITIALIZE
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INITIALIZE\_COMPOUND
  - codegen.c, [173](#)
- INITIALIZE\_DEFAULT
  - codegen.c, [173](#)
- INITIALIZE\_EXTERNAL
  - codegen.c, [173](#)
- INITIALIZE\_NONE
  - codegen.c, [173](#)
- INITIALIZE\_ONE
  - codegen.c, [173](#)
- INITIALIZED
  - parser.c, [243](#), [281](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INITIATE
  - parser.c, [243](#), [282](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INPUT
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INPUT\_OUTPUT
  - parser.c, [282](#), [293](#)
  - parser.h, [375](#), [387](#)
  - ppparse.c, [438](#), [449](#)
  - ppparse.h, [473](#), [484](#)
- INPUT\_OUTPUT
  - parser.c, [244](#)
  - parser.h, [346](#)
- INSPECT
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [346](#), [375](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [484](#)
- INSPECT\_ALL
  - strings.c, [1125](#)
- INSPECT\_FIRST
  - strings.c, [1125](#)
- INSPECT\_LEADING
  - strings.c, [1125](#)
- INSPECT\_TRAILING
  - strings.c, [1125](#)
- INT
  - config.c, [186](#)
- interface\_spec\_list
  - cb\_program, [76](#)
- INTO
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [346](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [484](#)
- intr\_enum
  - cb\_intrinsic\_table, [53](#)
- intr\_field
  - cb\_intrinsic, [52](#)
- intr\_routine
  - cb\_intrinsic\_table, [53](#)
- intr\_tab
  - cb\_intrinsic, [52](#)
- INTRINSIC
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [346](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [484](#)
- intrinsic.c
  - COB\_FIELD\_INIT, [963](#)
  - cob\_init\_intrinsic, [963](#)
  - cob\_intr\_abs, [963](#)
  - cob\_intr\_acos, [964](#)
  - cob\_intr\_annuity, [964](#)
  - cob\_intr\_asin, [965](#)
  - cob\_intr\_atan, [965](#)
  - cob\_intr\_binop, [966](#)
  - cob\_intr\_char, [967](#)
  - cob\_intr\_combined\_datetime, [968](#)
  - cob\_intr\_concatenate, [968](#)
  - cob\_intr\_cos, [969](#)
  - cob\_intr\_current\_date, [970](#)
  - cob\_intr\_date\_of\_integer, [971](#)
  - cob\_intr\_date\_to\_yyyymmdd, [972](#)
  - cob\_intr\_day\_of\_integer, [973](#)
  - cob\_intr\_day\_to\_yyyddd, [974](#)
  - cob\_intr\_exception\_file, [975](#)
  - cob\_intr\_exception\_location, [975](#)
  - cob\_intr\_exception\_statement, [976](#)
  - cob\_intr\_exception\_status, [977](#)
  - cob\_intr\_exp, [977](#)
  - cob\_intr\_exp10, [977](#)
  - cob\_intr\_factorial, [978](#)
  - cob\_intr\_fraction\_part, [978](#)



- cob\_intr\_integer, [979](#)
- cob\_intr\_integer\_of\_date, [979](#)
- cob\_intr\_integer\_of\_day, [980](#)
- cob\_intr\_integer\_part, [981](#)
- cob\_intr\_lcl\_time\_from\_secs, [981](#)
- cob\_intr\_length, [983](#)
- cob\_intr\_locale\_date, [984](#)
- cob\_intr\_locale\_time, [986](#)
- cob\_intr\_log, [989](#)
- cob\_intr\_log10, [989](#)
- cob\_intr\_lower\_case, [989](#)
- cob\_intr\_max, [990](#)
- cob\_intr\_mean, [990](#)
- cob\_intr\_median, [991](#)
- cob\_intr\_midrange, [992](#)
- cob\_intr\_min, [992](#)
- cob\_intr\_mod, [993](#)
- cob\_intr\_numval, [993](#)
- cob\_intr\_numval\_c, [995](#)
- cob\_intr\_ord, [996](#)
- cob\_intr\_ord\_max, [997](#)
- cob\_intr\_ord\_min, [997](#)
- cob\_intr\_present\_value, [998](#)
- cob\_intr\_random, [999](#)
- cob\_intr\_range, [1000](#)
- cob\_intr\_rem, [1000](#)
- cob\_intr\_reverse, [1001](#)
- cob\_intr\_seconds\_from\_formatted\_time, [1001](#)
- cob\_intr\_seconds\_past\_midnight, [1002](#)
- cob\_intr\_sign, [1003](#)
- cob\_intr\_sin, [1003](#)
- cob\_intr\_sqrt, [1004](#)
- cob\_intr\_standard\_deviation, [1004](#)
- cob\_intr\_stored\_char\_length, [1005](#)
- cob\_intr\_substitute, [1006](#)
- cob\_intr\_substitute\_case, [1008](#)
- cob\_intr\_sum, [1009](#)
- cob\_intr\_tan, [1010](#)
- cob\_intr\_test\_date\_yyyymmdd, [1010](#)
- cob\_intr\_test\_day\_yyyyddd, [1011](#)
- cob\_intr\_trim, [1012](#)
- cob\_intr\_upper\_case, [1012](#)
- cob\_intr\_variance, [1013](#)
- cob\_intr\_when\_compiled, [1014](#)
- cob\_intr\_year\_to\_yyyy, [1014](#)
- d2, [1015](#)
- d3, [1016](#)
- d4, [1016](#)
- d5, [1016](#)
- DEPTH\_LEVEL, [963](#)
- intrinsic.h
  - cob\_intr\_abs, [1018](#)
  - cob\_intr\_acos, [1019](#)
  - cob\_intr\_annuity, [1019](#)
  - cob\_intr\_asin, [1020](#)
  - cob\_intr\_atan, [1020](#)
  - cob\_intr\_binop, [1021](#)
  - cob\_intr\_char, [1022](#)
  - cob\_intr\_combined\_datetime, [1022](#)
  - cob\_intr\_concatenate, [1023](#)
  - cob\_intr\_cos, [1024](#)
  - cob\_intr\_current\_date, [1024](#)
  - cob\_intr\_date\_of\_integer, [1026](#)
  - cob\_intr\_date\_to\_yyyymmdd, [1027](#)
  - cob\_intr\_day\_of\_integer, [1028](#)
  - cob\_intr\_day\_to\_yyyyddd, [1029](#)
  - cob\_intr\_exception\_file, [1030](#)
  - cob\_intr\_exception\_location, [1030](#)
  - cob\_intr\_exception\_statement, [1031](#)
  - cob\_intr\_exception\_status, [1031](#)
  - cob\_intr\_exp, [1032](#)
  - cob\_intr\_exp10, [1032](#)
  - cob\_intr\_factorial, [1033](#)
  - cob\_intr\_fraction\_part, [1033](#)
  - cob\_intr\_integer, [1033](#)
  - cob\_intr\_integer\_of\_date, [1034](#)
  - cob\_intr\_integer\_of\_day, [1035](#)
  - cob\_intr\_integer\_part, [1036](#)
  - cob\_intr\_lcl\_time\_from\_secs, [1036](#)
  - cob\_intr\_length, [1038](#)
  - cob\_intr\_locale\_date, [1039](#)
  - cob\_intr\_locale\_time, [1041](#)
  - cob\_intr\_log, [1043](#)
  - cob\_intr\_log10, [1044](#)
  - cob\_intr\_lower\_case, [1044](#)
  - cob\_intr\_max, [1044](#)
  - cob\_intr\_mean, [1045](#)
  - cob\_intr\_median, [1046](#)
  - cob\_intr\_midrange, [1046](#)
  - cob\_intr\_min, [1047](#)
  - cob\_intr\_mod, [1047](#)
  - cob\_intr\_numval, [1048](#)
  - cob\_intr\_numval\_c, [1049](#)
  - cob\_intr\_ord, [1051](#)
  - cob\_intr\_ord\_max, [1051](#)
  - cob\_intr\_ord\_min, [1052](#)
  - cob\_intr\_present\_value, [1053](#)
  - cob\_intr\_random, [1053](#)
  - cob\_intr\_range, [1054](#)

- cob\_intr\_rem, [1055](#)
- cob\_intr\_reverse, [1056](#)
- cob\_intr\_seconds\_from\_formatted\_time, [1056](#)
- cob\_intr\_seconds\_past\_midnight, [1057](#)
- cob\_intr\_sign, [1057](#)
- cob\_intr\_sin, [1058](#)
- cob\_intr\_sqrt, [1059](#)
- cob\_intr\_standard\_deviation, [1059](#)
- cob\_intr\_stored\_char\_length, [1060](#)
- cob\_intr\_substitute, [1060](#)
- cob\_intr\_substitute\_case, [1062](#)
- cob\_intr\_sum, [1064](#)
- cob\_intr\_tan, [1064](#)
- cob\_intr\_test\_date\_yyyymmdd, [1065](#)
- cob\_intr\_test\_day\_yyyydd, [1066](#)
- cob\_intr\_trim, [1066](#)
- cob\_intr\_upper\_case, [1067](#)
- cob\_intr\_variance, [1067](#)
- cob\_intr\_when\_compiled, [1069](#)
- cob\_intr\_year\_to\_yyyy, [1069](#)
- INVALID
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [346](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [484](#)
- INVALID\_KEY
  - parser.c, [282](#), [293](#)
  - parser.h, [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- INVALID\_KEY
  - parser.c, [244](#)
  - parser.h, [347](#)
- IS
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- is\_entry
  - cb\_label, [57](#)
- is\_global
  - cb\_label, [57](#)
- is\_section
  - cb\_label, [57](#)
- is\_system
  - cb\_call, [23](#)
- item
  - cobitem, [111](#)
- items
  - cb\_word, [91](#)
- JUSTIFIED
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- KEY
  - parser.c, [244](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- key
  - cb\_alt\_key, [19](#)
  - cb\_field::cb\_key, [55](#)
  - cb\_file, [43](#)
  - indexed\_file, [123](#)
- key\_index
  - indexed\_file, [123](#)
- keys
  - cb\_field, [36](#)
  - cob\_file, [101](#)
- I
  - YYSTYPE, [136](#)
- LABEL
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- label\_list
  - cb\_program, [76](#)
- LAST
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- last
  - memory\_struct, [128](#)
- LAST\_DETAIL
  - parser.c, [282](#), [293](#)
  - parser.h, [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- LAST\_DETAIL
  - parser.c, [245](#)
  - parser.h, [347](#)
- last\_dupno
  - indexed\_file, [124](#)

- last\_key
  - indexed\_file, [124](#)
- last\_open\_mode
  - cob\_file, [101](#)
- last\_readkey
  - indexed\_file, [124](#)
- latbot
  - cb\_file, [43](#)
  - linage\_struct, [126](#)
- latfoot
  - cb\_file, [43](#)
  - linage\_struct, [126](#)
- lattop
  - cb\_file, [43](#)
  - linage\_struct, [126](#)
- LE
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- LEADING
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- LEFT
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [347](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- len
  - pplex.c, [413](#)
  - scanner.c, [507](#)
- LENGTH
  - parser.c, [245](#), [282](#), [293](#)
  - parser.h, [348](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- length
  - cb\_intrinsic, [52](#)
  - cb\_reference, [80](#)
- lenstr
  - cb\_picture, [69](#)
- LESS
  - parser.c, [245](#), [282](#), [294](#)
  - parser.h, [348](#), [376](#), [387](#)
  - ppparse.c, [438](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- level
  - cb\_field, [36](#)
- level\_except\_error
  - error.c, [195](#)
  - tree.h, [707](#)
- level\_redundant\_error
  - error.c, [195](#)
  - tree.h, [708](#)
- level\_require\_error
  - error.c, [195](#)
  - tree.h, [708](#)
- lib/ Directory Reference, [11](#)
- lib/dummy.c, [809](#)
- lib/getopt.c, [809](#)
- lib/getopt.h, [824](#)
- lib/getopt1.c, [826](#)
- lib/gettext.h, [828](#)
- libcob.h, [832](#)
- libcob/ Directory Reference, [12](#)
- libcob/byteswap.h, [833](#)
- libcob/call.c, [834](#)
- libcob/call.h, [845](#)
- libcob/coblocal.h, [854](#)
- libcob/codegen.h, [862](#)
- libcob/common.c, [871](#)
- libcob/common.h, [875](#)
- libcob/fileio.c, [892](#)
- libcob/fileio.h, [923](#)
- libcob/intrinsic.c, [960](#)
- libcob/intrinsic.h, [1016](#)
- libcob/move.c, [1070](#)
- libcob/move.h, [1075](#)
- libcob/numeric.c, [1079](#)
- libcob/numeric.h, [1092](#)
- libcob/screenio.c, [1104](#)
- libcob/screenio.h, [1112](#)
- libcob/strings.c, [1123](#)
- libcob/strings.h, [1134](#)
- libcob/termio.c, [1143](#)
- libcob/termio.h, [1146](#)
- likely
  - common.h, [886](#)
- LIMIT
  - parser.c, [245](#), [282](#), [294](#)
  - parser.h, [348](#), [376](#), [387](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [473](#), [485](#)
- LIMITS
  - parser.c, [245](#), [282](#), [294](#)
  - parser.h, [348](#), [376](#), [387](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [473](#), [485](#)

- lin\_bot
  - linage\_struct, 126
- lin\_foot
  - linage\_struct, 126
- lin\_lines
  - linage\_struct, 126
- lin\_top
  - linage\_struct, 126
- LINAGE
  - parser.c, 246, 282, 294
  - parser.h, 348, 376, 387
  - ppparse.c, 439, 450
  - ppparse.h, 473, 485
- linage
  - cb\_file, 43
  - linage\_struct, 126
- LINAGE\_COUNTER
  - parser.c, 282, 294
  - parser.h, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 473, 485
- LINAGE\_COUNTER
  - parser.c, 246
  - parser.h, 348
- linage\_ctr
  - cb\_file, 43
  - linage\_struct, 126
- linage\_struct, 124
  - latbot, 126
  - latfoot, 126
  - lattop, 126
  - lin\_bot, 126
  - lin\_foot, 126
  - lin\_lines, 126
  - lin\_top, 126
  - linage, 126
  - linage\_ctr, 126
- LINE
  - parser.c, 246, 282, 294
  - parser.h, 348, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- line
  - \_\_cob\_screen, 14
- LINES
  - parser.c, 246, 282, 294
  - parser.h, 348, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- LINKAGE
  - parser.c, 246, 282, 294
  - parser.h, 348, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- linkage\_storage
  - cb\_program, 76
- linorkeyptr
  - cob\_file, 101
- list
  - cb\_class\_name, 26
  - cb\_locale\_name, 64
- LITERAL
  - parser.c, 246, 282, 294
  - parser.h, 348, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- LOCAL\_STORAGE
  - parser.c, 282, 294
  - parser.h, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- local\_file\_list
  - cb\_program, 76
- local\_filename, 127
  - local\_fp, 127
  - local\_name, 127
  - next, 127
- local\_fp
  - local\_filename, 127
- local\_name
  - local\_filename, 127
- LOCAL\_STORAGE
  - parser.c, 246
  - parser.h, 349
- local\_storage
  - cb\_program, 76
- local\_storage\_file
  - cb\_program, 76
- local\_storage\_name
  - cb\_program, 77
- LOCALE
  - parser.c, 246, 282, 294
  - parser.h, 349, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485
- LOCALE\_DT\_FUNC
  - parser.c, 282, 294
  - parser.h, 376, 388
  - ppparse.c, 439, 450
  - ppparse.h, 474, 485

- LOCALE\_DT\_FUNC
  - parser.c, [246](#)
  - parser.h, [349](#)
- locale\_list
  - cb\_program, [77](#)
- LOCALEDIR
  - defaults.h, [808](#)
- localfile
  - filename, [119](#)
- LOCK
  - parser.c, [246](#), [282](#), [294](#)
  - parser.h, [349](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- lock\_mode
  - cb\_file, [43](#)
  - cob\_file, [101](#)
- lookup\_intrinsic
  - reserved.c, [494](#)
  - tree.h, [708](#)
- lookup\_reserved\_word
  - reserved.c, [495](#)
  - tree.h, [708](#)
- lookup\_system\_name
  - reserved.c, [495](#)
  - tree.h, [709](#)
- loop\_counter
  - cb\_program, [77](#)
- LOW\_VALUE
  - parser.c, [282](#), [294](#)
  - parser.h, [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- low\_val\_char
  - cb\_alphabet\_name, [18](#)
- LOW\_VALUE
  - parser.c, [247](#)
  - parser.h, [349](#)
- LOWER\_CASE\_FUNC
  - parser.c, [282](#), [294](#)
  - parser.h, [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- LOWER\_CASE\_FUNC
  - parser.c, [247](#)
  - parser.h, [349](#)
- LOWLIGHT
  - parser.c, [247](#), [282](#), [294](#)
  - parser.h, [349](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
- ppparse.h, [474](#), [485](#)
- lit\_dlclose
  - call.c, [836](#)
- lit\_dlerror
  - call.c, [837](#)
- lit\_dlhandle
  - call.c, [837](#)
- lit\_dlopen
  - call.c, [837](#)
- lit\_dlsym
  - call.c, [837](#)
- main
  - cobc.c, [147](#)
  - cobcrun.c, [139](#)
  - cpucheck.c, [806](#)
- mainpage.h, [1148](#)
- MANUAL
  - parser.c, [247](#), [282](#), [294](#)
  - parser.h, [349](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- MEMORY
  - parser.c, [247](#), [282](#), [294](#)
  - parser.h, [349](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- memory
  - cobsort, [114](#)
- memory\_size
  - cb\_field, [36](#)
- memory\_struct, [128](#)
  - count, [128](#)
  - first, [128](#)
  - last, [128](#)
- MERGE
  - parser.c, [247](#), [283](#), [294](#)
  - parser.h, [349](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- MINUS
  - parser.c, [247](#), [283](#), [294](#)
  - parser.h, [350](#), [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)
- MNEMONIC\_NAME
  - parser.c, [283](#), [294](#)
  - parser.h, [376](#), [388](#)
  - ppparse.c, [439](#), [450](#)
  - ppparse.h, [474](#), [485](#)

- MNEMONIC\_NAME
  - parser.c, [247](#)
  - parser.h, [350](#)
- MODE
  - parser.c, [247](#), [283](#), [294](#)
  - parser.h, [350](#), [376](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [485](#)
- MOVE
  - parser.c, [247](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [485](#)
- move.c
  - cob\_get\_int, [1071](#)
  - cob\_get\_long\_long, [1072](#)
  - cob\_init\_move, [1072](#)
  - cob\_move, [1072](#)
  - cob\_set\_int, [1075](#)
- move.h
  - cob\_get\_int, [1076](#)
  - cob\_move, [1076](#)
  - cob\_set\_int, [1079](#)
- MULTIPLE
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [485](#)
- MULTIPLY
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [485](#)
- N\_
  - gettext.h, [831](#)
- name
  - call\_hash, [16](#)
  - cb\_alphabet\_name, [18](#)
  - cb\_call, [23](#)
  - cb\_class\_name, [26](#)
  - cb\_exception, [29](#)
  - cb\_field, [36](#)
  - cb\_file, [43](#)
  - cb\_funcall, [45](#)
  - cb\_intrinsic, [52](#)
  - cb\_intrinsic\_table, [54](#)
  - cb\_label, [57](#)
  - cb\_locale\_name, [64](#)
  - cb\_perform\_varying, [67](#)
  - cb\_statement, [86](#)
  - cb\_word, [91](#)
  - cob\_exception, [94](#)
  - config.c, [190](#)
  - option, [130](#)
  - reserved, [131](#)
  - reserved.c, [496](#)
- NATIONAL
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- NATIONAL\_EDITED
  - parser.c, [283](#), [294](#)
  - parser.h, [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- NATIONAL\_EDITED
  - parser.c, [248](#)
  - parser.h, [350](#)
- NATIVE
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- NE
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [350](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- need\_assemble
  - filename, [119](#)
- need\_begin
  - cb\_label, [57](#)
- need\_preprocess
  - filename, [119](#)
- need\_return
  - cb\_label, [57](#)
- need\_terminator
  - cb\_statement, [86](#)
- need\_translate
  - filename, [119](#)
- NEGATIVE
  - parser.c, [248](#), [283](#), [294](#)
  - parser.h, [351](#), [377](#), [388](#)
  - ppparse.c, [439](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- nested\_level
  - cb\_program, [77](#)
- new\_text

- cb\_replace\_list, 82
- NEXT
  - parser.c, 248, 283, 294
  - parser.h, 351, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- next
  - \_\_cob\_screen, 15
  - call\_hash, 16
  - cb\_alt\_key, 19
  - cb\_level\_78, 60
  - cb\_replace\_list, 82
  - cb\_text\_list, 89
  - cb\_word, 92
  - cob\_alloc\_cache, 92
  - cob\_external, 95
  - cob\_module, 109
  - cobitem, 111
  - filename, 119
  - local\_filename, 127
  - noreserve, 129
  - sort\_list, 131
  - struct\_handle, 132
- NEXT\_SENTENCE
  - parser.c, 283, 294
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- next\_program
  - cb\_program, 77
- NEXT\_SENTENCE
  - parser.c, 248
  - parser.h, 351
- ngettext
  - gettext.h, 831
- nkeys
  - cb\_field, 36
  - cob\_file, 101
- NO
  - parser.c, 248, 283, 294
  - parser.h, 351, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NO\_ADVANCING
  - parser.c, 283, 295
  - parser.h, 377, 388
  - ppparse.c, 440, 451
  - ppparse.h, 474, 486
- NO\_ADVANCING
  - parser.c, 249
- parser.h, 351
- no\_argument
  - getopt.h, 825
- node
  - reserved.c, 496
- non\_const\_word
  - parser.c, 308
  - tree.h, 722
- NONOPTION\_P
  - getopt.c, 811
- noreserve, 129
  - next, 129
  - noesword, 129
- noestab
  - cobc.h, 171
  - config.c, 190
- noesword
  - noreserve, 129
- NOT
  - parser.c, 249, 283, 294
  - parser.h, 351, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_END
  - parser.c, 283, 294
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_EOP
  - parser.c, 283, 294
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_EXCEPTION
  - parser.c, 283, 294
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_INVALID\_KEY
  - parser.c, 283, 294
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_OVERFLOW
  - parser.c, 283, 295
  - parser.h, 377, 388
  - ppparse.c, 439, 451
  - ppparse.h, 474, 486
- NOT\_SIZE\_ERROR
  - parser.c, 283, 295

- parser.h, [377](#), [388](#)
- ppparse.c, [440](#), [451](#)
- ppparse.h, [474](#), [486](#)
- NOT\_END
  - parser.c, [249](#)
  - parser.h, [351](#)
- NOT\_EOP
  - parser.c, [249](#)
  - parser.h, [351](#)
- NOT\_EXCEPTION
  - parser.c, [249](#)
  - parser.h, [351](#)
- NOT\_INVALID\_KEY
  - parser.c, [249](#)
  - parser.h, [351](#)
- NOT\_OVERFLOW
  - parser.c, [249](#)
  - parser.h, [352](#)
- NOT\_SIZE\_ERROR
  - parser.c, [249](#)
  - parser.h, [352](#)
- NULL
  - getopt1.c, [827](#)
- null\_check
  - cb\_statement, [86](#)
- NUM\_INTRINSICS
  - reserved.c, [493](#)
- NUM\_PREFIX
  - fileio.c, [896](#)
- NUM\_RESERVED\_WORDS
  - reserved.c, [493](#)
- NUMBER
  - parser.c, [249](#), [283](#), [295](#)
  - parser.h, [352](#), [377](#), [388](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- NUMBERS
  - parser.c, [249](#), [283](#), [295](#)
  - parser.h, [352](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [474](#), [486](#)
- NUMERIC
  - parser.c, [250](#), [283](#), [295](#)
  - parser.h, [352](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- numeric.c
  - cob\_add, [1081](#)
  - cob\_add\_int, [1081](#)
  - cob\_cmp\_int, [1082](#)
  - cob\_cmp\_long\_numdisp, [1082](#)
  - cob\_cmp\_long\_sign\_numdisp, [1083](#)
  - cob\_cmp\_numdisp, [1083](#)
  - cob\_cmp\_packed, [1083](#)
  - cob\_cmp\_sign\_numdisp, [1085](#)
  - cob\_cmp\_uint, [1085](#)
  - cob\_decimal\_add, [1085](#)
  - cob\_decimal\_cmp, [1086](#)
  - cob\_decimal\_div, [1086](#)
  - cob\_decimal\_get\_field, [1086](#)
  - cob\_decimal\_init, [1087](#)
  - cob\_decimal\_mul, [1088](#)
  - cob\_decimal\_pow, [1088](#)
  - cob\_decimal\_set\_field, [1088](#)
  - cob\_decimal\_sub, [1089](#)
  - cob\_div\_quotient, [1089](#)
  - cob\_div\_remainder, [1089](#)
  - cob\_init\_numeric, [1090](#)
  - COB\_LIB\_INCLUDE, [1081](#)
  - COB\_MAX\_BINARY, [1081](#)
  - cob\_numeric\_cmp, [1090](#)
  - cob\_set\_packed\_int, [1090](#)
  - cob\_set\_packed\_zero, [1091](#)
  - cob\_sub, [1091](#)
  - cob\_sub\_int, [1091](#)
  - DECIMAL\_CHECK, [1081](#)
  - DECIMAL\_NAN, [1081](#)
- numeric.h
  - cob\_add, [1094](#)
  - cob\_add\_int, [1094](#)
  - cob\_cmp\_int, [1094](#)
  - cob\_cmp\_long\_numdisp, [1095](#)
  - cob\_cmp\_long\_sign\_numdisp, [1095](#)
  - cob\_cmp\_numdisp, [1096](#)
  - cob\_cmp\_packed, [1096](#)
  - cob\_cmp\_sign\_numdisp, [1097](#)
  - cob\_cmp\_uint, [1098](#)
  - cob\_decimal\_add, [1098](#)
  - cob\_decimal\_cmp, [1098](#)
  - cob\_decimal\_div, [1098](#)
  - cob\_decimal\_get\_field, [1099](#)
  - cob\_decimal\_init, [1100](#)
  - cob\_decimal\_mul, [1100](#)
  - cob\_decimal\_pow, [1100](#)
  - cob\_decimal\_set\_field, [1101](#)
  - cob\_decimal\_sub, [1101](#)
  - cob\_div\_quotient, [1101](#)
  - cob\_div\_remainder, [1102](#)
  - cob\_set\_packed\_int, [1102](#)
  - cob\_set\_packed\_zero, [1103](#)





- getopt.h, [826](#)
- opterr
  - getopt.c, [823](#)
  - getopt.h, [826](#)
- optimize\_flag
  - cobc.c, [154](#)
  - cobc.h, [171](#)
- optind
  - getopt.c, [823](#)
  - getopt.h, [826](#)
- option, [130](#)
  - flag, [130](#)
  - has\_arg, [130](#)
  - name, [130](#)
  - val, [130](#)
- OPTIONAL
  - parser.c, [251](#), [283](#), [295](#)
  - parser.h, [353](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- optional
  - cb\_file, [43](#)
- optional\_argument
  - getopt.h, [825](#)
- optopt
  - getopt.c, [823](#)
  - getopt.h, [826](#)
- OR
  - parser.c, [251](#), [283](#), [295](#)
  - parser.h, [353](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- ORDER
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [353](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- ORGANIZATION
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [353](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- organization
  - cb\_file, [43](#)
  - cob\_file, [101](#)
- orig
  - cb\_picture, [69](#)
- orig\_name
  - cb\_label, [58](#)
- orig\_source\_name
  - cb\_program, [77](#)
- OTHER
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [353](#), [377](#), [389](#)
  - ppparse.c, [440](#), [451](#)
  - ppparse.h, [475](#), [486](#)
- OUTPUT
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [354](#), [377](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [486](#)
- OVERFLOW
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [354](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [486](#)
- OVERLINE
  - parser.c, [251](#), [284](#), [295](#)
  - parser.h, [354](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [486](#)
- PACKAGE
  - config.h, [804](#)
- PACKAGE\_BUGREPORT
  - config.h, [804](#)
- PACKAGE\_NAME
  - config.h, [804](#)
- PACKAGE\_STRING
  - config.h, [804](#)
- PACKAGE\_TARNAME
  - config.h, [804](#)
- PACKAGE\_VERSION
  - config.h, [804](#)
- PACKED\_DECIMAL
  - parser.c, [284](#), [295](#)
  - parser.h, [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [486](#)
- PACKED\_DECIMAL
  - parser.c, [251](#)
  - parser.h, [354](#)
- PADDING
  - parser.c, [252](#), [284](#), [295](#)
  - parser.h, [354](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PAGE
  - parser.c, [252](#), [284](#), [295](#)
  - parser.h, [354](#), [378](#), [389](#)

- ppparse.c, [440](#), [452](#)
- ppparse.h, [475](#), [487](#)
- PAGE\_FOOTING
  - parser.c, [284](#), [295](#)
  - parser.h, [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PAGE\_HEADING
  - parser.c, [284](#), [295](#)
  - parser.h, [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PAGE\_FOOTING
  - parser.c, [252](#)
  - parser.h, [354](#)
- PAGE\_HEADING
  - parser.c, [252](#)
  - parser.h, [354](#)
- PARAGRAPH
  - parser.c, [252](#), [284](#), [295](#)
  - parser.h, [354](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- param\_num
  - cb\_field, [37](#)
- parameter\_list
  - cb\_program, [77](#)
- parent
  - cb\_field, [37](#)
- parser.c
  - ACCEPT, [224](#), [276](#), [288](#)
  - ACCESS, [224](#), [276](#), [288](#)
  - ADD, [224](#), [276](#), [288](#)
  - ADDRESS, [224](#), [276](#), [288](#)
  - ADVANCING, [224](#), [276](#), [288](#)
  - AFTER, [224](#), [276](#), [288](#)
  - ALL, [224](#), [277](#), [288](#)
  - ALLOCATE, [224](#), [277](#), [288](#)
  - ALPHABET, [224](#), [277](#), [288](#)
  - ALPHABETIC, [224](#), [277](#), [288](#)
  - ALPHABETIC\_LOWER, [277](#), [288](#)
  - ALPHABETIC\_UPPER, [277](#), [288](#)
  - ALPHABETIC\_LOWER, [225](#)
  - ALPHABETIC\_UPPER, [225](#)
  - ALPHANUMERIC, [225](#), [277](#), [288](#)
  - ALPHANUMERIC\_EDITED, [277](#), [288](#)
  - ALPHANUMERIC\_EDITED, [225](#)
  - ALSO, [225](#), [277](#), [288](#)
  - ALTER, [225](#), [277](#), [288](#)
  - ALTERNATE, [225](#), [277](#), [288](#)
  - AND, [225](#), [277](#), [288](#)
  - ANY, [225](#), [277](#), [288](#)
  - ARE, [225](#), [277](#), [288](#)
  - AREA, [226](#), [277](#), [288](#)
  - ARGUMENT\_NUMBER, [277](#), [288](#)
  - ARGUMENT\_VALUE, [277](#), [288](#)
  - ARGUMENT\_NUMBER, [226](#)
  - ARGUMENT\_VALUE, [226](#)
  - AS, [226](#), [277](#), [288](#)
  - ASCENDING, [226](#), [277](#), [288](#)
  - ASSIGN, [226](#), [277](#), [288](#)
  - AT, [226](#), [277](#), [289](#)
  - AUTO, [226](#), [277](#), [289](#)
  - AUTOMATIC, [226](#), [277](#), [289](#)
  - BACKGROUND\_COLOR, [277](#), [289](#)
  - BACKGROUND\_COLOR, [226](#)
  - BASED, [227](#), [277](#), [289](#)
  - BEFORE, [227](#), [277](#), [289](#)
  - BELL, [227](#), [277](#), [289](#)
  - BINARY, [227](#), [277](#), [289](#)
  - BINARY\_C\_LONG, [277](#), [289](#)
  - BINARY\_CHAR, [277](#), [289](#)
  - BINARY\_DOUBLE, [277](#), [289](#)
  - BINARY\_LONG, [277](#), [289](#)
  - BINARY\_SHORT, [277](#), [289](#)
  - BINARY\_C\_LONG, [227](#)
  - BINARY\_CHAR, [227](#)
  - BINARY\_DOUBLE, [227](#)
  - BINARY\_LONG, [227](#)
  - BINARY\_SHORT, [227](#)
  - BLANK, [227](#), [277](#), [289](#)
  - BLANK\_LINE, [277](#), [289](#)
  - BLANK\_SCREEN, [277](#), [289](#)
  - BLANK\_LINE, [228](#)
  - BLANK\_SCREEN, [228](#)
  - BLINK, [228](#), [277](#), [289](#)
  - BLOCK, [228](#), [277](#), [289](#)
  - BOTTOM, [228](#), [278](#), [289](#)
  - BY, [228](#), [278](#), [289](#), [300](#)
  - BYTE\_LENGTH, [278](#), [289](#)
  - BYTE\_LENGTH, [228](#)
  - CALL, [228](#), [278](#), [289](#)
  - CANCEL, [228](#), [278](#), [289](#)
  - CH, [228](#), [278](#), [289](#)
  - CHAINING, [229](#), [278](#), [289](#)
  - CHARACTER, [229](#), [278](#), [289](#)
  - CHARACTERS, [229](#), [278](#), [289](#)
  - CLASS, [229](#), [278](#), [289](#)
  - CLOSE, [229](#), [278](#), [289](#)
  - CODE, [229](#), [278](#), [289](#)

CODE\_SET, 278, 289  
CODE\_SET, 229  
COL, 229, 278, 289  
COLLATING, 229, 278, 289  
COLS, 229, 278, 289  
COLUMN, 230, 278, 289  
COLUMNS, 230, 278, 289  
COMMA, 230, 278, 289  
COMMA\_DELIM, 278, 290  
COMMA\_DELIM, 230  
COMMAND\_LINE, 278, 289  
COMMAND\_LINE, 230  
COMMIT, 230, 278, 290  
COMMON, 230, 278, 290  
COMP, 230, 278, 290  
COMP\_1, 278, 290  
COMP\_2, 278, 290  
COMP\_3, 278, 290  
COMP\_4, 278, 290  
COMP\_5, 278, 290  
COMP\_X, 278, 290  
COMP\_1, 230  
COMP\_2, 230  
COMP\_3, 231  
COMP\_4, 231  
COMP\_5, 231  
COMP\_X, 231  
COMPUTE, 231, 278, 290  
CONCATENATE\_FUNC, 278, 290  
CONCATENATE\_FUNC, 231  
CONFIGURATION, 231, 278, 290  
CONSTANT, 231, 278, 290  
CONTAINS, 231, 278, 290  
CONTENT, 231, 278, 290  
CONTINUE, 232, 278, 290  
CONTROL, 232, 278, 290  
CONTROL\_FOOTING, 279, 290  
CONTROL\_HEADING, 279, 290  
CONTROL\_FOOTING, 232  
CONTROL\_HEADING, 232  
CONTROLS, 232, 279, 290  
CONVERTING, 232, 279, 290  
COPY, 299, 300  
CORRESPONDING, 232, 279, 290  
COUNT, 232, 279, 290  
CRT, 232, 279, 290  
CURRENCY, 232, 279, 290  
CURRENT\_DATE\_FUNC, 279, 290  
CURRENT\_DATE\_FUNC, 233  
current\_paragraph, 307  
current\_program, 307  
current\_section, 307  
current\_statement, 307  
CURSOR, 233, 279, 290  
CYCLE, 233, 279, 290  
DATA, 233, 279, 290  
DATE, 233, 279, 290  
DAY, 233, 279, 290  
DAY\_OF\_WEEK, 279, 290  
DAY\_OF\_WEEK, 233  
DE, 233, 279, 290  
DEBUGGING, 233, 279, 290  
DECIMAL\_POINT, 279, 290  
DECIMAL\_POINT, 233  
DECLARATIVES, 234, 279, 290  
DEFAULT, 234, 279, 290  
DELETE, 234, 279, 291  
DELIMITED, 234, 279, 291  
DELIMITER, 234, 279, 291  
DEPENDING, 234, 279, 291  
DESCENDING, 234, 279, 291  
DETAIL, 234, 279, 291  
DISK, 234, 279, 291  
DISPLAY, 234, 279, 291  
DIVIDE, 235, 279, 291  
DIVISION, 235, 279, 291  
DOWN, 235, 279, 291  
DUPLICATES, 235, 279, 291  
DYNAMIC, 235, 279, 291  
EBCDIC, 235, 279, 291  
ELSE, 235, 279, 291  
emit\_statement, 235  
END, 235, 279, 291  
END\_ACCEPT, 279, 291  
END\_ADD, 279, 291  
END\_CALL, 280, 291  
END\_COMPUTE, 280, 291  
END\_DELETE, 280, 291  
END\_DISPLAY, 280, 291  
END\_DIVIDE, 280, 291  
END\_EVALUATE, 280, 291  
END\_FUNCTION, 280, 291  
END\_IF, 280, 291  
END\_MULTIPLY, 280, 291  
END\_PERFORM, 280, 291  
END\_PROGRAM, 280, 291  
END\_READ, 280, 291  
END\_RETURN, 280, 291  
END\_REWRITE, 280, 291  
END\_SEARCH, 280, 291

- END\_START, 280, 291
- END\_STRING, 280, 291
- END\_SUBTRACT, 280, 291
- END\_UNSTRING, 280, 291
- END\_WRITE, 280, 291
- END\_ACCEPT, 235
- END\_ADD, 236
- END\_CALL, 236
- END\_COMPUTE, 236
- END\_DELETE, 236
- END\_DISPLAY, 236
- END\_DIVIDE, 236
- END\_EVALUATE, 236
- END\_FUNCTION, 236
- END\_IF, 236
- END\_MULTIPLY, 236
- END\_PERFORM, 237
- END\_PROGRAM, 237
- END\_READ, 237
- END\_RETURN, 237
- END\_REWRITE, 237
- END\_SEARCH, 237
- END\_START, 237
- END\_STRING, 237
- END\_SUBTRACT, 237
- END\_UNSTRING, 237
- END\_WRITE, 238
- ENTRY, 238, 280, 292
- ENVIRONMENT, 238, 280, 292
- ENVIRONMENT\_NAME, 280, 292
- ENVIRONMENT\_VALUE, 280, 292
- ENVIRONMENT\_NAME, 238
- ENVIRONMENT\_VALUE, 238
- EOL, 238, 280, 292
- EOP, 238, 280, 292
- EOS, 238, 280, 292
- ESEQ, 300
- EQUAL, 238, 280, 292
- EQUALS, 238, 280, 292
- ERASE, 239, 280, 292
- ERROR, 239, 280, 292
- ESCAPE, 239, 280, 292
- EVALUATE, 239, 280, 292
- EVENT\_STATUS, 280, 292
- EVENT\_STATUS, 239
- EXCEPTION, 239, 280, 292
- EXCLUSIVE, 239, 280, 292
- EXIT, 239, 280, 292
- EXTEND, 239, 280, 292
- EXTERNAL, 239, 281, 292
- FD, 240, 281, 292
- FILE\_CONTROL, 281, 292
- FILE\_ID, 281, 292
- FILE\_CONTROL, 240
- FILE\_ID, 240
- FILLER, 240, 281, 292
- FINAL, 240, 281, 292
- FIRST, 240, 281, 292
- FOOTING, 240, 281, 292
- FOR, 240, 281, 292
- BACKGROUND\_COLOR, 281, 292
- BACKGROUND\_COLOR, 240
- FOREVER, 240, 281, 292
- FREE, 241, 281, 292
- FROM, 241, 281, 292
- FULL, 241, 281, 292
- FUNCTION, 241, 281, 292
- FUNCTION\_ID, 281, 292
- FUNCTION\_NAME, 281, 292
- FUNCTION\_ID, 241
- FUNCTION\_NAME, 241
- functions\_are\_all, 308
- GE, 241, 281, 292
- GENERATE, 241, 281, 292
- GIVING, 241, 281, 292
- GLOBAL, 241, 281, 293
- GO, 242, 281, 293
- GOBACK, 242, 281, 293
- GREATER, 242, 281, 293
- GROUP, 242, 281, 293
- HEADING, 242, 281, 293
- HIGH\_VALUE, 281, 293
- HIGH\_VALUE, 242
- HIGHLIGHT, 242, 281, 293
- I\_O, 282, 293
- I\_O\_CONTROL, 282, 293
- I\_O, 242
- I\_O\_CONTROL, 242
- IDENTIFICATION, 242, 281, 293
- IF, 243, 281, 293
- IGNORE, 243, 281, 293
- IGNORING, 243, 281, 293
- IN, 243, 281, 293, 299, 300
- INDEX, 243, 281, 293
- INDEXED, 243, 281, 293
- INDICATE, 243, 281, 293
- INITIALIZE, 243, 281, 293
- INITIALIZED, 243, 281, 293
- INITIATE, 243, 282, 293
- INPUT, 244, 282, 293

INPUT\_OUTPUT, 282, 293  
INPUT\_OUTPUT, 244  
INSPECT, 244, 282, 293  
INTO, 244, 282, 293  
INTRINSIC, 244, 282, 293  
INVALID, 244, 282, 293  
INVALID\_KEY, 282, 293  
INVALID\_KEY, 244  
IS, 244, 282, 293  
JUSTIFIED, 244, 282, 293  
KEY, 244, 282, 293  
LABEL, 245, 282, 293  
LAST, 245, 282, 293  
LAST\_DETAIL, 282, 293  
LAST\_DETAIL, 245  
LE, 245, 282, 293  
LEADING, 245, 282, 293  
LEFT, 245, 282, 293  
LENGTH, 245, 282, 293  
LESS, 245, 282, 294  
LIMIT, 245, 282, 294  
LIMITS, 245, 282, 294  
LINAGE, 246, 282, 294  
LINAGE\_COUNTER, 282, 294  
LINAGE\_COUNTER, 246  
LINE, 246, 282, 294  
LINES, 246, 282, 294  
LINKAGE, 246, 282, 294  
LITERAL, 246, 282, 294  
LOCAL\_STORAGE, 282, 294  
LOCAL\_STORAGE, 246  
LOCALE, 246, 282, 294  
LOCALE\_DT\_FUNC, 282, 294  
LOCALE\_DT\_FUNC, 246  
LOCK, 246, 282, 294  
LOW\_VALUE, 282, 294  
LOW\_VALUE, 247  
LOWER\_CASE\_FUNC, 282, 294  
LOWER\_CASE\_FUNC, 247  
LOWLIGHT, 247, 282, 294  
MANUAL, 247, 282, 294  
MEMORY, 247, 282, 294  
MERGE, 247, 283, 294  
MINUS, 247, 283, 294  
MNEMONIC\_NAME, 283, 294  
MNEMONIC\_NAME, 247  
MODE, 247, 283, 294  
MOVE, 247, 283, 294  
MULTIPLE, 248, 283, 294  
MULTIPLY, 248, 283, 294  
NATIONAL, 248, 283, 294  
NATIONAL\_EDITED, 283, 294  
NATIONAL\_EDITED, 248  
NATIVE, 248, 283, 294  
NE, 248, 283, 294  
NEGATIVE, 248, 283, 294  
NEXT, 248, 283, 294  
NEXT\_SENTENCE, 283, 294  
NEXT\_SENTENCE, 248  
NO, 248, 283, 294  
NO\_ADVANCING, 283, 295  
NO\_ADVANCING, 249  
non\_const\_word, 308  
NOT, 249, 283, 294  
NOT\_END, 283, 294  
NOT\_EOP, 283, 294  
NOT\_EXCEPTION, 283, 294  
NOT\_INVALID\_KEY, 283, 294  
NOT\_OVERFLOW, 283, 295  
NOT\_SIZE\_ERROR, 283, 295  
NOT\_END, 249  
NOT\_EOP, 249  
NOT\_EXCEPTION, 249  
NOT\_INVALID\_KEY, 249  
NOT\_OVERFLOW, 249  
NOT\_SIZE\_ERROR, 249  
NUMBER, 249, 283, 295  
NUMBERS, 249, 283, 295  
NUMERIC, 250, 283, 295  
NUMERIC\_EDITED, 283, 295  
NUMERIC\_EDITED, 250  
NUMVALC\_FUNC, 283, 295  
NUMVALC\_FUNC, 250  
OBJECT\_COMPUTER, 283, 295  
OBJECT\_COMPUTER, 250  
OCCURS, 250, 283, 295  
OF, 250, 283, 295, 299, 300  
OFF, 250, 283, 295, 299, 300  
OMITTED, 250, 283, 295  
ON, 250, 283, 295  
ONLY, 250, 283, 295  
OPEN, 251, 283, 295  
OPTIONAL, 251, 283, 295  
OR, 251, 283, 295  
ORDER, 251, 284, 295  
ORGANIZATION, 251, 284, 295  
OTHER, 251, 284, 295  
OUTPUT, 251, 284, 295  
OVERFLOW, 251, 284, 295  
OVERLINE, 251, 284, 295

- PACKED\_DECIMAL, 284, 295  
PACKED\_DECIMAL, 251  
PADDING, 252, 284, 295  
PAGE, 252, 284, 295  
PAGE\_FOOTING, 284, 295  
PAGE\_HEADING, 284, 295  
PAGE\_FOOTING, 252  
PAGE\_HEADING, 252  
PARAGRAPH, 252, 284, 295  
PENDING, 252  
PERFORM, 252, 284, 295  
PICTURE, 252, 284, 295  
PLUS, 252, 284, 295  
POINTER, 252, 284, 295  
POSITION, 253, 284, 295  
POSITIVE, 253, 284, 295  
PRESENT, 253, 284, 295  
PREVIOUS, 253, 284, 295  
PRINTER, 253, 284, 296  
PRINTING, 253, 284, 296, 299, 300  
PROCEDURE, 253, 284, 296  
PROCEDURES, 253, 284, 296  
PROCEED, 253, 284, 296  
PROGRAM, 253, 284, 296  
PROGRAM\_ID, 284, 296  
PROGRAM\_NAME, 284, 296  
PROGRAM\_POINTER, 284, 296  
PROGRAM\_ID, 254  
PROGRAM\_NAME, 254  
PROGRAM\_POINTER, 254  
PROMPT, 254, 284, 296  
push\_expr, 254  
QUOTE, 254, 284, 296  
RANDOM, 254, 284, 296  
RD, 254, 284, 296  
READ, 254, 284, 296  
RECORD, 254, 284, 296  
RECORDING, 255, 284, 296  
RECORDS, 255, 284, 296  
RECURSIVE, 255, 284, 296  
REDEFINES, 255, 285, 296  
REEL, 255, 285, 296  
REFERENCE, 255, 285, 296  
RELATIVE, 255, 285, 296  
RELEASE, 255, 285, 296  
REMAINDER, 255, 285, 296  
REMOVAL, 255, 285, 296  
RENAMES, 256, 285, 296  
REPLACE, 299, 300  
REPLACING, 256, 285, 296, 299, 300  
REPORT, 256, 285, 296  
REPORT\_FOOTING, 285, 296  
REPORT\_HEADING, 285, 296  
REPORT\_FOOTING, 256  
REPORT\_HEADING, 256  
REPORTING, 256, 285, 296  
REPORTS, 256, 285, 296  
REPOSITORY, 256, 285, 296  
REQUIRED, 256, 285, 296  
RESERVE, 256, 285, 296  
RETURN, 257, 285, 296  
RETURNING, 257, 285, 296  
REVERSE\_FUNC, 285, 296  
REVERSE\_VIDEO, 285, 297  
REVERSE\_FUNC, 257  
REVERSE\_VIDEO, 257  
REWIND, 257, 285, 297  
REWRITE, 257, 285, 297  
RIGHT, 257, 285, 297  
ROLLBACK, 257, 285, 297  
ROUNDED, 257, 285, 297  
RUN, 257, 285, 297  
SAME, 258, 285, 297  
SCREEN, 258, 285, 297  
SCREEN\_CONTROL, 285, 297  
SCREEN\_CONTROL, 258  
SCROLL, 258, 285, 297  
SD, 258, 285, 297  
SEARCH, 258, 285, 297  
SECTION, 258, 285, 297  
SECURE, 258, 285, 297  
SEGMENT\_LIMIT, 285, 297  
SEGMENT\_LIMIT, 258  
SELECT, 258, 285, 297  
SEMI\_COLON, 285, 297  
SEMI\_COLON, 259  
SENTENCE, 259, 286, 297  
SEPARATE, 259, 286, 297  
SEQUENCE, 259, 286, 297  
SEQUENTIAL, 259, 286, 297  
SET, 259, 286, 297  
SHARING, 259, 286, 297  
SIGN, 259, 286, 297  
SIGNED, 259, 286, 297  
SIGNED\_INT, 286, 297  
SIGNED\_LONG, 286, 297  
SIGNED\_SHORT, 286, 297  
SIGNED\_INT, 259  
SIGNED\_LONG, 260  
SIGNED\_SHORT, 260

SIZE, 260, 286, 297  
SIZE\_ERROR, 286, 297  
SIZE\_ERROR, 260  
SORT, 260, 286, 297  
SORT\_MERGE, 286, 297  
SORT\_MERGE, 260  
SOURCE, 260, 286, 297  
SOURCE\_COMPUTER, 286, 297  
SOURCE\_COMPUTER, 260  
SPACE, 260, 286, 297  
SPECIAL\_NAMES, 286, 297  
SPECIAL\_NAMES, 260  
STANDARD, 261, 286, 297  
STANDARD\_1, 286, 298  
STANDARD\_2, 286, 298  
STANDARD\_1, 261  
STANDARD\_2, 261  
START, 261, 286, 298  
STATUS, 261, 286, 298  
STOP, 261, 286, 298  
STRING, 261, 286, 298  
SUBSTITUTE\_CASE\_FUNC, 286, 298  
SUBSTITUTE\_FUNC, 286, 298  
SUBSTITUTE\_CASE\_FUNC, 261  
SUBSTITUTE\_FUNC, 261  
SUBTRACT, 261, 286, 298  
SUM, 262, 286, 298  
SUPPRESS, 262, 286, 298–300  
SYMBOLIC, 262, 286, 298  
SYNCHRONIZED, 262, 286, 298  
TALLYING, 262, 286, 298  
TAPE, 262, 286, 298  
TERM\_ACCEPT, 262  
TERM\_ADD, 262  
TERM\_CALL, 262  
TERM\_COMPUTE, 262  
TERM\_DELETE, 263  
TERM\_DISPLAY, 263  
TERM\_DIVIDE, 263  
TERM\_EVALUATE, 263  
TERM\_IF, 263  
TERM\_MAX, 263  
TERM\_MULTIPLY, 263  
TERM\_NONE, 263  
TERM\_PERFORM, 263  
TERM\_READ, 263  
TERM\_RECEIVE, 264  
TERM\_RETURN, 264  
TERM\_REWRITE, 264  
TERM\_SEARCH, 264  
TERM\_START, 264  
TERM\_STRING, 264  
TERM\_SUBTRACT, 264  
TERM\_UNSTRING, 264  
TERM\_WRITE, 264  
TERMINATE, 264, 286, 298  
TEST, 265, 286, 298  
THAN, 265, 286, 298  
THEN, 265, 287, 298  
THRU, 265, 287, 298  
TIME, 265, 287, 298  
TIMES, 265, 287, 298  
TO, 265, 287, 298  
TOK\_FALSE, 287, 298  
TOK\_FILE, 287, 298  
TOK\_INITIAL, 287, 298  
TOK\_NULL, 287, 298  
TOK\_TRUE, 287, 298  
TOK\_FALSE, 265  
TOK\_FILE, 265  
TOK\_INITIAL, 265  
TOK\_NULL, 266  
TOK\_TRUE, 266  
TOKEN, 300  
TOKEN\_EOF, 276, 288, 299, 300  
TOKEN\_EOF, 266  
TOP, 266, 287, 298  
TRAILING, 266, 287, 298  
TRANSFORM, 266, 287, 298  
TRIM\_FUNCTION, 287, 298  
TRIM\_FUNCTION, 266  
TYPE, 266, 287, 298  
UNARY\_SIGN, 288, 299  
UNARY\_SIGN, 266  
UNDERLINE, 266, 287, 298  
UNIT, 267, 287, 298  
UNLOCK, 267, 287, 298  
UNSIGNED, 267, 287, 298  
UNSIGNED\_INT, 287, 298  
UNSIGNED\_LONG, 287, 299  
UNSIGNED\_SHORT, 287, 299  
UNSIGNED\_INT, 267  
UNSIGNED\_LONG, 267  
UNSIGNED\_SHORT, 267  
UNSTRING, 267, 287, 299  
UNTIL, 267, 287, 299  
UP, 267, 287, 299  
UPDATE, 267, 287, 299  
UPON, 268, 287, 299



- UPON\_ARGUMENT\_NUMBER, 287, 299
- UPON\_COMMAND\_LINE, 287, 299
- UPON\_ENVIRONMENT\_NAME, 287, 299
- UPON\_ENVIRONMENT\_VALUE, 287, 299
- UPON\_ARGUMENT\_NUMBER, 268
- UPON\_COMMAND\_LINE, 268
- UPON\_ENVIRONMENT\_NAME, 268
- UPON\_ENVIRONMENT\_VALUE, 268
- UPPER\_CASE\_FUNC, 287, 299
- UPPER\_CASE\_FUNC, 268
- USAGE, 268, 287, 299
- USE, 268, 287, 299
- USING, 268, 287, 299
- VALUE, 268, 287, 299
- VARYING, 269, 287, 299
- WAIT, 269, 287, 299
- WHEN, 269, 288, 299
- WHEN\_COMPILED\_FUNC, 288, 299
- WHEN\_COMPILED\_FUNC, 269
- WITH, 269, 288, 299
- WORD, 269, 288, 299
- WORDS, 269, 288, 299
- WORKING\_STORAGE, 288, 299
- WORKING\_STORAGE, 269
- WRITE, 269, 288, 299
- YY\_REDUCE\_PRINT, 269
- YY\_STACK\_PRINT, 270
- YYABORT, 270
- YYACCEPT, 270
- YYBACKUP, 270
- YYBISON, 270
- yychar, 308
- yyclearin, 271
- YYCOPY, 271
- YYDEBUG, 271
- yydebug, 308
- YYDPRINTF, 271
- YYDSYMPRINT, 271
- YYDSYMPRINTF, 271
- YYEMPTY, 272
- YYEOF, 272
- YYERRCODE, 272
- yyerrok, 272
- YYERROR, 272
- yyerror, 272
- YYERROR\_VERBOSE, 272
- YYFAIL, 273
- YYFINAL, 273
- YYFPRINTF, 273
- YYINITDEPTH, 273
- YYLAST, 273
- YYLEX, 273
- YYLLOC\_DEFAULT, 273
- YYLSP\_NEEDED, 273
- yylval, 308
- YYMAXDEPTH, 273
- YYMAXUTOK, 274
- yynerres, 308
- YYNNTS, 274
- YYNRULES, 274
- YYNSTATES, 274
- YYNTOKENS, 274
- YYPACT\_NINF, 274
- yyparse, 307
- YYPOPSTACK, 274
- YYPURE, 274
- YYRECOVERING, 274
- yysigned\_char, 276
- YYSIZE\_T, 274
- YYSKELETON\_NAME, 274
- YYSTACK\_ALLOC, 275
- YYSTACK\_BYTES, 275
- YYSTACK\_FREE, 275
- YYSTACK\_GAP\_MAXIMUM, 275
- YYSTACK\_RELOCATE, 275
- YYTABLE\_NINF, 275
- YYTERROR, 275
- YYTOKENTYPE, 275
- yytokentype, 276
- YYTRANSLATE, 276
- YYUNDEFTOK, 276
- YYYYDDD, 276, 288, 299
- YYYYMMDD, 276, 288, 299
- ZERO, 276, 288, 299
- parser.h
  - ACCEPT, 326, 370, 382
  - ACCESS, 326, 370, 382
  - ADD, 326, 370, 382
  - ADDRESS, 326, 370, 382
  - ADVANCING, 326, 370, 382
  - AFTER, 327, 370, 382
  - ALL, 327, 370, 382
  - ALLOCATE, 327, 370, 382
  - ALPHABET, 327, 370, 382
  - ALPHABETIC, 327, 370, 382
  - ALPHABETIC\_LOWER, 371, 382
  - ALPHABETIC\_UPPER, 371, 382

ALPHABETIC\_LOWER, 327  
ALPHABETIC\_UPPER, 327  
ALPHANUMERIC, 327, 371, 382  
ALPHANUMERIC\_EDITED, 371, 382  
ALPHANUMERIC\_EDITED, 327  
ALSO, 327, 371, 382  
ALTER, 328, 371, 382  
ALTERNATE, 328, 371, 382  
AND, 328, 371, 382  
ANY, 328, 371, 382  
ARE, 328, 371, 382  
AREA, 328, 371, 382  
ARGUMENT\_NUMBER, 371, 382  
ARGUMENT\_VALUE, 371, 382  
ARGUMENT\_NUMBER, 328  
ARGUMENT\_VALUE, 328  
AS, 328, 371, 382  
ASCENDING, 328, 371, 382  
ASSIGN, 329, 371, 382  
AT, 329, 371, 382  
AUTO, 329, 371, 382  
AUTOMATIC, 329, 371, 382  
BACKGROUND\_COLOR, 371, 382  
BACKGROUND\_COLOR, 329  
BASED, 329, 371, 383  
BEFORE, 329, 371, 383  
BELL, 329, 371, 383  
BINARY, 329, 371, 383  
BINARY\_C\_LONG, 371, 383  
BINARY\_CHAR, 371, 383  
BINARY\_DOUBLE, 371, 383  
BINARY\_LONG, 371, 383  
BINARY\_SHORT, 371, 383  
BINARY\_C\_LONG, 329  
BINARY\_CHAR, 330  
BINARY\_DOUBLE, 330  
BINARY\_LONG, 330  
BINARY\_SHORT, 330  
BLANK, 330, 371, 383  
BLANK\_LINE, 371, 383  
BLANK\_SCREEN, 371, 383  
BLANK\_LINE, 330  
BLANK\_SCREEN, 330  
BLINK, 330, 371, 383  
BLOCK, 330, 371, 383  
BOTTOM, 330, 371, 383  
BY, 331, 371, 383, 393, 394  
BYTE\_LENGTH, 371, 383  
BYTE\_LENGTH, 331  
CALL, 331, 371, 383  
CANCEL, 331, 372, 383  
CH, 331, 372, 383  
CHAINING, 331, 372, 383  
CHARACTER, 331, 372, 383  
CHARACTERS, 331, 372, 383  
CLASS, 331, 372, 383  
CLOSE, 331, 372, 383  
CODE, 332, 372, 383  
CODE\_SET, 372, 383  
CODE\_SET, 332  
COL, 332, 372, 383  
COLLATING, 332, 372, 383  
COLS, 332, 372, 383  
COLUMN, 332, 372, 383  
COLUMNS, 332, 372, 383  
COMMA, 332, 372, 383  
COMMA\_DELIM, 372, 383  
COMMA\_DELIM, 332  
COMMAND\_LINE, 372, 383  
COMMAND\_LINE, 332  
COMMIT, 333, 372, 383  
COMMON, 333, 372, 383  
COMP, 333, 372, 383  
COMP\_1, 372, 384  
COMP\_2, 372, 384  
COMP\_3, 372, 384  
COMP\_4, 372, 384  
COMP\_5, 372, 384  
COMP\_X, 372, 384  
COMP\_1, 333  
COMP\_2, 333  
COMP\_3, 333  
COMP\_4, 333  
COMP\_5, 333  
COMP\_X, 333  
COMPUTE, 333, 372, 384  
CONCATENATE\_FUNC, 372, 384  
CONCATENATE\_FUNC, 334  
CONFIGURATION, 334, 372, 384  
CONSTANT, 334, 372, 384  
CONTAINS, 334, 372, 384  
CONTENT, 334, 372, 384  
CONTINUE, 334, 372, 384  
CONTROL, 334, 372, 384  
CONTROL\_FOOTING, 372, 384  
CONTROL\_HEADING, 372, 384  
CONTROL\_FOOTING, 334  
CONTROL\_HEADING, 334  
CONTROLS, 334, 372, 384  
CONVERTING, 335, 372, 384

- COPY, 393, 394
- CORRESPONDING, 335, 373, 384
- COUNT, 335, 373, 384
- CRT, 335, 373, 384
- CURRENCY, 335, 373, 384
- CURRENT\_DATE\_FUNC, 373, 384
- CURRENT\_DATE\_FUNC, 335
- CURSOR, 335, 373, 384
- CYCLE, 335, 373, 384
- DATA, 335, 373, 384
- DATE, 335, 373, 384
- DAY, 336, 373, 384
- DAY\_OF\_WEEK, 373, 384
- DAY\_OF\_WEEK, 336
- DE, 336, 373, 384
- DEBUGGING, 336, 373, 384
- DECIMAL\_POINT, 373, 384
- DECIMAL\_POINT, 336
- DECLARATIVES, 336, 373, 384
- DEFAULT, 336, 373, 384
- DELETE, 336, 373, 384
- DELIMITED, 336, 373, 384
- DELIMITER, 336, 373, 384
- DEPENDING, 337, 373, 384
- DESCENDING, 337, 373, 385
- DETAIL, 337, 373, 385
- DISK, 337, 373, 385
- DISPLAY, 337, 373, 385
- DIVIDE, 337, 373, 385
- DIVISION, 337, 373, 385
- DOWN, 337, 373, 385
- DUPLICATES, 337, 373, 385
- DYNAMIC, 337, 373, 385
- EBCDIC, 338, 373, 385
- ELSE, 338, 373, 385
- END, 338, 373, 385
- END\_ACCEPT, 373, 385
- END\_ADD, 373, 385
- END\_CALL, 373, 385
- END\_COMPUTE, 373, 385
- END\_DELETE, 373, 385
- END\_DISPLAY, 373, 385
- END\_DIVIDE, 374, 385
- END\_EVALUATE, 374, 385
- END\_FUNCTION, 374, 385
- END\_IF, 374, 385
- END\_MULTIPLY, 374, 385
- END\_PERFORM, 374, 385
- END\_PROGRAM, 374, 385
- END\_READ, 374, 385
- END\_RETURN, 374, 385
- END\_REWRITE, 374, 385
- END\_SEARCH, 374, 385
- END\_START, 374, 385
- END\_STRING, 374, 385
- END\_SUBTRACT, 374, 385
- END\_UNSTRING, 374, 385
- END\_WRITE, 374, 385
- END\_ACCEPT, 338
- END\_ADD, 338
- END\_CALL, 338
- END\_COMPUTE, 338
- END\_DELETE, 338
- END\_DISPLAY, 338
- END\_DIVIDE, 338
- END\_EVALUATE, 339
- END\_FUNCTION, 339
- END\_IF, 339
- END\_MULTIPLY, 339
- END\_PERFORM, 339
- END\_PROGRAM, 339
- END\_READ, 339
- END\_RETURN, 339
- END\_REWRITE, 339
- END\_SEARCH, 339
- END\_START, 340
- END\_STRING, 340
- END\_SUBTRACT, 340
- END\_UNSTRING, 340
- END\_WRITE, 340
- ENTRY, 340, 374, 385
- ENVIRONMENT, 340, 374, 385
- ENVIRONMENT\_NAME, 374, 385
- ENVIRONMENT\_VALUE, 374, 385
- ENVIRONMENT\_NAME, 340
- ENVIRONMENT\_VALUE, 340
- EOL, 340, 374, 386
- EOP, 341, 374, 386
- EOS, 341, 374, 386
- EQEQ, 393, 394
- EQUAL, 341, 374, 386
- EQUALS, 341, 374, 386
- ERASE, 341, 374, 386
- ERROR, 341, 374, 386
- ESCAPE, 341, 374, 386
- EVALUATE, 341, 374, 386
- EVENT\_STATUS, 374, 386
- EVENT\_STATUS, 341
- EXCEPTION, 341, 374, 386
- EXCLUSIVE, 342, 374, 386

EXIT, 342, 374, 386  
EXTEND, 342, 374, 386  
EXTERNAL, 342, 374, 386  
FD, 342, 374, 386  
FILE\_CONTROL, 374, 386  
FILE\_ID, 374, 386  
FILE\_CONTROL, 342  
FILE\_ID, 342  
FILLER, 342, 375, 386  
FINAL, 342, 375, 386  
FIRST, 342, 375, 386  
FOOTING, 343, 375, 386  
FOR, 343, 375, 386  
FOREGROUND\_COLOR, 375, 386  
FOREGROUND\_COLOR, 343  
FOREVER, 343, 375, 386  
FREE, 343, 375, 386  
FROM, 343, 375, 386  
FULL, 343, 375, 386  
FUNCTION, 343, 375, 386  
FUNCTION\_ID, 375, 386  
FUNCTION\_NAME, 375, 386  
FUNCTION\_ID, 343  
FUNCTION\_NAME, 343  
GE, 344, 375, 386  
GENERATE, 344, 375, 386  
GIVING, 344, 375, 386  
GLOBAL, 344, 375, 386  
GO, 344, 375, 386  
GOBACK, 344, 375, 386  
GREATER, 344, 375, 386  
GROUP, 344, 375, 387  
HEADING, 344, 375, 387  
HIGH\_VALUE, 375, 387  
HIGH\_VALUE, 344  
HIGHLIGHT, 345, 375, 387  
I\_O, 376, 387  
I\_O\_CONTROL, 376, 387  
I\_O, 345  
I\_O\_CONTROL, 345  
IDENTIFICATION, 345, 375, 387  
IF, 345, 375, 387  
IGNORE, 345, 375, 387  
IGNORING, 345, 375, 387  
IN, 345, 375, 387, 393, 394  
INDEX, 345, 375, 387  
INDEXED, 345, 375, 387  
INDICATE, 346, 375, 387  
INITIALIZE, 346, 375, 387  
INITIALIZED, 346, 375, 387  
INITIATE, 346, 375, 387  
INPUT, 346, 375, 387  
INPUT\_OUTPUT, 375, 387  
INPUT\_OUTPUT, 346  
INSPECT, 346, 375, 387  
INTO, 346, 376, 387  
INTRINSIC, 346, 376, 387  
INVALID, 346, 376, 387  
INVALID\_KEY, 376, 387  
INVALID\_KEY, 347  
IS, 347, 376, 387  
JUSTIFIED, 347, 376, 387  
KEY, 347, 376, 387  
LABEL, 347, 376, 387  
LAST, 347, 376, 387  
LAST\_DETAIL, 376, 387  
LAST\_DETAIL, 347  
LE, 347, 376, 387  
LEADING, 347, 376, 387  
LEFT, 347, 376, 387  
LENGTH, 348, 376, 387  
LESS, 348, 376, 387  
LIMIT, 348, 376, 387  
LIMITS, 348, 376, 387  
LINAGE, 348, 376, 387  
LINAGE\_COUNTER, 376, 388  
LINAGE\_COUNTER, 348  
LINE, 348, 376, 388  
LINES, 348, 376, 388  
LINKAGE, 348, 376, 388  
LITERAL, 348, 376, 388  
LOCAL\_STORAGE, 376, 388  
LOCAL\_STORAGE, 349  
LOCALE, 349, 376, 388  
LOCALE\_DT\_FUNC, 376, 388  
LOCALE\_DT\_FUNC, 349  
LOCK, 349, 376, 388  
LOW\_VALUE, 376, 388  
LOW\_VALUE, 349  
LOWER\_CASE\_FUNC, 376, 388  
LOWER\_CASE\_FUNC, 349  
LOWLIGHT, 349, 376, 388  
MANUAL, 349, 376, 388  
MEMORY, 349, 376, 388  
MERGE, 349, 376, 388  
MINUS, 350, 376, 388  
MNEMONIC\_NAME, 376, 388  
MNEMONIC\_NAME, 350  
MODE, 350, 376, 388  
MOVE, 350, 377, 388

MULTIPLE, 350, 377, 388  
MULTIPLY, 350, 377, 388  
NATIONAL, 350, 377, 388  
NATIONAL\_EDITED, 377, 388  
NATIONAL\_EDITED, 350  
NATIVE, 350, 377, 388  
NE, 350, 377, 388  
NEGATIVE, 351, 377, 388  
NEXT, 351, 377, 388  
NEXT\_SENTENCE, 377, 388  
NEXT\_SENTENCE, 351  
NO, 351, 377, 388  
NO\_ADVANCING, 377, 388  
NO\_ADVANCING, 351  
NOT, 351, 377, 388  
NOT\_END, 377, 388  
NOT\_EOP, 377, 388  
NOT\_EXCEPTION, 377, 388  
NOT\_INVALID\_KEY, 377, 388  
NOT\_OVERFLOW, 377, 388  
NOT\_SIZE\_ERROR, 377, 388  
NOT\_END, 351  
NOT\_EOP, 351  
NOT\_EXCEPTION, 351  
NOT\_INVALID\_KEY, 351  
NOT\_OVERFLOW, 352  
NOT\_SIZE\_ERROR, 352  
NUMBER, 352, 377, 388  
NUMBERS, 352, 377, 389  
NUMERIC, 352, 377, 389  
NUMERIC\_EDITED, 377, 389  
NUMERIC\_EDITED, 352  
NUMVALC\_FUNC, 377, 389  
NUMVALC\_FUNC, 352  
OBJECT\_COMPUTER, 377, 389  
OBJECT\_COMPUTER, 352  
OCCURS, 352, 377, 389  
OF, 352, 377, 389, 393, 394  
OFF, 353, 377, 389, 393, 394  
OMITTED, 353, 377, 389  
ON, 353, 377, 389  
ONLY, 353, 377, 389  
OPEN, 353, 377, 389  
OPTIONAL, 353, 377, 389  
OR, 353, 377, 389  
ORDER, 353, 377, 389  
ORGANIZATION, 353, 377, 389  
OTHER, 353, 377, 389  
OUTPUT, 354, 377, 389  
OVERFLOW, 354, 378, 389  
OVERLINE, 354, 378, 389  
PACKED\_DECIMAL, 378, 389  
PACKED\_DECIMAL, 354  
PADDING, 354, 378, 389  
PAGE, 354, 378, 389  
PAGE\_FOOTING, 378, 389  
PAGE\_HEADING, 378, 389  
PAGE\_FOOTING, 354  
PAGE\_HEADING, 354  
PARAGRAPH, 354, 378, 389  
PERFORM, 354, 378, 389  
PICTURE, 355, 378, 389  
PLUS, 355, 378, 389  
POINTER, 355, 378, 389  
POSITION, 355, 378, 389  
POSITIVE, 355, 378, 389  
PRESENT, 355, 378, 389  
PREVIOUS, 355, 378, 389  
PRINTER, 355, 378, 389  
PRINTING, 355, 378, 389, 393, 394  
PROCEDURE, 355, 378, 389  
PROCEDURES, 356, 378, 389  
PROCEED, 356, 378, 390  
PROGRAM, 356, 378, 390  
PROGRAM\_ID, 378, 390  
PROGRAM\_NAME, 378, 390  
PROGRAM\_POINTER, 378, 390  
PROGRAM\_ID, 356  
PROGRAM\_NAME, 356  
PROGRAM\_POINTER, 356  
PROMPT, 356, 378, 390  
QUOTE, 356, 378, 390  
RANDOM, 356, 378, 390  
RD, 356, 378, 390  
READ, 357, 378, 390  
RECORD, 357, 378, 390  
RECORDING, 357, 378, 390  
RECORDS, 357, 378, 390  
RECURSIVE, 357, 378, 390  
REDEFINES, 357, 378, 390  
REEL, 357, 378, 390  
REFERENCE, 357, 378, 390  
RELATIVE, 357, 378, 390  
RELEASE, 357, 379, 390  
REMAINDER, 358, 379, 390  
REMOVAL, 358, 379, 390  
RENAMES, 358, 379, 390  
REPLACE, 393, 394  
REPLACING, 358, 379, 390, 393, 394  
REPORT, 358, 379, 390

REPORT\_FOOTING, 379, 390  
REPORT\_HEADING, 379, 390  
REPORT\_FOOTING, 358  
REPORT\_HEADING, 358  
REPORTING, 358, 379, 390  
REPORTS, 358, 379, 390  
REPOSITORY, 358, 379, 390  
REQUIRED, 359, 379, 390  
RESERVE, 359, 379, 390  
RETURN, 359, 379, 390  
RETURNING, 359, 379, 390  
REVERSE\_FUNC, 379, 390  
REVERSE\_VIDEO, 379, 390  
REVERSE\_FUNC, 359  
REVERSE\_VIDEO, 359  
REWIND, 359, 379, 390  
REWRITE, 359, 379, 390  
RIGHT, 359, 379, 390  
ROLLBACK, 359, 379, 391  
ROUNDED, 360, 379, 391  
RUN, 360, 379, 391  
SAME, 360, 379, 391  
SCREEN, 360, 379, 391  
SCREEN\_CONTROL, 379, 391  
SCREEN\_CONTROL, 360  
SCROLL, 360, 379, 391  
SD, 360, 379, 391  
SEARCH, 360, 379, 391  
SECTION, 360, 379, 391  
SECURE, 360, 379, 391  
SEGMENT\_LIMIT, 379, 391  
SEGMENT\_LIMIT, 361  
SELECT, 361, 379, 391  
SEMI\_COLON, 379, 391  
SEMI\_COLON, 361  
SENTENCE, 361, 379, 391  
SEPARATE, 361, 379, 391  
SEQUENCE, 361, 379, 391  
SEQUENTIAL, 361, 379, 391  
SET, 361, 380, 391  
SHARING, 361, 380, 391  
SIGN, 361, 380, 391  
SIGNED, 362, 380, 391  
SIGNED\_INT, 380, 391  
SIGNED\_LONG, 380, 391  
SIGNED\_SHORT, 380, 391  
SIGNED\_INT, 362  
SIGNED\_LONG, 362  
SIGNED\_SHORT, 362  
SIZE, 362, 380, 391  
SIZE\_ERROR, 380, 391  
SIZE\_ERROR, 362  
SORT, 362, 380, 391  
SORT\_MERGE, 380, 391  
SORT\_MERGE, 362  
SOURCE, 362, 380, 391  
SOURCE\_COMPUTER, 380, 391  
SOURCE\_COMPUTER, 362  
SPACE, 363, 380, 391  
SPECIAL\_NAMES, 380, 391  
SPECIAL\_NAMES, 363  
STANDARD, 363, 380, 391  
STANDARD\_1, 380, 391  
STANDARD\_2, 380, 391  
STANDARD\_1, 363  
STANDARD\_2, 363  
START, 363, 380, 391  
STATUS, 363, 380, 391  
STOP, 363, 380, 392  
STRING, 363, 380, 392  
SUBSTITUTE\_CASE\_FUNC, 380, 392  
SUBSTITUTE\_FUNC, 380, 392  
SUBSTITUTE\_CASE\_FUNC, 363  
SUBSTITUTE\_FUNC, 364  
SUBTRACT, 364, 380, 392  
SUM, 364, 380, 392  
SUPPRESS, 364, 380, 392–394  
SYMBOLIC, 364, 380, 392  
SYNCHRONIZED, 364, 380, 392  
TALLYING, 364, 380, 392  
TAPE, 364, 380, 392  
TERMINATE, 364, 380, 392  
TEST, 364, 380, 392  
THAN, 365, 380, 392  
THEN, 365, 380, 392  
THRU, 365, 380, 392  
TIME, 365, 380, 392  
TIMES, 365, 380, 392  
TO, 365, 381, 392  
TOK\_FALSE, 381, 392  
TOK\_FILE, 381, 392  
TOK\_INITIAL, 381, 392  
TOK\_NULL, 381, 392  
TOK\_TRUE, 381, 392  
TOK\_FALSE, 365  
TOK\_FILE, 365  
TOK\_INITIAL, 365  
TOK\_NULL, 365  
TOK\_TRUE, 366  
TOKEN, 393, 394

- TOKEN\_EOF, 370, 382, 393
- TOKEN\_EOF, 366
- TOP, 366, 381, 392
- TRAILING, 366, 381, 392
- TRANSFORM, 366, 381, 392
- TRIM\_FUNCTION, 381, 392
- TRIM\_FUNCTION, 366
- TYPE, 366, 381, 392
- UNARY\_SIGN, 382, 393
- UNARY\_SIGN, 366
- UNDERLINE, 366, 381, 392
- UNIT, 366, 381, 392
- UNLOCK, 367, 381, 392
- UNSIGNED, 367, 381, 392
- UNSIGNED\_INT, 381, 392
- UNSIGNED\_LONG, 381, 392
- UNSIGNED\_SHORT, 381, 392
- UNSIGNED\_INT, 367
- UNSIGNED\_LONG, 367
- UNSIGNED\_SHORT, 367
- UNSTRING, 367, 381, 392
- UNTIL, 367, 381, 392
- UP, 367, 381, 393
- UPDATE, 367, 381, 393
- UPON, 367, 381, 393
- UPON\_ARGUMENT\_NUMBER, 381, 393
- UPON\_COMMAND\_LINE, 381, 393
- UPON\_ENVIRONMENT\_NAME, 381, 393
- UPON\_ENVIRONMENT\_VALUE, 381, 393
- UPON\_ARGUMENT\_NUMBER, 368
- UPON\_COMMAND\_LINE, 368
- UPON\_ENVIRONMENT\_NAME, 368
- UPON\_ENVIRONMENT\_VALUE, 368
- UPPER\_CASE\_FUNC, 381, 393
- UPPER\_CASE\_FUNC, 368
- USAGE, 368, 381, 393
- USE, 368, 381, 393
- USING, 368, 381, 393
- VALUE, 368, 381, 393
- VARYING, 368, 381, 393
- WAIT, 369, 381, 393
- WHEN, 369, 381, 393
- WHEN\_COMPILED\_FUNC, 381, 393
- WHEN\_COMPILED\_FUNC, 369
- WITH, 369, 381, 393
- WORD, 369, 381, 393
- WORDS, 369, 382, 393
- WORKING\_STORAGE, 382, 393
- WORKING\_STORAGE, 369
- WRITE, 369, 382, 393
- yylval, 401
- YYSTYPE, 370
- yystype, 369
- YYSTYPE\_IS\_DECLARED, 369
- YYSTYPE\_IS\_TRIVIAL, 370
- yytokentype, 370
- YYYYDDD, 370, 382, 393
- YYYYMMDD, 370, 382, 393
- ZERO, 370, 382, 393
- PATCH\_LEVEL
  - config.h, 805
- PATHSEPC
  - call.c, 837
- PATHSEPS
  - call.c, 837
  - cobc.c, 144
- PENDING
  - parser.c, 252
- PERFORM
  - parser.c, 252, 284, 295
  - parser.h, 354, 378, 389
  - ppparse.c, 440, 452
  - ppparse.h, 475, 487
- PERMUTE
  - getopt.c, 811
- pic
  - cb\_field, 37
  - cob\_field\_attr, 97
- PIC\_ALPHABETIC
  - tree.c, 511
- PIC\_ALPHABETIC\_EDITED
  - tree.c, 511
- PIC\_ALPHANUMERIC
  - tree.c, 511
- PIC\_ALPHANUMERIC\_EDITED
  - tree.c, 511
- PIC\_EDITED
  - tree.c, 511
- PIC\_NATIONAL
  - tree.c, 511
- PIC\_NATIONAL\_EDITED
  - tree.c, 511
- PIC\_NUMERIC
  - tree.c, 512
- PIC\_NUMERIC\_EDITED
  - tree.c, 512
- PICTURE

- parser.c, [252](#), [284](#), [295](#)
- parser.h, [355](#), [378](#), [389](#)
- ppparse.c, [440](#), [452](#)
- ppparse.h, [475](#), [487](#)
- PICTURE\_STATE
  - scanner.c, [500](#)
- PLUS
  - parser.c, [252](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- POINTER
  - parser.c, [252](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- pointer
  - cobsort, [114](#)
- POSITION
  - parser.c, [253](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- POSITIVE
  - parser.c, [253](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- pp\_set\_replace\_list
  - cobc.h, [167](#)
- ppcopy
  - cobc.h, [167](#)
- pperror
  - ppparse.c, [424](#)
- ppin
  - cobc.h, [171](#)
- pplex
  - cobc.h, [167](#)
- pplex.c
  - BEGIN, [404](#)
  - COPY\_STATE, [404](#)
  - ECHO, [404](#)
  - EOB\_ACT\_CONTINUE\_SCAN, [404](#)
  - EOB\_ACT\_END\_OF\_FILE, [405](#)
  - EOB\_ACT\_LAST\_MATCH, [405](#)
  - file, [413](#)
  - FLEX\_SCANNER, [405](#)
  - INITIAL, [405](#)
  - len, [413](#)
  - PROCESS\_STATE, [405](#)
  - PSEUDO\_STATE, [405](#)
  - REJECT, [405](#)
  - size, [413](#)
  - unput, [405](#)
  - YY\_AT\_BOL, [405](#)
  - yy\_bp, [413](#)
  - YY\_BREAK, [405](#)
  - YY\_BUF\_SIZE, [406](#)
  - YY\_BUFFER\_EOF\_PENDING, [406](#)
  - YY\_BUFFER\_NEW, [406](#)
  - YY\_BUFFER\_NORMAL, [406](#)
  - YY\_BUFFER\_STATE, [412](#)
  - YY\_CHAR, [412](#)
  - yy\_create\_buffer, [406](#)
  - YY\_CURRENT\_BUFFER, [406](#)
  - YY\_DECL, [406](#)
  - yy\_delete\_buffer, [406](#)
  - YY\_DO\_BEFORE\_ACTION, [406](#)
  - YY\_END\_OF\_BUFFER, [407](#)
  - YY\_END\_OF\_BUFFER\_CHAR, [407](#)
  - YY\_EXIT\_FAILURE, [407](#)
  - YY\_FATAL\_ERROR, [407](#)
  - yy\_flex\_debug, [407](#)
  - YY\_FLEX\_MAJOR\_VERSION, [407](#)
  - YY\_FLEX\_MINOR\_VERSION, [407](#)
  - YY\_FLUSH\_BUFFER, [407](#)
  - yy\_flush\_buffer, [407](#)
  - yy\_init\_buffer, [407](#)
  - YY\_INPUT, [407](#)
  - yy\_load\_buffer\_state, [408](#)
  - YY\_MORE\_ADJ, [408](#)
  - YY\_NEVER\_INTERACTIVE, [408](#)
  - yy\_new\_buffer, [408](#)
  - YY\_NEW\_FILE, [408](#)
  - YY\_NO\_POP\_STATE, [408](#)
  - YY\_NO\_PUSH\_STATE, [408](#)
  - YY\_NO\_TOP\_STATE, [408](#)
  - YY\_NULL, [408](#)
  - YY\_NUM\_RULES, [408](#)
  - YY\_PROTO, [409](#), [413](#)
  - YY\_READ\_BUF\_SIZE, [409](#)
  - YY\_RESTORE\_YY\_MORE\_OFFSET, [409](#)
  - YY\_RULE\_SETUP, [409](#)
  - YY\_SC\_TO\_UI, [409](#)
  - yy\_scan\_buffer, [409](#)
  - yy\_scan\_bytes, [409](#)
  - yy\_scan\_string, [409](#)
  - yy\_set\_bol, [409](#)
  - yy\_set\_interactive, [410](#)



- yy\_size\_t, 412
- YY\_SKIP\_YYWRAP, 410
- YY\_START, 410
- YY\_START\_STACK\_INCR, 410
- YY\_STATE\_EOF, 410
- yy\_state\_type, 413
- yy\_switch\_to\_buffer, 410
- YY\_USER\_ACTION, 410
- yyconst, 410
- yyin, 411, 413
- yylen, 411, 414
- yyless, 411
- yylex, 411
- yyload, 411
- yyout, 412, 414
- yyrestart, 412
- YYSTATE, 412
- yyterminate, 412
- yytext, 412, 414
- yytext\_ptr, 412
- yywrap, 412
- pplval
  - ppparse.h, 492
- ppopen
  - cobc.h, 167
- ppout
  - cobc.h, 171
- ppparse
  - cobc.h, 167
- ppparse.c
  - ACCEPT, 433, 444
  - ACCESS, 433, 444
  - ADD, 433, 444
  - ADDRESS, 433, 444
  - ADVANCING, 433, 444
  - AFTER, 433, 444
  - ALL, 433, 444
  - ALLOCATE, 433, 444
  - ALPHABET, 433, 444
  - ALPHABETIC, 433, 445
  - ALPHABETIC\_LOWER, 433, 445
  - ALPHABETIC\_UPPER, 433, 445
  - ALPHANUMERIC, 433, 445
  - ALPHANUMERIC\_EDITED, 433, 445
  - ALSO, 433, 445
  - ALTER, 433, 445
  - ALTERNATE, 433, 445
  - AND, 433, 445
  - ANY, 433, 445
  - ARE, 433, 445
  - AREA, 433, 445
  - ARGUMENT\_NUMBER, 433, 445
  - ARGUMENT\_VALUE, 433, 445
  - AS, 433, 445
  - ASCENDING, 433, 445
  - ASSIGN, 433, 445
  - AT, 433, 445
  - AUTO, 434, 445
  - AUTOMATIC, 434, 445
  - BACKGROUND\_COLOR, 434, 445
  - BASED, 434, 445
  - BEFORE, 434, 445
  - BELL, 434, 445
  - BINARY, 434, 445
  - BINARY\_C\_LONG, 434, 445
  - BINARY\_CHAR, 434, 445
  - BINARY\_DOUBLE, 434, 445
  - BINARY\_LONG, 434, 445
  - BINARY\_SHORT, 434, 445
  - BLANK, 434, 445
  - BLANK\_LINE, 434, 445
  - BLANK\_SCREEN, 434, 445
  - BLINK, 434, 445
  - BLOCK, 434, 445
  - BOTTOM, 434, 445
  - BY, 424, 434, 445, 456
  - BYTE\_LENGTH, 434, 445
  - CALL, 434, 446
  - CANCEL, 434, 446
  - CH, 434, 446
  - CHAINING, 434, 446
  - CHARACTER, 434, 446
  - CHARACTERS, 434, 446
  - CLASS, 434, 446
  - CLOSE, 434, 446
  - CODE, 434, 446
  - CODE\_SET, 434, 446
  - COL, 434, 446
  - COLLATING, 434, 446
  - COLS, 434, 446
  - COLUMN, 434, 446
  - COLUMNS, 434, 446
  - COMMA, 434, 446
  - COMMA\_DELIM, 434, 446
  - COMMAND\_LINE, 434, 446
  - COMMIT, 435, 446
  - COMMON, 435, 446
  - COMP, 435, 446
  - COMP\_1, 435, 446
  - COMP\_2, 435, 446

COMP\_3, 435, 446  
COMP\_4, 435, 446  
COMP\_5, 435, 446  
COMP\_X, 435, 446  
COMPUTE, 435, 446  
CONCATENATE\_FUNC, 435, 446  
CONFIGURATION, 435, 446  
CONSTANT, 435, 446  
CONTAINS, 435, 446  
CONTENT, 435, 446  
CONTINUE, 435, 446  
CONTROL, 435, 446  
CONTROL\_FOOTING, 435, 446  
CONTROL\_HEADING, 435, 446  
CONTROLS, 435, 446  
CONVERTING, 435, 447  
COPY, 424, 456  
CORRESPONDING, 435, 447  
COUNT, 435, 447  
CRT, 435, 447  
CURRENCY, 435, 447  
CURRENT\_DATE\_FUNC, 435, 447  
CURSOR, 435, 447  
CYCLE, 435, 447  
DATA, 435, 447  
DATE, 435, 447  
DAY, 435, 447  
DAY\_OF\_WEEK, 435, 447  
DE, 435, 447  
DEBUGGING, 435, 447  
DECIMAL\_POINT, 435, 447  
DECLARATIVES, 435, 447  
DEFAULT, 435, 447  
DELETE, 435, 447  
DELIMITED, 436, 447  
DELIMITER, 436, 447  
DEPENDING, 436, 447  
DESCENDING, 436, 447  
DETAIL, 436, 447  
DISK, 436, 447  
DISPLAY, 436, 447  
DIVIDE, 436, 447  
DIVISION, 436, 447  
DOWN, 436, 447  
DUPLICATES, 436, 447  
DYNAMIC, 436, 447  
EBCDIC, 436, 447  
ELSE, 436, 447  
END, 436, 447  
END\_ACCEPT, 436, 447  
END\_ADD, 436, 447  
END\_CALL, 436, 447  
END\_COMPUTE, 436, 447  
END\_DELETE, 436, 447  
END\_DISPLAY, 436, 448  
END\_DIVIDE, 436, 448  
END\_EVALUATE, 436, 448  
END\_FUNCTION, 436, 448  
END\_IF, 436, 448  
END\_MULTIPLY, 436, 448  
END\_PERFORM, 436, 448  
END\_PROGRAM, 436, 448  
END\_READ, 436, 448  
END\_RETURN, 436, 448  
END\_REWRITE, 436, 448  
END\_SEARCH, 436, 448  
END\_START, 436, 448  
END\_STRING, 436, 448  
END\_SUBTRACT, 436, 448  
END\_UNSTRING, 436, 448  
END\_WRITE, 436, 448  
ENTRY, 436, 448  
ENVIRONMENT, 437, 448  
ENVIRONMENT\_NAME, 437, 448  
ENVIRONMENT\_VALUE, 437, 448  
EOL, 437, 448  
EOP, 437, 448  
EOS, 437, 448  
EQEQ, 424, 456  
EQUAL, 437, 448  
EQUALS, 437, 448  
ERASE, 437, 448  
ERROR, 437, 448  
ESCAPE, 437, 448  
EVALUATE, 437, 448  
EVENT\_STATUS, 437, 448  
EXCEPTION, 437, 448  
EXCLUSIVE, 437, 448  
EXIT, 437, 448  
EXTEND, 437, 448  
EXTERNAL, 437, 448  
FD, 437, 448  
FILE\_CONTROL, 437, 448  
FILE\_ID, 437, 449  
FILLER, 437, 449  
FINAL, 437, 449  
FIRST, 437, 449  
FOOTING, 437, 449  
FOR, 437, 449  
BACKGROUND\_COLOR, 437, 449

FOREVER, 437, 449  
FREE, 437, 449  
FROM, 437, 449  
FULL, 437, 449  
FUNCTION, 437, 449  
FUNCTION\_ID, 437, 449  
FUNCTION\_NAME, 437, 449  
GE, 437, 449  
GENERATE, 437, 449  
GIVING, 437, 449  
GLOBAL, 437, 449  
GO, 438, 449  
GOBACK, 438, 449  
GREATER, 438, 449  
GROUP, 438, 449  
HEADING, 438, 449  
HIGH\_VALUE, 438, 449  
HIGHLIGHT, 438, 449  
I\_O, 438, 450  
I\_O\_CONTROL, 438, 450  
IDENTIFICATION, 438, 449  
IF, 438, 449  
IGNORE, 438, 449  
IGNORING, 438, 449  
IN, 424, 438, 449, 456  
INDEX, 438, 449  
INDEXED, 438, 449  
INDICATE, 438, 449  
INITIALIZE, 438, 449  
INITIALIZED, 438, 449  
INITIATE, 438, 449  
INPUT, 438, 449  
INPUT\_OUTPUT, 438, 449  
INSPECT, 438, 450  
INTO, 438, 450  
INTRINSIC, 438, 450  
INVALID, 438, 450  
INVALID\_KEY, 438, 450  
IS, 438, 450  
JUSTIFIED, 438, 450  
KEY, 438, 450  
LABEL, 438, 450  
LAST, 438, 450  
LAST\_DETAIL, 438, 450  
LE, 438, 450  
LEADING, 438, 450  
LEFT, 438, 450  
LENGTH, 438, 450  
LESS, 438, 450  
LIMIT, 439, 450  
LIMITS, 439, 450  
LINAGE, 439, 450  
LINAGE\_COUNTER, 439, 450  
LINE, 439, 450  
LINES, 439, 450  
LINKAGE, 439, 450  
LITERAL, 439, 450  
LOCAL\_STORAGE, 439, 450  
LOCALE, 439, 450  
LOCALE\_DT\_FUNC, 439, 450  
LOCK, 439, 450  
LOW\_VALUE, 439, 450  
LOWER\_CASE\_FUNC, 439, 450  
LOWLIGHT, 439, 450  
MANUAL, 439, 450  
MEMORY, 439, 450  
MERGE, 439, 450  
MINUS, 439, 450  
MNEMONIC\_NAME, 439, 450  
MODE, 439, 451  
MOVE, 439, 451  
MULTIPLE, 439, 451  
MULTIPLY, 439, 451  
NATIONAL, 439, 451  
NATIONAL\_EDITED, 439, 451  
NATIVE, 439, 451  
NE, 439, 451  
NEGATIVE, 439, 451  
NEXT, 439, 451  
NEXT\_SENTENCE, 439, 451  
NO, 439, 451  
NO\_ADVANCING, 440, 451  
NOT, 439, 451  
NOT\_END, 439, 451  
NOT\_EOP, 439, 451  
NOT\_EXCEPTION, 439, 451  
NOT\_INVALID\_KEY, 439, 451  
NOT\_OVERFLOW, 439, 451  
NOT\_SIZE\_ERROR, 440, 451  
NUMBER, 440, 451  
NUMBERS, 440, 451  
NUMERIC, 440, 451  
NUMERIC\_EDITED, 440, 451  
NUMVALC\_FUNC, 440, 451  
OBJECT\_COMPUTER, 440, 451  
OCCURS, 440, 451  
OF, 424, 440, 451, 456  
OFF, 424, 440, 451, 456  
OMITTED, 440, 451  
ON, 440, 451

ONLY, 440, 451  
OPEN, 440, 451  
OPTIONAL, 440, 451  
OR, 440, 451  
ORDER, 440, 451  
ORGANIZATION, 440, 451  
OTHER, 440, 451  
OUTPUT, 440, 452  
OVERFLOW, 440, 452  
OVERLINE, 440, 452  
PACKED\_DECIMAL, 440, 452  
PADDING, 440, 452  
PAGE, 440, 452  
PAGE\_FOOTING, 440, 452  
PAGE\_HEADING, 440, 452  
PARAGRAPH, 440, 452  
PERFORM, 440, 452  
PICTURE, 440, 452  
PLUS, 440, 452  
POINTER, 440, 452  
POSITION, 440, 452  
POSITIVE, 440, 452  
pperror, 424  
PRESENT, 440, 452  
PREVIOUS, 440, 452  
PRINTER, 440, 452  
PRINTING, 425, 441, 452, 456  
PROCEDURE, 441, 452  
PROCEDURES, 441, 452  
PROCEED, 441, 452  
PROGRAM, 441, 452  
PROGRAM\_ID, 441, 452  
PROGRAM\_NAME, 441, 452  
PROGRAM\_POINTER, 441, 452  
PROMPT, 441, 452  
QUOTE, 441, 452  
RANDOM, 441, 452  
RD, 441, 452  
READ, 441, 452  
RECORD, 441, 452  
RECORDING, 441, 452  
RECORDS, 441, 452  
RECURSIVE, 441, 452  
REDEFINES, 441, 452  
REEL, 441, 452  
REFERENCE, 441, 452  
RELATIVE, 441, 453  
RELEASE, 441, 453  
REMAINDER, 441, 453  
REMOVAL, 441, 453  
RENAMES, 441, 453  
REPLACE, 425, 456  
REPLACING, 425, 441, 453, 456  
REPORT, 441, 453  
REPORT\_FOOTING, 441, 453  
REPORT\_HEADING, 441, 453  
REPORTING, 441, 453  
REPORTS, 441, 453  
REPOSITORY, 441, 453  
REQUIRED, 441, 453  
RESERVE, 441, 453  
RETURN, 441, 453  
RETURNING, 441, 453  
REVERSE\_FUNC, 441, 453  
REVERSE\_VIDEO, 441, 453  
REWIND, 442, 453  
REWRITE, 442, 453  
RIGHT, 442, 453  
ROLLBACK, 442, 453  
ROUNDED, 442, 453  
RUN, 442, 453  
SAME, 442, 453  
SCREEN, 442, 453  
SCREEN\_CONTROL, 442, 453  
SCROLL, 442, 453  
SD, 442, 453  
SEARCH, 442, 453  
SECTION, 442, 453  
SECURE, 442, 453  
SEGMENT\_LIMIT, 442, 453  
SELECT, 442, 453  
SEMI\_COLON, 442, 453  
SENTENCE, 442, 453  
SEPARATE, 442, 453  
SEQUENCE, 442, 453  
SEQUENTIAL, 442, 454  
SET, 442, 454  
SHARING, 442, 454  
SIGN, 442, 454  
SIGNED, 442, 454  
SIGNED\_INT, 442, 454  
SIGNED\_LONG, 442, 454  
SIGNED\_SHORT, 442, 454  
SIZE, 442, 454  
SIZE\_ERROR, 442, 454  
SORT, 442, 454  
SORT\_MERGE, 442, 454  
SOURCE, 442, 454  
SOURCE\_COMPUTER, 442, 454  
SPACE, 442, 454

- SPECIAL\_NAMES, 442, 454  
STANDARD, 442, 454  
STANDARD\_1, 442, 454  
STANDARD\_2, 443, 454  
START, 443, 454  
STATUS, 443, 454  
STOP, 443, 454  
STRING, 443, 454  
SUBSTITUTE\_CASE\_FUNC, 443, 454  
SUBSTITUTE\_FUNC, 443, 454  
SUBTRACT, 443, 454  
SUM, 443, 454  
SUPPRESS, 425, 443, 454, 456  
SYMBOLIC, 443, 454  
SYNCHRONIZED, 443, 454  
TALLYING, 443, 454  
TAPE, 443, 454  
TERMINATE, 443, 454  
TEST, 443, 454  
THAN, 443, 454  
THEN, 443, 454  
THRU, 443, 454  
TIME, 443, 454  
TIMES, 443, 455  
TO, 443, 455  
TOK\_FALSE, 443, 455  
TOK\_FILE, 443, 455  
TOK\_INITIAL, 443, 455  
TOK\_NULL, 443, 455  
TOK\_TRUE, 443, 455  
TOKEN, 425, 456  
TOKEN\_EOF, 433, 444, 456  
TOKEN\_EOF, 425  
TOP, 443, 455  
TRAILING, 443, 455  
TRANSFORM, 443, 455  
TRIM\_FUNCTION, 443, 455  
TYPE, 443, 455  
UNARY\_SIGN, 444, 456  
UNDERLINE, 443, 455  
UNIT, 443, 455  
UNLOCK, 443, 455  
UNSIGNED, 443, 455  
UNSIGNED\_INT, 443, 455  
UNSIGNED\_LONG, 443, 455  
UNSIGNED\_SHORT, 444, 455  
UNSTRING, 444, 455  
UNTIL, 444, 455  
UP, 444, 455  
UPDATE, 444, 455  
UPON, 444, 455  
UPON\_ARGUMENT\_NUMBER, 444, 455  
UPON\_COMMAND\_LINE, 444, 455  
UPON\_ENVIRONMENT\_NAME, 444, 455  
UPON\_ENVIRONMENT\_VALUE, 444, 455  
UPPER\_CASE\_FUNC, 444, 455  
USAGE, 444, 455  
USE, 444, 455  
USING, 444, 455  
VALUE, 444, 455  
VARYING, 444, 455  
WAIT, 444, 455  
WHEN, 444, 455  
WHEN\_COMPILED\_FUNC, 444, 455  
WITH, 444, 455  
WORD, 444, 456  
WORDS, 444, 456  
WORKING\_STORAGE, 444, 456  
WRITE, 444, 456  
YY\_REDUCE\_PRINT, 425  
YY\_STACK\_PRINT, 425  
YYABORT, 426  
YYACCEPT, 426  
YYBACKUP, 426  
YYBISON, 426  
yychar, 426, 457  
yyclearin, 426  
YYCOPY, 426  
YYDEBUG, 427  
yydebug, 427, 457  
YYDPRINTF, 427  
YYDSYMPRINT, 427  
YYDSYMPRINTF, 427  
YYEMPTY, 428  
YYEOF, 428  
YYERRCODE, 428  
yyerrok, 428  
YYERROR, 428  
yyerror, 428  
YYERROR\_VERBOSE, 428  
YYFAIL, 428  
YYFINAL, 429  
YYFPRINTF, 429  
YYINITDEPTH, 429  
YYLAST, 429  
YYLEX, 429  
yylex, 429

- YYLLOC\_DEFAULT, 429
- YYLSP\_NEEDED, 429
- yylval, 429, 457
- YYMAXDEPTH, 430
- YYMAXUTOK, 430
- yyners, 430, 457
- YYNNTS, 430
- YYNRULES, 430
- YYNSTATES, 430
- YYNTOKENS, 430
- YYPACT\_NINF, 430
- yyparse, 430, 457
- YYPOPSTACK, 430
- YYPURE, 431
- YYRECOVERING, 431
- yysigned\_char, 433
- YYSIZE\_T, 431
- YYSKELETON\_NAME, 431
- YYSTACK\_ALLOC, 431
- YYSTACK\_BYTES, 431
- YYSTACK\_FREE, 431
- YYSTACK\_GAP\_MAXIMUM, 431
- YYSTACK\_RELOCATE, 431
- YYSTYPE, 433
- yystype, 432
- YYSTYPE\_IS\_DECLARED, 432
- YYSTYPE\_IS\_TRIVIAL, 432
- YYTABLE\_NINF, 432
- YYTERROR, 432
- YYTOKENTYPE, 432
- yytokentype, 433
- YYTRANSLATE, 432
- YYUNDEFTOK, 432
- YYYYDDD, 444, 456
- YYYYMMDD, 444, 456
- ZERO, 444, 456
- ppparse.h
  - ACCEPT, 468, 479
  - ACCESS, 468, 479
  - ADD, 468, 479
  - ADDRESS, 468, 479
  - ADVANCING, 468, 479
  - AFTER, 468, 479
  - ALL, 468, 479
  - ALLOCATE, 468, 479
  - ALPHABET, 468, 479
  - ALPHABETIC, 468, 479
  - ALPHABETIC\_LOWER, 468, 479
  - ALPHABETIC\_UPPER, 468, 479
  - ALPHANUMERIC, 468, 479
  - ALPHANUMERIC\_EDITED, 468, 480
  - ALSO, 468, 480
  - ALTER, 468, 480
  - ALTERNATE, 468, 480
  - AND, 468, 480
  - ANY, 468, 480
  - ARE, 468, 480
  - AREA, 468, 480
  - ARGUMENT\_NUMBER, 468, 480
  - ARGUMENT\_VALUE, 468, 480
  - AS, 468, 480
  - ASCENDING, 468, 480
  - ASSIGN, 468, 480
  - AT, 468, 480
  - AUTO, 468, 480
  - AUTOMATIC, 468, 480
  - BACKGROUND\_COLOR, 468, 480
  - BASED, 468, 480
  - BEFORE, 469, 480
  - BELL, 469, 480
  - BINARY, 469, 480
  - BINARY\_C\_LONG, 469, 480
  - BINARY\_CHAR, 469, 480
  - BINARY\_DOUBLE, 469, 480
  - BINARY\_LONG, 469, 480
  - BINARY\_SHORT, 469, 480
  - BLANK, 469, 480
  - BLANK\_LINE, 469, 480
  - BLANK\_SCREEN, 469, 480
  - BLINK, 469, 480
  - BLOCK, 469, 480
  - BOTTOM, 469, 480
  - BY, 466, 469, 480, 491
  - BYTE\_LENGTH, 469, 480
  - CALL, 469, 480
  - CANCEL, 469, 480
  - CH, 469, 480
  - CHAINING, 469, 480
  - CHARACTER, 469, 481
  - CHARACTERS, 469, 481
  - CLASS, 469, 481
  - CLOSE, 469, 481
  - CODE, 469, 481
  - CODE\_SET, 469, 481
  - COL, 469, 481
  - COLLATING, 469, 481
  - COLS, 469, 481
  - COLUMN, 469, 481
  - COLUMNS, 469, 481
  - COMMA, 469, 481

COMMA\_DELIM, 469, 481  
COMMAND\_LINE, 469, 481  
COMMIT, 469, 481  
COMMON, 469, 481  
COMP, 469, 481  
COMP\_1, 470, 481  
COMP\_2, 470, 481  
COMP\_3, 470, 481  
COMP\_4, 470, 481  
COMP\_5, 470, 481  
COMP\_X, 470, 481  
COMPUTE, 469, 481  
CONCATENATE\_FUNC, 470, 481  
CONFIGURATION, 470, 481  
CONSTANT, 470, 481  
CONTAINS, 470, 481  
CONTENT, 470, 481  
CONTINUE, 470, 481  
CONTROL, 470, 481  
CONTROL\_FOOTING, 470, 481  
CONTROL\_HEADING, 470, 481  
CONTROLS, 470, 481  
CONVERTING, 470, 481  
COPY, 466, 491  
CORRESPONDING, 470, 481  
COUNT, 470, 481  
CRT, 470, 481  
CURRENCY, 470, 482  
CURRENT\_DATE\_FUNC, 470, 482  
CURSOR, 470, 482  
CYCLE, 470, 482  
DATA, 470, 482  
DATE, 470, 482  
DAY, 470, 482  
DAY\_OF\_WEEK, 470, 482  
DE, 470, 482  
DEBUGGING, 470, 482  
DECIMAL\_POINT, 470, 482  
DECLARATIVES, 470, 482  
DEFAULT, 470, 482  
DELETE, 470, 482  
DELIMITED, 470, 482  
DELIMITER, 470, 482  
DEPENDING, 470, 482  
DESCENDING, 470, 482  
DETAIL, 471, 482  
DISK, 471, 482  
DISPLAY, 471, 482  
DIVIDE, 471, 482  
DIVISION, 471, 482  
DOWN, 471, 482  
DUPLICATES, 471, 482  
DYNAMIC, 471, 482  
EBCDIC, 471, 482  
ELSE, 471, 482  
END, 471, 482  
END\_ACCEPT, 471, 482  
END\_ADD, 471, 482  
END\_CALL, 471, 482  
END\_COMPUTE, 471, 482  
END\_DELETE, 471, 482  
END\_DISPLAY, 471, 482  
END\_DIVIDE, 471, 482  
END\_EVALUATE, 471, 482  
END\_FUNCTION, 471, 482  
END\_IF, 471, 483  
END\_MULTIPLY, 471, 483  
END\_PERFORM, 471, 483  
END\_PROGRAM, 471, 483  
END\_READ, 471, 483  
END\_RETURN, 471, 483  
END\_REWRITE, 471, 483  
END\_SEARCH, 471, 483  
END\_START, 471, 483  
END\_STRING, 471, 483  
END\_SUBTRACT, 471, 483  
END\_UNSTRING, 471, 483  
END\_WRITE, 471, 483  
ENTRY, 471, 483  
ENVIRONMENT, 471, 483  
ENVIRONMENT\_NAME, 471, 483  
ENVIRONMENT\_VALUE, 471, 483  
EOL, 471, 483  
EOP, 472, 483  
EOS, 472, 483  
EQEQ, 466, 491  
EQUAL, 472, 483  
EQUALS, 472, 483  
ERASE, 472, 483  
ERROR, 472, 483  
ESCAPE, 472, 483  
EVALUATE, 472, 483  
EVENT\_STATUS, 472, 483  
EXCEPTION, 472, 483  
EXCLUSIVE, 472, 483  
EXIT, 472, 483  
EXTEND, 472, 483  
EXTERNAL, 472, 483  
FD, 472, 483  
FILE\_CONTROL, 472, 483

FILE\_ID, 472, 483  
FILLER, 472, 483  
FINAL, 472, 483  
FIRST, 472, 483  
FOOTING, 472, 484  
FOR, 472, 484  
FOREGROUND\_COLOR, 472, 484  
FOREVER, 472, 484  
FREE, 472, 484  
FROM, 472, 484  
FULL, 472, 484  
FUNCTION, 472, 484  
FUNCTION\_ID, 472, 484  
FUNCTION\_NAME, 472, 484  
GE, 472, 484  
GENERATE, 472, 484  
GIVING, 472, 484  
GLOBAL, 472, 484  
GO, 472, 484  
GOBACK, 472, 484  
GREATER, 472, 484  
GROUP, 472, 484  
HEADING, 473, 484  
HIGH\_VALUE, 473, 484  
HIGHLIGHT, 473, 484  
I\_O, 473, 485  
I\_O\_CONTROL, 473, 485  
IDENTIFICATION, 473, 484  
IF, 473, 484  
IGNORE, 473, 484  
IGNORING, 473, 484  
IN, 466, 473, 484, 491  
INDEX, 473, 484  
INDEXED, 473, 484  
INDICATE, 473, 484  
INITIALIZE, 473, 484  
INITIALIZED, 473, 484  
INITIATE, 473, 484  
INPUT, 473, 484  
INPUT\_OUTPUT, 473, 484  
INSPECT, 473, 484  
INTO, 473, 484  
INTRINSIC, 473, 484  
INVALID, 473, 484  
INVALID\_KEY, 473, 485  
IS, 473, 485  
JUSTIFIED, 473, 485  
KEY, 473, 485  
LABEL, 473, 485  
LAST, 473, 485  
LAST\_DETAIL, 473, 485  
LE, 473, 485  
LEADING, 473, 485  
LEFT, 473, 485  
LENGTH, 473, 485  
LESS, 473, 485  
LIMIT, 473, 485  
LIMITS, 473, 485  
LINAGE, 473, 485  
LINAGE\_COUNTER, 473, 485  
LINE, 474, 485  
LINES, 474, 485  
LINKAGE, 474, 485  
LITERAL, 474, 485  
LOCAL\_STORAGE, 474, 485  
LOCALE, 474, 485  
LOCALE\_DT\_FUNC, 474, 485  
LOCK, 474, 485  
LOW\_VALUE, 474, 485  
LOWER\_CASE\_FUNC, 474, 485  
LOWLIGHT, 474, 485  
MANUAL, 474, 485  
MEMORY, 474, 485  
MERGE, 474, 485  
MINUS, 474, 485  
MNEMONIC\_NAME, 474, 485  
MODE, 474, 485  
MOVE, 474, 485  
MULTIPLE, 474, 485  
MULTIPLY, 474, 485  
NATIONAL, 474, 486  
NATIONAL\_EDITED, 474, 486  
NATIVE, 474, 486  
NE, 474, 486  
NEGATIVE, 474, 486  
NEXT, 474, 486  
NEXT\_SENTENCE, 474, 486  
NO, 474, 486  
NO\_ADVANCING, 474, 486  
NOT, 474, 486  
NOT\_END, 474, 486  
NOT\_EOP, 474, 486  
NOT\_EXCEPTION, 474, 486  
NOT\_INVALID\_KEY, 474, 486  
NOT\_OVERFLOW, 474, 486  
NOT\_SIZE\_ERROR, 474, 486  
NUMBER, 474, 486  
NUMBERS, 474, 486  
NUMERIC, 475, 486  
NUMERIC\_EDITED, 475, 486



NUMVALC\_FUNC, 475, 486  
OBJECT\_COMPUTER, 475, 486  
OCCURS, 475, 486  
OF, 466, 475, 486, 491  
OFF, 466, 475, 486, 491  
OMITTED, 475, 486  
ON, 475, 486  
ONLY, 475, 486  
OPEN, 475, 486  
OPTIONAL, 475, 486  
OR, 475, 486  
ORDER, 475, 486  
ORGANIZATION, 475, 486  
OTHER, 475, 486  
OUTPUT, 475, 486  
OVERFLOW, 475, 486  
OVERLINE, 475, 486  
PACKED\_DECIMAL, 475, 486  
PADDING, 475, 487  
PAGE, 475, 487  
PAGE\_FOOTING, 475, 487  
PAGE\_HEADING, 475, 487  
PARAGRAPH, 475, 487  
PERFORM, 475, 487  
PICTURE, 475, 487  
PLUS, 475, 487  
POINTER, 475, 487  
POSITION, 475, 487  
POSITIVE, 475, 487  
pplval, 492  
PRESENT, 475, 487  
PREVIOUS, 475, 487  
PRINTER, 475, 487  
PRINTING, 467, 475, 487, 491  
PROCEDURE, 475, 487  
PROCEDURES, 475, 487  
PROCEED, 475, 487  
PROGRAM, 476, 487  
PROGRAM\_ID, 476, 487  
PROGRAM\_NAME, 476, 487  
PROGRAM\_POINTER, 476, 487  
PROMPT, 476, 487  
QUOTE, 476, 487  
RANDOM, 476, 487  
RD, 476, 487  
READ, 476, 487  
RECORD, 476, 487  
RECORDING, 476, 487  
RECORDS, 476, 487  
RECURSIVE, 476, 487  
REDEFINES, 476, 487  
REEL, 476, 487  
REFERENCE, 476, 487  
RELATIVE, 476, 487  
RELEASE, 476, 487  
REMAINDER, 476, 487  
REMOVAL, 476, 487  
RENAMES, 476, 488  
REPLACE, 467, 491  
REPLACING, 467, 476, 488, 491  
REPORT, 476, 488  
REPORT\_FOOTING, 476, 488  
REPORT\_HEADING, 476, 488  
REPORTING, 476, 488  
REPORTS, 476, 488  
REPOSITORY, 476, 488  
REQUIRED, 476, 488  
RESERVE, 476, 488  
RETURN, 476, 488  
RETURNING, 476, 488  
REVERSE\_FUNC, 476, 488  
REVERSE\_VIDEO, 476, 488  
REWIND, 476, 488  
REWRITE, 476, 488  
RIGHT, 476, 488  
ROLLBACK, 476, 488  
ROUNDED, 477, 488  
RUN, 477, 488  
SAME, 477, 488  
SCREEN, 477, 488  
SCREEN\_CONTROL, 477, 488  
SCROLL, 477, 488  
SD, 477, 488  
SEARCH, 477, 488  
SECTION, 477, 488  
SECURE, 477, 488  
SEGMENT\_LIMIT, 477, 488  
SELECT, 477, 488  
SEMI\_COLON, 477, 488  
SENTENCE, 477, 488  
SEPARATE, 477, 488  
SEQUENCE, 477, 488  
SEQUENTIAL, 477, 488  
SET, 477, 488  
SHARING, 477, 488  
SIGN, 477, 488  
SIGNED, 477, 489  
SIGNED\_INT, 477, 489  
SIGNED\_LONG, 477, 489  
SIGNED\_SHORT, 477, 489

- SIZE, [477](#), [489](#)
- SIZE\_ERROR, [477](#), [489](#)
- SORT, [477](#), [489](#)
- SORT\_MERGE, [477](#), [489](#)
- SOURCE, [477](#), [489](#)
- SOURCE\_COMPUTER, [477](#), [489](#)
- SPACE, [477](#), [489](#)
- SPECIAL\_NAMES, [477](#), [489](#)
- STANDARD, [477](#), [489](#)
- STANDARD\_1, [477](#), [489](#)
- STANDARD\_2, [477](#), [489](#)
- START, [477](#), [489](#)
- STATUS, [477](#), [489](#)
- STOP, [477](#), [489](#)
- STRING, [478](#), [489](#)
- SUBSTITUTE\_CASE\_FUNC, [478](#), [489](#)
- SUBSTITUTE\_FUNC, [478](#), [489](#)
- SUBTRACT, [478](#), [489](#)
- SUM, [478](#), [489](#)
- SUPPRESS, [467](#), [478](#), [489](#), [491](#)
- SYMBOLIC, [478](#), [489](#)
- SYNCHRONIZED, [478](#), [489](#)
- TALLYING, [478](#), [489](#)
- TAPE, [478](#), [489](#)
- TERMINATE, [478](#), [489](#)
- TEST, [478](#), [489](#)
- THAN, [478](#), [489](#)
- THEN, [478](#), [489](#)
- THRU, [478](#), [489](#)
- TIME, [478](#), [489](#)
- TIMES, [478](#), [489](#)
- TO, [478](#), [489](#)
- TOK\_FALSE, [478](#), [489](#)
- TOK\_FILE, [478](#), [489](#)
- TOK\_INITIAL, [478](#), [490](#)
- TOK\_NULL, [478](#), [490](#)
- TOK\_TRUE, [478](#), [490](#)
- TOKEN, [467](#), [491](#)
- TOKEN\_EOF, [468](#), [479](#), [491](#)
- TOKEN\_EOF, [467](#)
- TOP, [478](#), [490](#)
- TRAILING, [478](#), [490](#)
- TRANSFORM, [478](#), [490](#)
- TRIM\_FUNCTION, [478](#), [490](#)
- TYPE, [478](#), [490](#)
- UNARY\_SIGN, [479](#), [491](#)
- UNDERLINE, [478](#), [490](#)
- UNIT, [478](#), [490](#)
- UNLOCK, [478](#), [490](#)
- UNSIGNED, [478](#), [490](#)
- UNSIGNED\_INT, [478](#), [490](#)
- UNSIGNED\_LONG, [478](#), [490](#)
- UNSIGNED\_SHORT, [478](#), [490](#)
- UNSTRING, [478](#), [490](#)
- UNTIL, [478](#), [490](#)
- UP, [478](#), [490](#)
- UPDATE, [479](#), [490](#)
- UPON, [479](#), [490](#)
- UPON\_ARGUMENT\_NUMBER, [479](#), [490](#)
- UPON\_COMMAND\_LINE, [479](#), [490](#)
- UPON\_ENVIRONMENT\_NAME, [479](#), [490](#)
- UPON\_ENVIRONMENT\_VALUE, [479](#), [490](#)
- UPPER\_CASE\_FUNC, [479](#), [490](#)
- USAGE, [479](#), [490](#)
- USE, [479](#), [490](#)
- USING, [479](#), [490](#)
- VALUE, [479](#), [490](#)
- VARYING, [479](#), [490](#)
- WAIT, [479](#), [490](#)
- WHEN, [479](#), [490](#)
- WHEN\_COMPILED\_FUNC, [479](#), [490](#)
- WITH, [479](#), [490](#)
- WORD, [479](#), [490](#)
- WORDS, [479](#), [490](#)
- WORKING\_STORAGE, [479](#), [490](#)
- WRITE, [479](#), [490](#)
- YYSTYPE, [468](#)
- yystype, [467](#)
- YYSTYPE\_IS\_DECLARED, [467](#)
- YYSTYPE\_IS\_TRIVIAL, [467](#)
- yypointer, [468](#)
- YYYYDDD, [479](#), [491](#)
- YYYYMMDD, [479](#), [491](#)
- ZERO, [479](#), [491](#)
- preload\_handle
  - struct\_handle, [132](#)
- preprocess
  - filename, [120](#)
- PRESENT
  - parser.c, [253](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PREVIOUS
  - parser.c, [253](#), [284](#), [295](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)

- ppparse.h, [475](#), [487](#)
- PRINTER
  - parser.c, [253](#), [284](#), [296](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [440](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PRINTING
  - parser.c, [253](#), [284](#), [296](#), [299](#), [300](#)
  - parser.h, [355](#), [378](#), [389](#), [393](#), [394](#)
  - ppparse.c, [425](#), [441](#), [452](#), [456](#)
  - ppparse.h, [467](#), [475](#), [487](#), [491](#)
- PROCEDURE
  - parser.c, [253](#), [284](#), [296](#)
  - parser.h, [355](#), [378](#), [389](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PROCEDURES
  - parser.c, [253](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [389](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PROCEED
  - parser.c, [253](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [475](#), [487](#)
- PROCESS\_STATE
  - pplex.c, [405](#)
- prog\_type
  - cb\_program, [77](#)
- PROGRAM
  - parser.c, [253](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- PROGRAM\_ID
  - parser.c, [284](#), [296](#)
  - parser.h, [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- PROGRAM\_NAME
  - parser.c, [284](#), [296](#)
  - parser.h, [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- PROGRAM\_POINTER
  - parser.c, [284](#), [296](#)
  - parser.h, [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- PROGRAM\_ID
  - parser.c, [254](#)
  - parser.h, [356](#)
- program\_id
  - cb\_program, [77](#)
- PROGRAM\_NAME
  - parser.c, [254](#)
  - parser.h, [356](#)
- PROGRAM\_POINTER
  - parser.c, [254](#)
  - parser.h, [356](#)
- program\_spec\_list
  - cb\_program, [78](#)
- PROMPT
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- property\_spec\_list
  - cb\_program, [78](#)
- PSEUDO\_STATE
  - pplex.c, [405](#)
- purpose
  - cb\_list, [61](#)
- push\_expr
  - parser.c, [254](#)
- PUT\_SIGN\_ASCII
  - common.h, [886](#)
- queue
  - cobsort, [114](#)
- QUOTE
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- r
  - YYSTYPE, [136](#)
- r\_size
  - cobsort, [114](#)
- RANDOM
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- RD
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [356](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)

- ppparse.h, [476](#), [487](#)
- READ
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- read
  - cob\_fileio\_funcs, [104](#)
- read\_next
  - cob\_fileio\_funcs, [105](#)
- RECORD
  - parser.c, [254](#), [284](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- record
  - cb\_file, [44](#)
  - cob\_file, [101](#)
- record\_dependent
  - cb\_file, [44](#)
- record\_locked
  - indexed\_file, [124](#)
- record\_max
  - cb\_file, [44](#)
  - cob\_file, [101](#)
- record\_min
  - cb\_file, [44](#)
  - cob\_file, [101](#)
- record\_size
  - cob\_file, [102](#)
- RECORDING
  - parser.c, [255](#), [284](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- RECORDS
  - parser.c, [255](#), [284](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- RECURSIVE
  - parser.c, [255](#), [284](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- REDEFINES
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- redefines
  - cb\_field, [37](#)
- redefinition\_error
  - error.c, [196](#)
  - tree.h, [709](#)
- redefinition\_warning
  - error.c, [196](#)
  - tree.h, [709](#)
- REEL
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- ref
  - cb\_field::cb\_key, [55](#)
- REFERENCE
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [452](#)
  - ppparse.h, [476](#), [487](#)
- reference\_list
  - cb\_program, [78](#)
- refmod
  - cb\_intrinsic\_table, [54](#)
- REJECT
  - pplex.c, [405](#)
  - scanner.c, [500](#)
- RELATIVE
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [357](#), [378](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [487](#)
- RELEASE
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [357](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [487](#)
- REMAINDER
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [487](#)
- REMOVAL
  - parser.c, [255](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [487](#)
- rename\_thru
  - cb\_field, [37](#)
- RENAMES

- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- rep
- cb\_initialize, [50](#)
- REPLACE
- parser.c, [299](#), [300](#)
  - parser.h, [393](#), [394](#)
  - ppparse.c, [425](#), [456](#)
  - ppparse.h, [467](#), [491](#)
- REPLACING
- parser.c, [256](#), [285](#), [296](#), [299](#), [300](#)
  - parser.h, [358](#), [379](#), [390](#), [393](#), [394](#)
  - ppparse.c, [425](#), [441](#), [453](#), [456](#)
  - ppparse.h, [467](#), [476](#), [488](#), [491](#)
- REPORT
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REPORT\_FOOTING
- parser.c, [285](#), [296](#)
  - parser.h, [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REPORT\_HEADING
- parser.c, [285](#), [296](#)
  - parser.h, [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REPORT\_FOOTING
- parser.c, [256](#)
  - parser.h, [358](#)
- REPORT\_HEADING
- parser.c, [256](#)
  - parser.h, [358](#)
- REPORTING
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REPORTS
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REPOSITORY
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [358](#), [379](#), [390](#)
- ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REQUIRED
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- required\_argument
- getopt.h, [825](#)
- RESERVE
- parser.c, [256](#), [285](#), [296](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- reserved, [130](#)
- name, [131](#)
  - token, [131](#)
- reserved.c
- category, [496](#)
  - cb\_init\_reserved, [493](#)
  - cb\_list\_intrinsics, [493](#)
  - cb\_list\_mnemonics, [494](#)
  - cb\_list\_reserved, [494](#)
  - lookup\_intrinsic, [494](#)
  - lookup\_reserved\_word, [495](#)
  - lookup\_system\_name, [495](#)
  - name, [496](#)
  - node, [496](#)
  - NUM\_INTRINSICS, [493](#)
  - NUM\_RESERVED\_WORDS, [493](#)
  - token, [496](#)
- retrieval\_queue
- cobsort, [114](#)
- retrieving
- cobsort, [115](#)
- RETURN
- parser.c, [257](#), [285](#), [296](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- RETURN\_IN\_ORDER
- getopt.c, [811](#)
- RETURN\_STATUS
- fileio.c, [897](#)
- RETURNING
- parser.c, [257](#), [285](#), [296](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)

- ppparse.h, [476](#), [488](#)
- returning
  - cb\_call, [23](#)
  - cb\_program, [78](#)
- REVERSE\_FUNC
  - parser.c, [285](#), [296](#)
  - parser.h, [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REVERSE\_VIDEO
  - parser.c, [285](#), [297](#)
  - parser.h, [379](#), [390](#)
  - ppparse.c, [441](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REVERSE\_FUNC
  - parser.c, [257](#)
  - parser.h, [359](#)
- REVERSE\_VIDEO
  - parser.c, [257](#)
  - parser.h, [359](#)
- REWIND
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- REWRITE
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- rewrite
  - cob\_fileio\_funcs, [105](#)
- rewrite\_sec\_key
  - indexed\_file, [124](#)
- RIGHT
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [359](#), [379](#), [390](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- ROLLBACK
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [359](#), [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [476](#), [488](#)
- ROUNDED
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [360](#), [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [477](#), [488](#)
- RUN
  - parser.c, [257](#), [285](#), [297](#)
  - parser.h, [360](#), [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [477](#), [488](#)
- s
  - YYSTYPE, [137](#)
- SAME
  - parser.c, [258](#), [285](#), [297](#)
  - parser.h, [360](#), [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [477](#), [488](#)
- same\_clause
  - cb\_file, [44](#)
- scale
  - cb\_literal, [62](#)
  - cb\_picture, [69](#)
  - cob\_decimal, [93](#)
  - cob\_field\_attr, [97](#)
- scanner.c
  - BEGIN, [499](#)
  - DECIMAL\_IS\_COMMA, [499](#)
  - DECIMAL\_IS\_PERIOD, [499](#)
  - ECHO, [499](#)
  - EOB\_ACT\_CONTINUE\_SCAN, [499](#)
  - EOB\_ACT\_END\_OF\_FILE, [499](#)
  - EOB\_ACT\_LAST\_MATCH, [500](#)
  - file, [507](#)
  - FLEX\_SCANNER, [500](#)
  - FUNCTION\_STATE, [500](#)
  - INITIAL, [500](#)
  - len, [507](#)
  - PICTURE\_STATE, [500](#)
  - REJECT, [500](#)
  - SET\_LOCATION, [500](#)
  - size, [507](#)
  - unput, [500](#)
  - YY\_AT\_BOL, [500](#)
  - yy\_bp, [507](#)
  - YY\_BREAK, [500](#)
  - YY\_BUF\_SIZE, [501](#)
  - YY\_BUFFER\_EOF\_PENDING, [501](#)
  - YY\_BUFFER\_NEW, [501](#)
  - YY\_BUFFER\_NORMAL, [501](#)
  - YY\_BUFFER\_STATE, [506](#)
  - YY\_CHAR, [506](#)
  - YY\_CURRENT\_BUFFER, [501](#)
  - YY\_DECL, [501](#)
  - YY\_DO\_BEFORE\_ACTION, [501](#)
  - YY\_END\_OF\_BUFFER, [501](#)

- YY\_END\_OF\_BUFFER\_CHAR, [501](#)
- YY\_EXIT\_FAILURE, [502](#)
- YY\_FATAL\_ERROR, [502](#)
- YY\_FLEX\_MAJOR\_VERSION, [502](#)
- YY\_FLEX\_MINOR\_VERSION, [502](#)
- YY\_FLUSH\_BUFFER, [502](#)
- YY\_INPUT, [502](#)
- YY\_MORE\_ADJ, [502](#)
- YY\_NEVER\_INTERACTIVE, [502](#)
- yy\_new\_buffer, [503](#)
- YY\_NEW\_FILE, [503](#)
- YY\_NO\_POP\_STATE, [503](#)
- YY\_NO\_PUSH\_STATE, [503](#)
- YY\_NO\_TOP\_STATE, [503](#)
- YY\_NULL, [503](#)
- YY\_NUM\_RULES, [503](#)
- YY\_PROTO, [503](#), [507](#)
- YY\_READ\_BUF\_SIZE, [503](#)
- YY\_RESTORE\_YY\_MORE\_OFFSET, [503](#)
- YY\_RULE\_SETUP, [504](#)
- YY\_SC\_TO\_UI, [504](#)
- yy\_set\_bol, [504](#)
- yy\_set\_interactive, [504](#)
- yy\_size\_t, [506](#)
- YY\_SKIP\_YYWRAP, [504](#)
- YY\_START, [504](#)
- YY\_START\_STACK\_INCR, [505](#)
- YY\_STATE\_EOF, [505](#)
- yy\_state\_type, [506](#)
- YY\_USER\_ACTION, [505](#)
- yyconst, [505](#)
- yyin, [507](#)
- yylen, [507](#)
- yyless, [505](#)
- yymore, [506](#)
- yyout, [507](#)
- YYSTATE, [506](#)
- yyterminate, [506](#)
- yytext, [508](#)
- yytext\_ptr, [506](#)
- yywrap, [506](#)
- scr
  - cob\_inp\_struct, [107](#)
- SCREEN
  - parser.c, [258](#), [285](#), [297](#)
  - parser.h, [360](#), [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [477](#), [488](#)
- SCREEN\_CONTROL
  - parser.c, [285](#), [297](#)
  - parser.h, [379](#), [391](#)
  - ppparse.c, [442](#), [453](#)
  - ppparse.h, [477](#), [488](#)
- screen\_backg
  - cb\_field, [37](#)
- screen\_column
  - cb\_field, [37](#)
- SCREEN\_CONTROL
  - parser.c, [258](#)
  - parser.h, [360](#)
- screen\_flag
  - cb\_field, [37](#)
- screen\_foreg
  - cb\_field, [38](#)
- screen\_from
  - cb\_field, [38](#)
- screen\_line
  - cb\_field, [38](#)
- screen\_storage
  - cb\_program, [78](#)
- screen\_to
  - cb\_field, [38](#)
- screenio.c
  - \_XOPEN\_SOURCE\_EXTENDED, [1105](#)
  - cob\_field\_accept, [1105](#)
  - cob\_field\_display, [1108](#)
  - COB\_GEN\_SCREENIO, [1105](#)
  - COB\_INP\_SIZE, [1105](#)
  - cob\_screen\_accept, [1109](#)
  - cob\_screen\_display, [1110](#)
  - cob\_screen\_initialized, [1111](#)
  - cob\_screen\_line\_col, [1111](#)
  - cob\_screen\_mode, [1111](#)
  - cob\_screen\_set\_mode, [1111](#)
  - cob\_screen\_terminate, [1111](#)
- screenio.h
  - cob\_field\_accept, [1117](#)
  - cob\_field\_display, [1121](#)
  - cob\_screen, [1117](#)
  - cob\_screen\_accept, [1121](#)
  - COB\_SCREEN\_AUTO, [1114](#)
  - COB\_SCREEN\_BELL, [1114](#)
  - COB\_SCREEN\_BLACK, [1114](#)
  - COB\_SCREEN\_BLANK\_LINE, [1114](#)
  - COB\_SCREEN\_BLANK\_SCREEN, [1114](#)
  - COB\_SCREEN\_BLINK, [1114](#)
  - COB\_SCREEN\_BLUE, [1114](#)
  - COB\_SCREEN\_COLUMN\_MINUS, [1114](#)
  - COB\_SCREEN\_COLUMN\_PLUS, [1115](#)

- COB\_SCREEN\_CYAN, 1115
- cob\_screen\_display, 1122
- COB\_SCREEN\_ERASE\_EOL, 1115
- COB\_SCREEN\_ERASE\_EOS, 1115
- COB\_SCREEN\_FULL, 1115
- COB\_SCREEN\_GREEN, 1115
- COB\_SCREEN\_HIGHLIGHT, 1115
- COB\_SCREEN\_INPUT, 1115
- cob\_screen\_line\_col, 1123
- COB\_SCREEN\_LINE\_MINUS, 1115
- COB\_SCREEN\_LINE\_PLUS, 1115
- COB\_SCREEN\_LOWLIGHT, 1116
- COB\_SCREEN\_MAGENTA, 1116
- cob\_screen\_mode, 1123
- COB\_SCREEN\_OVERLINE, 1116
- COB\_SCREEN\_PROMPT, 1116
- COB\_SCREEN\_RED, 1116
- COB\_SCREEN\_REQUIRED, 1116
- COB\_SCREEN\_REVERSE, 1116
- COB\_SCREEN\_SCROLL\_DOWN, 1116
- COB\_SCREEN\_SECURE, 1116
- COB\_SCREEN\_TYPE\_ATTRIBUTE, 1116
- COB\_SCREEN\_TYPE\_FIELD, 1117
- COB\_SCREEN\_TYPE\_GROUP, 1117
- COB\_SCREEN\_TYPE\_VALUE, 1117
- COB\_SCREEN\_UNDERLINE, 1117
- COB\_SCREEN\_UPDATE, 1117
- COB\_SCREEN\_WHITE, 1117
- COB\_SCREEN\_YELLOW, 1117
- screenptr
  - cb\_funcall, 46
- SCROLL
  - parser.c, 258, 285, 297
  - parser.h, 360, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SD
  - parser.c, 258, 285, 297
  - parser.h, 360, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEARCH
  - parser.c, 258, 285, 297
  - parser.h, 360, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SECTION
  - parser.c, 258, 285, 297
  - parser.h, 360, 379, 391
  - ppparse.c, 442, 453
- ppparse.h, 477, 488
- section
  - cb\_label, 58
- SECURE
  - parser.c, 258, 285, 297
  - parser.h, 360, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEEK\_INIT
  - fileio.c, 897
- SEGMENT\_LIMIT
  - parser.c, 285, 297
  - parser.h, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEGMENT\_LIMIT
  - parser.c, 258
  - parser.h, 361
- SELECT
  - parser.c, 258, 285, 297
  - parser.h, 361, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- select\_name
  - cob\_file, 102
- SEMI\_COLON
  - parser.c, 285, 297
  - parser.h, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEMI\_COLON
  - parser.c, 259
  - parser.h, 361
- sending\_id
  - cobc.h, 171
  - typeck.c, 796
- SENTENCE
  - parser.c, 259, 286, 297
  - parser.h, 361, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEPARATE
  - parser.c, 259, 286, 297
  - parser.h, 361, 379, 391
  - ppparse.c, 442, 453
  - ppparse.h, 477, 488
- SEQUENCE
  - parser.c, 259, 286, 297
  - parser.h, 361, 379, 391
  - ppparse.c, 442, 453



- ppparse.h, 477, 488
- SEQUENTIAL
  - parser.c, 259, 286, 297
  - parser.h, 361, 379, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 488
- SET
  - parser.c, 259, 286, 297
  - parser.h, 361, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 488
- SET\_LOCATION
  - scanner.c, 500
- SHARING
  - parser.c, 259, 286, 297
  - parser.h, 361, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 488
- sharing
  - cb\_file, 44
- SIGN
  - parser.c, 259, 286, 297
  - parser.h, 361, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 488
- sign
  - cb\_literal, 63
- SIGNED
  - parser.c, 259, 286, 297
  - parser.h, 362, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SIGNED\_INT
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SIGNED\_LONG
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SIGNED\_SHORT
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SIGNED\_INT
  - parser.c, 259
  - parser.h, 362
- SIGNED\_LONG
  - parser.c, 260
  - parser.h, 362
- SIGNED\_SHORT
  - parser.c, 260
  - parser.h, 362
- sister
  - cb\_field, 38
- SIZE
  - parser.c, 260, 286, 297
  - parser.h, 362, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- size
  - cb\_field, 38
  - cb\_literal, 63
  - cb\_picture, 69
  - cb\_string, 87
  - cob\_alloc\_cache, 93
  - cob\_field, 96
  - cobsort, 115
  - pplex.c, 413
  - scanner.c, 507
- SIZE\_ERROR
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SIZE\_ERROR
  - parser.c, 260
  - parser.h, 362
- sizes
  - cb\_list, 61
- SORT
  - parser.c, 260, 286, 297
  - parser.h, 362, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SORT\_MERGE
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- sort\_collating
  - cb\_file, 102
- sort\_list, 131
  - next, 131
- SORT\_MERGE
  - parser.c, 260
  - parser.h, 362

- sort\_return
  - cobsort, 115
- SOURCE
  - parser.c, 260, 286, 297
  - parser.h, 362, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- source
  - filename, 120
- SOURCE\_COMPUTER
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SOURCE\_COMPUTER
  - parser.c, 260
  - parser.h, 362
- source\_file
  - cb\_tree\_common, 90
- source\_line
  - cb\_tree\_common, 90
- source\_name
  - cb\_program, 78
  - cobc.c, 154
  - cobc.h, 171
- SPACE
  - parser.c, 260, 286, 297
  - parser.h, 363, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- spare
  - cb\_label, 58
  - cb\_literal, 63
  - cb\_picture, 70
  - cb\_program, 78
- spare8
  - cob\_module, 110
- special
  - cb\_file, 44
  - cob\_file, 102
- SPECIAL\_NAMES
  - parser.c, 286, 297
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- SPECIAL\_NAMES
  - parser.c, 260
  - parser.h, 363
- STANDARD
  - parser.c, 261, 286, 297
  - parser.h, 363, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- STANDARD\_1
  - parser.c, 286, 298
  - parser.h, 380, 391
  - ppparse.c, 442, 454
  - ppparse.h, 477, 489
- STANDARD\_2
  - parser.c, 286, 298
  - parser.h, 380, 391
  - ppparse.c, 443, 454
  - ppparse.h, 477, 489
- STANDARD\_1
  - parser.c, 261
  - parser.h, 363
- STANDARD\_2
  - parser.c, 261
  - parser.h, 363
- START
  - parser.c, 261, 286, 298
  - parser.h, 363, 380, 391
  - ppparse.c, 443, 454
  - ppparse.h, 477, 489
- start
  - cob\_fileio\_funcs, 105
- START\_STACK\_SIZE
  - typeck.c, 726
- STATUS
  - parser.c, 261, 286, 298
  - parser.h, 363, 380, 391
  - ppparse.c, 443, 454
  - ppparse.h, 477, 489
- STDC\_HEADERS
  - config.h, 805
- step
  - cb\_perform\_varying, 68
- stmt1
  - cb\_call, 23
  - cb\_if, 48
- stmt2
  - cb\_call, 23
  - cb\_if, 48
- STOP
  - parser.c, 261, 286, 298
  - parser.h, 363, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 477, 489
- storage
  - cb\_field, 38

- storage\_id
  - cb\_field, 38
- str
  - cb\_picture, 70
- STRING
  - config.c, 186
  - parser.c, 261, 286, 298
  - parser.h, 363, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- strings.c
  - cob\_init\_strings, 1126
  - cob\_inspect\_after, 1126
  - cob\_inspect\_all, 1126
  - cob\_inspect\_before, 1126
  - cob\_inspect\_characters, 1127
  - cob\_inspect\_converting, 1127
  - cob\_inspect\_finish, 1128
  - cob\_inspect\_first, 1128
  - cob\_inspect\_init, 1128
  - cob\_inspect\_leading, 1129
  - cob\_inspect\_start, 1129
  - cob\_inspect\_trailing, 1129
  - cob\_string\_append, 1129
  - cob\_string\_delimited, 1130
  - cob\_string\_finish, 1130
  - cob\_string\_init, 1130
  - cob\_unstring\_delimited, 1131
  - cob\_unstring\_finish, 1131
  - cob\_unstring\_init, 1131
  - cob\_unstring\_into, 1132
  - cob\_unstring\_tallying, 1134
  - DLM\_DEFAULT\_NUM, 1125
  - INSPECT\_ALL, 1125
  - INSPECT\_FIRST, 1125
  - INSPECT\_LEADING, 1125
  - INSPECT\_TRAILING, 1125
- strings.h
  - cob\_inspect\_after, 1135
  - cob\_inspect\_all, 1136
  - cob\_inspect\_before, 1136
  - cob\_inspect\_characters, 1136
  - cob\_inspect\_converting, 1137
  - cob\_inspect\_finish, 1137
  - cob\_inspect\_first, 1137
  - cob\_inspect\_init, 1138
  - cob\_inspect\_leading, 1138
  - cob\_inspect\_start, 1138
  - cob\_inspect\_trailing, 1138
  - cob\_string\_append, 1139
  - cob\_string\_delimited, 1139
  - cob\_string\_finish, 1139
  - cob\_string\_init, 1140
  - cob\_unstring\_delimited, 1140
  - cob\_unstring\_finish, 1140
  - cob\_unstring\_init, 1141
  - cob\_unstring\_into, 1141
  - cob\_unstring\_tallying, 1143
- struct\_handle, 132
  - next, 132
  - preload\_handle, 132
- subs
  - cb\_reference, 80
- SUBSTITUTE\_CASE\_FUNC
  - parser.c, 286, 298
  - parser.h, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- SUBSTITUTE\_FUNC
  - parser.c, 286, 298
  - parser.h, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- SUBSTITUTE\_CASE\_FUNC
  - parser.c, 261
  - parser.h, 363
- SUBSTITUTE\_FUNC
  - parser.c, 261
  - parser.h, 364
- SUBTRACT
  - parser.c, 261, 286, 298
  - parser.h, 364, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- SUM
  - parser.c, 262, 286, 298
  - parser.h, 364, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- SUPPORT
  - config.c, 186
- SUPPRESS
  - parser.c, 262, 286, 298–300
  - parser.h, 364, 380, 392–394
  - ppparse.c, 425, 443, 454, 456
  - ppparse.h, 467, 478, 489, 491
- suppress\_warn
  - cobc.h, 171
  - typeck.c, 796
- SWAP\_FLAGS

- getopt.c, [811](#)
- SYMBOLIC
  - parser.c, [262](#), [286](#), [298](#)
  - parser.h, [364](#), [380](#), [392](#)
  - ppparse.c, [443](#), [454](#)
  - ppparse.h, [478](#), [489](#)
- symbolic\_list
  - cb\_program, [78](#)
- SYNCHRONIZED
  - parser.c, [262](#), [286](#), [298](#)
  - parser.h, [364](#), [380](#), [392](#)
  - ppparse.c, [443](#), [454](#)
  - ppparse.h, [478](#), [489](#)
- syst\_call
  - system\_table, [133](#)
- syst\_name
  - system\_table, [133](#)
- syst\_params
  - system\_table, [133](#)
- SYSTEM
  - common.h, [891](#)
- system\_table, [133](#)
  - syst\_call, [133](#)
  - syst\_name, [133](#)
  - syst\_params, [133](#)
- table
  - cb\_search, [84](#)
- tag
  - cb\_tree\_common, [90](#)
- TALLYING
  - parser.c, [262](#), [286](#), [298](#)
  - parser.h, [364](#), [380](#), [392](#)
  - ppparse.c, [443](#), [454](#)
  - ppparse.h, [478](#), [489](#)
- TAPE
  - parser.c, [262](#), [286](#), [298](#)
  - parser.h, [364](#), [380](#), [392](#)
  - ppparse.c, [443](#), [454](#)
  - ppparse.h, [478](#), [489](#)
- target
  - cb\_goto, [47](#)
- tarstamp.h, [1148](#)
- temp\_key
  - indexed\_file, [124](#)
- TERM\_ACCEPT
  - parser.c, [262](#)
- TERM\_ADD
  - parser.c, [262](#)
- TERM\_CALL
  - parser.c, [262](#)
- TERM\_COMPUTE
  - parser.c, [262](#)
- TERM\_DELETE
  - parser.c, [263](#)
- TERM\_DISPLAY
  - parser.c, [263](#)
- TERM\_DIVIDE
  - parser.c, [263](#)
- TERM\_EVALUATE
  - parser.c, [263](#)
- TERM\_IF
  - parser.c, [263](#)
- TERM\_MAX
  - parser.c, [263](#)
- TERM\_MULTIPLY
  - parser.c, [263](#)
- TERM\_NONE
  - parser.c, [263](#)
- TERM\_PERFORM
  - parser.c, [263](#)
- TERM\_READ
  - parser.c, [263](#)
- TERM\_RECEIVE
  - parser.c, [264](#)
- TERM\_RETURN
  - parser.c, [264](#)
- TERM\_REWRITE
  - parser.c, [264](#)
- TERM\_SEARCH
  - parser.c, [264](#)
- TERM\_START
  - parser.c, [264](#)
- TERM\_STRING
  - parser.c, [264](#)
- TERM\_SUBTRACT
  - parser.c, [264](#)
- TERM\_UNSTRING
  - parser.c, [264](#)
- TERM\_WRITE
  - parser.c, [264](#)
- TERMINATE
  - parser.c, [264](#), [286](#), [298](#)
  - parser.h, [364](#), [380](#), [392](#)
  - ppparse.c, [443](#), [454](#)
  - ppparse.h, [478](#), [489](#)
- termio.c
  - cob\_accept, [1144](#)
  - cob\_display, [1145](#)
  - cob\_init\_termio, [1145](#)

- termio.h
  - cob\_accept, 1147
  - cob\_display, 1147
- TEST
  - parser.c, 265, 286, 298
  - parser.h, 364, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- test
  - cb\_if, 48
  - cb\_perform, 66
- text
  - cb\_text\_list, 89
- textdomain
  - gettext.h, 831
- THAN
  - parser.c, 265, 286, 298
  - parser.h, 365, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- THEN
  - parser.c, 265, 287, 298
  - parser.h, 365, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- this\_x
  - cob\_inp\_struct, 107
- this\_y
  - cob\_inp\_struct, 107
- THRU
  - parser.c, 265, 287, 298
  - parser.h, 365, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- TIME
  - parser.c, 265, 287, 298
  - parser.h, 365, 380, 392
  - ppparse.c, 443, 454
  - ppparse.h, 478, 489
- TIMES
  - parser.c, 265, 287, 298
  - parser.h, 365, 380, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 489
- TO
  - parser.c, 265, 287, 298
  - parser.h, 365, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 489
- TOK\_FALSE
  - parser.c, 287, 298
  - parser.h, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 489
- TOK\_FILE
  - parser.c, 287, 298
  - parser.h, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 489
- TOK\_INITIAL
  - parser.c, 287, 298
  - parser.h, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 490
- TOK\_NULL
  - parser.c, 287, 298
  - parser.h, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 490
- TOK\_TRUE
  - parser.c, 287, 298
  - parser.h, 381, 392
  - ppparse.c, 443, 455
  - ppparse.h, 478, 490
- TOK\_FALSE
  - parser.c, 265
  - parser.h, 365
- TOK\_FILE
  - parser.c, 265
  - parser.h, 365
- TOK\_INITIAL
  - parser.c, 265
  - parser.h, 365
- TOK\_NULL
  - parser.c, 266
  - parser.h, 365
- TOK\_TRUE
  - parser.c, 266
  - parser.h, 366
- TOKEN
  - parser.c, 300
  - parser.h, 393, 394
  - ppparse.c, 425, 456
  - ppparse.h, 467, 491
  - typeck.c, 726
- token
  - cb\_system\_name, 88
  - expr\_node, 117
  - reserved, 131
  - reserved.c, 496

- TOKEN\_EOF
  - parser.c, [276](#), [288](#), [299](#), [300](#)
  - parser.h, [370](#), [382](#), [393](#)
  - ppparse.c, [433](#), [444](#), [456](#)
  - ppparse.h, [468](#), [479](#), [491](#)
- TOKEN\_EOF
  - parser.c, [266](#)
  - parser.h, [366](#)
  - ppparse.c, [425](#)
  - ppparse.h, [467](#)
- TOP
  - parser.c, [266](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- TRAILING
  - parser.c, [266](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- TRANSFORM
  - parser.c, [266](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- translate
  - filename, [120](#)
- tree.c
  - build\_file, [512](#)
  - build\_literal, [512](#)
  - cb\_any, [549](#)
  - cb\_build\_alphabet\_name, [512](#)
  - cb\_build\_alphanumeric\_literal, [513](#)
  - cb\_build\_any\_intrinsic, [513](#)
  - cb\_build\_assign, [513](#)
  - cb\_build\_binary\_list, [513](#)
  - cb\_build\_binary\_op, [514](#)
  - cb\_build\_call, [515](#)
  - cb\_build\_cast, [515](#)
  - cb\_build\_class\_name, [516](#)
  - cb\_build\_constant, [516](#)
  - cb\_build\_continue, [516](#)
  - cb\_build\_decimal, [516](#)
  - cb\_build\_field, [517](#)
  - cb\_build\_field\_reference, [517](#)
  - cb\_build\_filler, [517](#)
  - cb\_build\_funcall, [518](#)
  - cb\_build\_goto, [518](#)
  - cb\_build\_if, [518](#)
  - cb\_build\_implicit\_field, [519](#)
  - cb\_build\_initialize, [519](#)
  - cb\_build\_intrinsic, [519](#)
  - cb\_build\_label, [522](#)
  - cb\_build\_list, [522](#)
  - cb\_build\_locale\_name, [523](#)
  - cb\_build\_numeric\_literal, [523](#)
  - cb\_build\_perform, [523](#)
  - cb\_build\_perform\_varying, [523](#)
  - cb\_build\_picture, [524](#)
  - cb\_build\_program, [529](#)
  - cb\_build\_reference, [530](#)
  - cb\_build\_search, [530](#)
  - cb\_build\_statement, [530](#)
  - cb\_build\_string, [531](#)
  - cb\_build\_system\_name, [531](#)
  - cb\_concat\_literals, [531](#)
  - cb\_define, [532](#)
  - cb\_define\_system\_name, [533](#)
  - cb\_error\_node, [549](#)
  - cb\_false, [549](#)
  - cb\_field, [533](#)
  - cb\_field\_add, [533](#)
  - cb\_field\_founder, [533](#)
  - cb\_field\_size, [534](#)
  - cb\_field\_subordinate, [534](#)
  - cb\_field\_variable\_address, [535](#)
  - cb\_field\_variable\_size, [535](#)
  - cb\_fits\_int, [535](#)
  - cb\_fits\_long\_long, [536](#)
  - cb\_get\_int, [537](#)
  - cb\_get\_long\_long, [537](#)
  - cb\_high, [549](#)
  - cb\_i, [549](#)
  - cb\_init\_constants, [538](#)
  - cb\_int, [539](#)
  - cb\_int0, [549](#)
  - cb\_int1, [549](#)
  - cb\_int2, [549](#)
  - cb\_int3, [549](#)
  - cb\_int4, [549](#)
  - cb\_int5, [550](#)
  - cb\_intr\_e, [550](#)
  - cb\_intr\_pi, [550](#)
  - cb\_intr\_whencomp, [550](#)
  - cb\_list\_add, [539](#)
  - cb\_list\_append, [539](#)
  - cb\_list\_length, [540](#)
  - cb\_list\_map, [540](#)
  - cb\_list\_reverse, [540](#)
  - cb\_low, [550](#)

- cb\_name, 540
- cb\_norm\_high, 550
- cb\_norm\_low, 550
- cb\_null, 550
- cb\_one, 550
- cb\_quote, 550
- cb\_ref, 541
- cb\_space, 551
- cb\_standard\_error\_handler, 551
- cb\_tree\_category, 544
- cb\_tree\_class, 545
- cb\_tree\_type, 545
- cb\_true, 551
- cb\_zero, 551
- finalize\_file, 546
- gen\_screen\_ptr, 551
- PIC\_ALPHABETIC, 511
- PIC\_ALPHABETIC\_EDITED, 511
- PIC\_ALPHANUMERIC, 511
- PIC\_ALPHANUMERIC\_EDITED, 511
- PIC\_EDITED, 511
- PIC\_NATIONAL, 511
- PIC\_NATIONAL\_EDITED, 511
- PIC\_NUMERIC, 512
- PIC\_NUMERIC\_EDITED, 512
- validate\_file, 548
- tree.h
  - ambiguous\_error, 589
  - build\_file, 590
  - build\_literal, 590
  - CB\_ALPHABET\_CUSTOM, 577
  - CB\_ALPHABET\_EBCDIC, 577
  - CB\_ALPHABET\_NATIVE, 577
  - CB\_ALPHABET\_STANDARD\_1, 577
  - CB\_ALPHABET\_STANDARD\_2, 577
  - CB\_CALL\_CONVENTION\_NAME, 586
  - CB\_CAST\_ADDR\_OF\_ADDR, 577
  - CB\_CAST\_ADDRESS, 577
  - CB\_CAST\_INTEGER, 577
  - CB\_CAST\_LENGTH, 577
  - CB\_CAST\_PROGRAM\_POINTER, 577
  - CB\_CATEGORY\_ALPHABETIC, 578
  - CB\_CATEGORY\_ALPHANUMERIC, 578
  - CB\_CATEGORY\_ALPHANUMERIC\_EDITED, 578
  - CB\_CATEGORY\_BOOLEAN, 578
  - CB\_CATEGORY\_DATA\_POINTER, 578
  - CB\_CATEGORY\_INDEX, 578
  - CB\_CATEGORY\_NATIONAL, 578
  - CB\_CATEGORY\_NATIONAL\_EDITED, 578
  - CB\_CATEGORY\_NUMERIC, 578
  - CB\_CATEGORY\_NUMERIC\_EDITED, 578
  - CB\_CATEGORY\_OBJECT\_REFERENCE, 578
  - CB\_CATEGORY\_PROGRAM\_POINTER, 578
  - CB\_CATEGORY\_UNKNOWN, 578
  - CB\_CLASS\_ALPHABETIC, 578
  - CB\_CLASS\_ALPHANUMERIC, 578
  - CB\_CLASS\_BOOLEAN, 578
  - CB\_CLASS\_INDEX, 578
  - CB\_CLASS\_NATIONAL, 578
  - CB\_CLASS\_NUMERIC, 578
  - CB\_CLASS\_OBJECT, 578
  - CB\_CLASS\_POINTER, 579
  - CB\_CLASS\_UNKNOWN, 578
  - CB\_CODE\_NAME, 586
  - CB\_COMPUTER\_NAME, 586
  - CB\_DEVICE\_CONSOLE, 579
  - CB\_DEVICE\_NAME, 586
  - CB\_DEVICE\_SYSERR, 579
  - CB\_DEVICE\_SYSIN, 579
  - CB\_DEVICE\_SYSOUT, 579
  - CB\_ENTRY\_CONVENTION\_NAME, 586
  - CB\_EXTERNAL\_LOCALE\_NAME, 586
  - CB\_FEATURE\_C01, 579
  - CB\_FEATURE\_C02, 579
  - CB\_FEATURE\_C03, 579
  - CB\_FEATURE\_C04, 579
  - CB\_FEATURE\_C05, 579
  - CB\_FEATURE\_C06, 579
  - CB\_FEATURE\_C07, 579
  - CB\_FEATURE\_C08, 579
  - CB\_FEATURE\_C09, 580
  - CB\_FEATURE\_C10, 580
  - CB\_FEATURE\_C11, 580
  - CB\_FEATURE\_C12, 580
  - CB\_FEATURE\_FORMFEED, 579
  - CB\_FEATURE\_NAME, 586
  - CB\_INTR\_ABS, 580
  - CB\_INTR\_ACOS, 580
  - CB\_INTR\_ANNUITY, 580
  - CB\_INTR\_ASIN, 580
  - CB\_INTR\_ATAN, 580
  - CB\_INTR\_BOOLEAN\_OF\_INTEGER, 580
  - CB\_INTR\_BYTE\_LENGTH, 580

- CB\_INTR\_CHAR, 580
- CB\_INTR\_CHAR\_NATIONAL, 580
- CB\_INTR\_COMBINED\_DATETIME, 580
- CB\_INTR\_CONCATENATE, 580
- CB\_INTR\_COS, 580
- CB\_INTR\_CURRENT\_DATE, 580
- CB\_INTR\_DATE\_OF\_INTEGER, 580
- CB\_INTR\_DATE\_TO\_YYYYMMDD, 580
- CB\_INTR\_DAY\_OF\_INTEGER, 580
- CB\_INTR\_DAY\_TO\_YYYYDDD, 580
- CB\_INTR\_DISPLAY\_OF, 580
- CB\_INTR\_E, 581
- CB\_INTR\_EXCEPTION\_FILE, 581
- CB\_INTR\_EXCEPTION\_FILE\_N, 581
- CB\_INTR\_EXCEPTION\_LOCATION,  
581
- CB\_INTR\_EXCEPTION\_LOCATION\_-  
N, 581
- CB\_INTR\_EXCEPTION\_STATEMENT,  
581
- CB\_INTR\_EXCEPTION\_STATUS, 581
- CB\_INTR\_EXP, 581
- CB\_INTR\_EXP10, 581
- CB\_INTR\_FACTORIAL, 581
- CB\_INTR\_FRACTION\_PART, 581
- CB\_INTR\_HIGHEST\_ALGEBRAIC, 581
- CB\_INTR\_INTEGER, 581
- CB\_INTR\_INTEGER\_OF\_BOOLEAN,  
581
- CB\_INTR\_INTEGER\_OF\_DATE, 581
- CB\_INTR\_INTEGER\_OF\_DAY, 581
- CB\_INTR\_INTEGER\_PART, 581
- CB\_INTR\_LENGTH, 581
- CB\_INTR\_LOCALE\_COMPARE, 581
- CB\_INTR\_LOCALE\_DATE, 581
- CB\_INTR\_LOCALE\_TIME, 581
- CB\_INTR\_LOCALE\_TIME\_FROM\_SECS,  
581
- CB\_INTR\_LOG, 581
- CB\_INTR\_LOG10, 581
- CB\_INTR\_LOWER\_CASE, 581
- CB\_INTR\_LOWEST\_ALGEBRAIC, 581
- CB\_INTR\_MAX, 581
- CB\_INTR\_MEAN, 581
- CB\_INTR\_MEDIAN, 581
- CB\_INTR\_MIDRANGE, 581
- CB\_INTR\_MIN, 581
- CB\_INTR\_MOD, 581
- CB\_INTR\_NATIONAL\_OF, 581
- CB\_INTR\_NUMVAL, 581
- CB\_INTR\_NUMVAL\_C, 581
- CB\_INTR\_NUMVAL\_F, 581
- CB\_INTR\_ORD, 581
- CB\_INTR\_ORD\_MAX, 581
- CB\_INTR\_ORD\_MIN, 582
- CB\_INTR\_PI, 582
- CB\_INTR\_PRESENT\_VALUE, 582
- CB\_INTR\_RANDOM, 582
- CB\_INTR\_RANGE, 582
- CB\_INTR\_REM, 582
- CB\_INTR\_REVERSE, 582
- CB\_INTR\_SECONDS\_FROM\_FORMATTED\_-  
TIME, 582
- CB\_INTR\_SECONDS\_PAST\_MIDNIGHT,  
582
- CB\_INTR\_SIGN, 582
- CB\_INTR\_SIN, 582
- CB\_INTR\_SQRT, 582
- CB\_INTR\_STANDARD\_COMPARE, 582
- CB\_INTR\_STANDARD\_DEVIATION, 582
- CB\_INTR\_STORED\_CHAR\_LENGTH,  
582
- CB\_INTR\_SUBSTITUTE, 582
- CB\_INTR\_SUBSTITUTE\_CASE, 582
- CB\_INTR\_SUM, 582
- CB\_INTR\_TAN, 582
- CB\_INTR\_TEST\_DATE\_YYYYMMDD,  
582
- CB\_INTR\_TEST\_DAY\_YYYYDDD, 582
- CB\_INTR\_TEST\_NUMVAL, 582
- CB\_INTR\_TEST\_NUMVAL\_C, 582
- CB\_INTR\_TEST\_NUMVAL\_F, 582
- CB\_INTR\_TRIM, 582
- CB\_INTR\_UPPER\_CASE, 582
- CB\_INTR\_VARIANCE, 582
- CB\_INTR\_WHEN\_COMPILED, 582
- CB\_INTR\_YEAR\_TO\_YYYY, 582
- CB\_LIBRARY\_NAME, 586
- CB\_PERFORM\_EXIT, 584
- CB\_PERFORM\_FOREVER, 584
- CB\_PERFORM\_ONCE, 584
- CB\_PERFORM\_TIMES, 584
- CB\_PERFORM\_UNTIL, 584
- CB\_RECEIVING\_OPERAND, 584
- CB\_SENDING\_OPERAND, 584
- CB\_STORAGE\_COMMUNICATION, 585
- CB\_STORAGE\_CONSTANT, 585
- CB\_STORAGE\_FILE, 585
- CB\_STORAGE\_LINKAGE, 585
- CB\_STORAGE\_LOCAL, 585



- CB\_STORAGE\_REPORT, 585
- CB\_STORAGE\_SCREEN, 585
- CB\_STORAGE\_WORKING, 585
- CB\_SWITCH\_1, 585
- CB\_SWITCH\_2, 585
- CB\_SWITCH\_3, 585
- CB\_SWITCH\_4, 585
- CB\_SWITCH\_5, 585
- CB\_SWITCH\_6, 585
- CB\_SWITCH\_7, 585
- CB\_SWITCH\_8, 585
- CB\_SWITCH\_NAME, 586
- CB\_TAG\_ALPHABET\_NAME, 586
- CB\_TAG\_ASSIGN, 587
- CB\_TAG\_BINARY\_OP, 587
- CB\_TAG\_CALL, 587
- CB\_TAG\_CAST, 587
- CB\_TAG\_CLASS\_NAME, 586
- CB\_TAG\_CONST, 586
- CB\_TAG\_CONTINUE, 587
- CB\_TAG\_DECIMAL, 586
- CB\_TAG\_FIELD, 586
- CB\_TAG\_FILE, 586
- CB\_TAG\_FUNCALL, 587
- CB\_TAG\_GOTO, 587
- CB\_TAG\_IF, 587
- CB\_TAG\_INITIALIZE, 587
- CB\_TAG\_INTEGER, 586
- CB\_TAG\_INTRINSIC, 587
- CB\_TAG\_LABEL, 587
- CB\_TAG\_LIST, 587
- CB\_TAG\_LITERAL, 586
- CB\_TAG\_LOCALE\_NAME, 586
- CB\_TAG\_PERFORM, 587
- CB\_TAG\_PERFORM\_VARYING, 587
- CB\_TAG\_PICTURE, 587
- CB\_TAG\_REFERENCE, 586
- CB\_TAG\_SEARCH, 587
- CB\_TAG\_STATEMENT, 587
- CB\_TAG\_STRING, 586
- CB\_TAG\_SYSTEM\_NAME, 586
- CB\_TEXT\_NAME, 586
- CB\_USAGE\_BINARY, 588
- CB\_USAGE\_BIT, 588
- CB\_USAGE\_COMP\_5, 588
- CB\_USAGE\_COMP\_X, 588
- CB\_USAGE\_DISPLAY, 588
- CB\_USAGE\_DOUBLE, 588
- CB\_USAGE\_FLOAT, 588
- CB\_USAGE\_INDEX, 588
- CB\_USAGE\_LENGTH, 588
- CB\_USAGE\_NATIONAL, 588
- CB\_USAGE\_OBJECT, 588
- CB\_USAGE\_PACKED, 588
- CB\_USAGE\_POINTER, 588
- CB\_USAGE\_PROGRAM, 588
- CB\_USAGE\_PROGRAM\_POINTER, 588
- CB\_USAGE\_SIGNED\_CHAR, 588
- CB\_USAGE\_SIGNED\_INT, 588
- CB\_USAGE\_SIGNED\_LONG, 588
- CB\_USAGE\_SIGNED\_SHORT, 588
- CB\_USAGE\_UNSIGNED\_CHAR, 588
- CB\_USAGE\_UNSIGNED\_INT, 588
- CB\_USAGE\_UNSIGNED\_LONG, 588
- CB\_USAGE\_UNSIGNED\_SHORT, 588
- cb\_add\_78, 591
- CB\_AFTER, 564
- CB\_ALPHABET\_NAME, 564
- CB\_ALPHABET\_NAME\_P, 564
- cb\_alphabet\_name\_type, 577
- cb\_any, 719
- CB\_ASSIGN, 565
- CB\_ASSIGN\_P, 565
- CB\_BEFORE, 565
- CB\_BINARY\_OP, 565
- CB\_BINARY\_OP\_P, 565
- cb\_build\_add, 591
- cb\_build\_address, 591
- cb\_build\_alphabet\_name, 592
- cb\_build\_alphanumeric\_literal, 592
- cb\_build\_any\_intrinsic, 592
- cb\_build\_assign, 592
- cb\_build\_assignment\_name, 593
- cb\_build\_binary\_list, 594
- cb\_build\_binary\_op, 594
- cb\_build\_call, 595
- cb\_build\_cast, 596
- cb\_build\_cast\_addr\_of\_addr, 565
- cb\_build\_cast\_address, 565
- cb\_build\_cast\_integer, 565
- cb\_build\_cast\_length, 565
- cb\_build\_cast\_ppointer, 565
- cb\_build\_class\_name, 596
- cb\_build\_cond, 596
- cb\_build\_const\_length, 598
- cb\_build\_constant, 599
- cb\_build\_continue, 599
- cb\_build\_converting, 600
- cb\_build\_decimal, 600

- cb\_build\_display\_upon, [600](#)
- cb\_build\_display\_upon\_direct, [600](#)
- cb\_build\_expr, [601](#)
- cb\_build\_field, [602](#)
- cb\_build\_field\_reference, [603](#)
- cb\_build\_field\_tree, [603](#)
- cb\_build\_filler, [606](#)
- cb\_build\_funcall, [606](#)
- cb\_build\_funcall\_0, [566](#)
- cb\_build\_funcall\_1, [566](#)
- cb\_build\_funcall\_2, [566](#)
- cb\_build\_funcall\_3, [566](#)
- cb\_build\_funcall\_4, [566](#)
- cb\_build\_funcall\_5, [566](#)
- cb\_build\_funcall\_6, [566](#)
- cb\_build\_funcall\_7, [566](#)
- cb\_build\_goto, [606](#)
- cb\_build\_identifier, [606](#)
- cb\_build\_if, [610](#)
- cb\_build\_implicit\_field, [610](#)
- cb\_build\_index, [610](#)
- cb\_build\_initialize, [611](#)
- cb\_build\_inspect\_region, [611](#)
- cb\_build\_inspect\_region\_start, [611](#)
- cb\_build\_intrinsic, [611](#)
- cb\_build\_label, [614](#)
- cb\_build\_length, [614](#)
- cb\_build\_list, [615](#)
- cb\_build\_locale\_name, [615](#)
- cb\_build\_move, [616](#)
- cb\_build\_negation, [566](#)
- cb\_build\_numeric\_literal, [617](#)
- cb\_build\_pair, [567](#)
- cb\_build\_parenthesis, [567](#)
- cb\_build\_perform, [617](#)
- cb\_build\_perform\_exit, [618](#)
- cb\_build\_perform\_forever, [618](#)
- cb\_build\_perform\_once, [618](#)
- cb\_build\_perform\_times, [618](#)
- cb\_build\_perform\_until, [619](#)
- cb\_build\_perform\_varying, [619](#)
- cb\_build\_picture, [619](#)
- cb\_build\_ppointer, [625](#)
- cb\_build\_program, [625](#)
- cb\_build\_program\_id, [626](#)
- cb\_build\_reference, [626](#)
- cb\_build\_registers, [627](#)
- cb\_build\_replacing\_all, [628](#)
- cb\_build\_replacing\_characters, [628](#)
- cb\_build\_replacing\_first, [628](#)
- cb\_build\_replacing\_leading, [628](#)
- cb\_build\_replacing\_trailing, [629](#)
- cb\_build\_search, [629](#)
- cb\_build\_section\_name, [629](#)
- cb\_build\_statement, [630](#)
- cb\_build\_string, [630](#)
- cb\_build\_string0, [567](#)
- cb\_build\_sub, [630](#)
- cb\_build\_system\_name, [631](#)
- cb\_build\_tarrying\_all, [631](#)
- cb\_build\_tarrying\_characters, [631](#)
- cb\_build\_tarrying\_data, [632](#)
- cb\_build\_tarrying\_leading, [632](#)
- cb\_build\_tarrying\_trailing, [632](#)
- cb\_build\_tarrying\_value, [632](#)
- cb\_build\_unstring\_delimited, [633](#)
- cb\_build\_unstring\_into, [633](#)
- cb\_build\_write\_advancing\_lines, [633](#)
- cb\_build\_write\_advancing\_mnemonic, [634](#)
- cb\_build\_write\_advancing\_page, [634](#)
- CB\_CALL, [567](#)
- CB\_CALL\_BY\_CONTENT, [567](#)
- CB\_CALL\_BY\_REFERENCE, [567](#)
- CB\_CALL\_BY\_VALUE, [567](#)
- CB\_CALL\_P, [567](#)
- CB\_CAST, [567](#)
- CB\_CAST\_P, [567](#)
- cb\_cast\_type, [577](#)
- cb\_category, [577](#)
- CB\_CHAIN, [568](#)
- cb\_check\_numeric\_value, [634](#)
- cb\_class, [578](#)
- CB\_CLASS\_NAME, [568](#)
- CB\_CLASS\_NAME\_P, [568](#)
- cb\_clear\_real\_field, [635](#)
- cb\_concat\_literals, [635](#)
- cb\_cons, [568](#)
- CB\_CONST, [568](#)
- CB\_CONST\_P, [568](#)
- CB\_CONTINUE, [568](#)
- CB\_CONTINUE\_P, [568](#)
- CB\_DECIMAL, [568](#)
- CB\_DECIMAL\_P, [568](#)
- cb\_define, [636](#)
- cb\_define\_switch\_name, [636](#)
- cb\_define\_system\_name, [637](#)
- cb\_device\_name, [579](#)
- cb\_emit\_accept, [637](#)
- cb\_emit\_accept\_arg\_number, [639](#)

- cb\_emit\_accept\_arg\_value, 639
- cb\_emit\_accept\_command\_line, 639
- cb\_emit\_accept\_date, 639
- cb\_emit\_accept\_date\_yyyymmdd, 640
- cb\_emit\_accept\_day, 640
- cb\_emit\_accept\_day\_of\_week, 640
- cb\_emit\_accept\_day\_yyyyddd, 640
- cb\_emit\_accept\_environment, 641
- cb\_emit\_accept\_line\_or\_col, 641
- cb\_emit\_accept\_mnemonic, 641
- cb\_emit\_accept\_name, 641
- cb\_emit\_accept\_time, 642
- cb\_emit\_allocate, 642
- cb\_emit\_arg\_number, 643
- cb\_emit\_arithmetic, 643
- cb\_emit\_call, 644
- cb\_emit\_cancel, 646
- cb\_emit\_close, 646
- cb\_emit\_command\_line, 646
- cb\_emit\_commit, 647
- cb\_emit\_continue, 647
- cb\_emit\_corresponding, 647
- cb\_emit\_delete, 647
- cb\_emit\_display, 648
- cb\_emit\_divide, 650
- cb\_emit\_env\_name, 650
- cb\_emit\_env\_value, 650
- cb\_emit\_evaluate, 651
- cb\_emit\_exit, 651
- cb\_emit\_free, 651
- cb\_emit\_get\_environment, 652
- cb\_emit\_goto, 652
- cb\_emit\_if, 653
- cb\_emit\_initialize, 653
- cb\_emit\_inspect, 653
- cb\_emit\_move, 654
- cb\_emit\_move\_corresponding, 654
- cb\_emit\_open, 655
- cb\_emit\_perform, 655
- cb\_emit\_read, 655
- cb\_emit\_release, 656
- cb\_emit\_return, 657
- cb\_emit\_rewrite, 658
- cb\_emit\_rollback, 658
- cb\_emit\_search, 659
- cb\_emit\_search\_all, 659
- cb\_emit\_set\_false, 659
- cb\_emit\_set\_on\_off, 660
- cb\_emit\_set\_to, 660
- cb\_emit\_set\_true, 662
- cb\_emit\_set\_up\_down, 662
- cb\_emit\_setenv, 663
- cb\_emit\_sort\_finish, 663
- cb\_emit\_sort\_giving, 663
- cb\_emit\_sort\_init, 663
- cb\_emit\_sort\_input, 664
- cb\_emit\_sort\_output, 665
- cb\_emit\_sort\_using, 665
- cb\_emit\_start, 665
- cb\_emit\_stop\_run, 665
- cb\_emit\_string, 666
- cb\_emit\_unlock, 666
- cb\_emit\_unstring, 667
- cb\_emit\_write, 667
- cb\_encode\_program\_id, 668
- cb\_error\_node, 719
- cb\_error\_x, 669
- cb\_false, 719
- cb\_feature\_name, 579
- CB\_FIELD, 569
- cb\_field, 669
- cb\_field\_add, 669
- cb\_field\_founder, 670
- CB\_FIELD\_P, 569
- cb\_field\_size, 670
- cb\_field\_subordinate, 671
- cb\_field\_variable\_address, 671
- cb\_field\_variable\_size, 671
- CB\_FILE, 569
- CB\_FILE\_P, 569
- cb\_fits\_int, 672
- cb\_fits\_long\_long, 673
- CB\_FUNCALL, 569
- CB\_FUNCALL\_P, 569
- CB\_FUNCTION\_TYPE, 569
- cb\_get\_int, 673
- cb\_get\_level, 674
- cb\_get\_long\_long, 674
- CB\_GOTO, 569
- CB\_GOTO\_P, 569
- cb\_high, 719
- cb\_i, 719
- CB\_IF, 569
- CB\_IF\_P, 570
- CB\_INDEX\_P, 570
- cb\_init\_constants, 675
- cb\_init\_reserved, 676
- cb\_init\_tarrying, 676
- CB\_INITIALIZE, 570
- CB\_INITIALIZE\_P, 570

cb\_int, [676](#)  
 cb\_int0, [720](#)  
 cb\_int1, [720](#)  
 cb\_int2, [720](#)  
 cb\_int3, [720](#)  
 cb\_int4, [720](#)  
 cb\_int5, [720](#)  
 CB\_INTEGER, [570](#)  
 CB\_INTEGER\_P, [570](#)  
 cb\_intr\_e, [720](#)  
 cb\_intr\_enum, [580](#)  
 cb\_intr\_pi, [720](#)  
 cb\_intr\_whencomp, [720](#)  
 CB\_INTRINSIC, [570](#)  
 CB\_INTRINSIC\_P, [570](#)  
 CB\_LABEL, [570](#)  
 CB\_LABEL\_P, [571](#)  
 CB\_LIST, [571](#)  
 cb\_list\_add, [677](#)  
 cb\_list\_append, [677](#)  
 cb\_list\_init, [571](#)  
 cb\_list\_intrinsics, [677](#)  
 cb\_list\_length, [678](#)  
 cb\_list\_map, [678](#)  
 cb\_list\_mnemonics, [678](#)  
 CB\_LIST\_P, [571](#)  
 cb\_list\_reserved, [678](#)  
 cb\_list\_reverse, [679](#)  
 CB\_LITERAL, [571](#)  
 CB\_LITERAL\_P, [571](#)  
 CB\_LOCALE\_NAME, [571](#)  
 CB\_LOCALE\_NAME\_P, [571](#)  
 cb\_low, [720](#)  
 CB\_NAME, [571](#)  
 cb\_name, [679](#)  
 cb\_needs\_01, [721](#)  
 cb\_norm\_high, [721](#)  
 cb\_norm\_low, [721](#)  
 cb\_null, [721](#)  
 CB\_NUMERIC\_LITERAL\_P, [571](#)  
 cb\_one, [721](#)  
 cb\_operand\_type, [584](#)  
 CB\_PAIR\_P, [572](#)  
 CB\_PAIR\_X, [572](#)  
 CB\_PAIR\_Y, [572](#)  
 CB\_PERFORM, [572](#)  
 CB\_PERFORM\_P, [572](#)  
 cb\_perform\_type, [584](#)  
 CB\_PERFORM\_VARYING, [572](#)  
 CB\_PICTURE, [572](#)  
 CB\_PICTURE\_P, [572](#)  
 CB\_PREFIX\_ATTR, [572](#)  
 CB\_PREFIX\_BASE, [572](#)  
 CB\_PREFIX\_CONST, [573](#)  
 CB\_PREFIX\_DECIMAL, [573](#)  
 CB\_PREFIX\_FIELD, [573](#)  
 CB\_PREFIX\_FILE, [573](#)  
 CB\_PREFIX\_KEYS, [573](#)  
 CB\_PREFIX\_LABEL, [573](#)  
 CB\_PREFIX\_SEQUENCE, [573](#)  
 CB\_PROGRAM\_TYPE, [573](#)  
 CB\_PURPOSE, [573](#)  
 CB\_PURPOSE\_INT, [573](#)  
 cb\_quote, [721](#)  
 cb\_ref, [680](#)  
 CB\_REF\_OR\_FIELD\_P, [574](#)  
 CB\_REFERENCE, [574](#)  
 CB\_REFERENCE\_P, [574](#)  
 cb\_reset\_78, [682](#)  
 cb\_reset\_in\_procedure, [683](#)  
 cb\_resolve\_redefines, [683](#)  
 CB\_SEARCH, [574](#)  
 CB\_SEARCH\_P, [574](#)  
 cb\_set\_in\_procedure, [684](#)  
 CB\_SIZE\_1, [574](#)  
 CB\_SIZE\_2, [574](#)  
 CB\_SIZE\_4, [574](#)  
 CB\_SIZE\_8, [574](#)  
 CB\_SIZE\_AUTO, [574](#)  
 CB\_SIZE\_UNSIGNED, [575](#)  
 CB\_SIZES, [575](#)  
 CB\_SIZES\_INT, [575](#)  
 CB\_SIZES\_INT\_UNSIGNED, [575](#)  
 cb\_space, [721](#)  
 cb\_standard\_error\_handler, [721](#)  
 CB\_STATEMENT, [575](#)  
 CB\_STATEMENT\_P, [575](#)  
 cb\_storage, [584](#)  
 CB\_STRING, [575](#)  
 CB\_STRING\_P, [575](#)  
 cb\_switch\_name, [585](#)  
 CB\_SYSTEM\_NAME, [575](#)  
 cb\_system\_name\_category, [585](#)  
 CB\_SYSTEM\_NAME\_P, [575](#)  
 cb\_tag, [586](#)  
 CB\_TREE, [576](#)  
 cb\_tree, [577](#)  
 CB\_TREE\_CAST, [576](#)  
 CB\_TREE\_CATEGORY, [576](#)  
 cb\_tree\_category, [684](#)

- CB\_TREE\_CLASS, [576](#)
- cb\_tree\_class, [685](#)
- CB\_TREE\_TAG, [576](#)
- cb\_tree\_type, [685](#)
- cb\_true, [721](#)
- cb\_usage, [588](#)
- cb\_validate\_78\_item, [686](#)
- cb\_validate\_88\_item, [687](#)
- cb\_validate\_field, [687](#)
- cb\_validate\_program\_body, [688](#)
- cb\_validate\_program\_data, [688](#)
- cb\_validate\_program\_environment, [691](#)
- CB\_VALUE, [576](#)
- cb\_warning\_x, [694](#)
- CB\_WORD\_HASH\_SIZE, [576](#)
- cb\_zero, [721](#)
- check\_filler\_name, [695](#)
- check\_level\_78, [695](#)
- COB\_MAX\_SUBSCRIPTS, [576](#)
- cobc\_tree\_cast\_error, [695](#)
- codegen, [695](#)
- finalize\_file, [705](#)
- gen\_screen\_ptr, [722](#)
- group\_error, [707](#)
- level\_except\_error, [707](#)
- level\_redundant\_error, [708](#)
- level\_require\_error, [708](#)
- lookup\_intrinsic, [708](#)
- lookup\_reserved\_word, [708](#)
- lookup\_system\_name, [709](#)
- non\_const\_word, [722](#)
- redefinition\_error, [709](#)
- redefinition\_warning, [709](#)
- undefined\_error, [710](#)
- validate\_file, [710](#)
- validate\_move, [711](#)
- YYSTYPE, [576](#)
- TRIM\_FUNCTION
  - parser.c, [287](#), [298](#)
  - parser.h, [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- TRIM\_FUNCTION
  - parser.c, [266](#)
  - parser.h, [366](#)
- trstorage
  - filename, [120](#)
- TYPE
  - parser.c, [266](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
- ppparse.c, [443](#), [455](#)
- ppparse.h, [478](#), [490](#)
- type
  - \_\_cob\_screen, [15](#)
  - cb\_alphabet\_name, [18](#)
  - cb\_cast, [24](#)
  - cb\_perform, [66](#)
  - cb\_reference, [80](#)
  - cob\_field\_attr, [97](#)
  - config.c, [190](#)
- typeck.c
  - cb\_build\_add, [727](#)
  - cb\_build\_address, [727](#)
  - cb\_build\_assignment\_name, [727](#)
  - cb\_build\_cond, [729](#)
  - cb\_build\_const\_length, [731](#)
  - cb\_build\_converting, [731](#)
  - cb\_build\_display\_upon, [732](#)
  - cb\_build\_display\_upon\_direct, [732](#)
  - cb\_build\_expr, [733](#)
  - cb\_build\_identifier, [734](#)
  - cb\_build\_index, [737](#)
  - cb\_build\_inspect\_region, [737](#)
  - cb\_build\_inspect\_region\_start, [738](#)
  - cb\_build\_length, [738](#)
  - cb\_build\_move, [738](#)
  - cb\_build\_perform\_exit, [740](#)
  - cb\_build\_perform\_forever, [740](#)
  - cb\_build\_perform\_once, [740](#)
  - cb\_build\_perform\_times, [741](#)
  - cb\_build\_perform\_until, [741](#)
  - cb\_build\_ppointer, [741](#)
  - cb\_build\_program\_id, [742](#)
  - cb\_build\_registers, [742](#)
  - cb\_build\_replacing\_all, [744](#)
  - cb\_build\_replacing\_characters, [744](#)
  - cb\_build\_replacing\_first, [744](#)
  - cb\_build\_replacing\_leading, [744](#)
  - cb\_build\_replacing\_trailing, [744](#)
  - cb\_build\_section\_name, [744](#)
  - cb\_build\_sub, [745](#)
  - cb\_build\_tarrying\_all, [746](#)
  - cb\_build\_tarrying\_characters, [746](#)
  - cb\_build\_tarrying\_data, [746](#)
  - cb\_build\_tarrying\_leading, [746](#)
  - cb\_build\_tarrying\_trailing, [746](#)
  - cb\_build\_tarrying\_value, [747](#)
  - cb\_build\_unstring\_delimited, [747](#)
  - cb\_build\_unstring\_into, [747](#)
  - cb\_build\_write\_advancing\_lines, [747](#)

- cb\_build\_write\_advancing\_mnemonic, 748
- cb\_build\_write\_advancing\_page, 748
- cb\_check\_numeric\_value, 748
- cb\_define\_switch\_name, 749
- cb\_emit, 726
- cb\_emit\_accept, 749
- cb\_emit\_accept\_arg\_number, 751
- cb\_emit\_accept\_arg\_value, 751
- cb\_emit\_accept\_command\_line, 751
- cb\_emit\_accept\_date, 752
- cb\_emit\_accept\_date\_yyyymmdd, 752
- cb\_emit\_accept\_day, 752
- cb\_emit\_accept\_day\_of\_week, 752
- cb\_emit\_accept\_day\_yyyddd, 752
- cb\_emit\_accept\_environment, 753
- cb\_emit\_accept\_line\_or\_col, 753
- cb\_emit\_accept\_mnemonic, 753
- cb\_emit\_accept\_name, 753
- cb\_emit\_accept\_time, 754
- cb\_emit\_allocate, 754
- cb\_emit\_arg\_number, 755
- cb\_emit\_arithmetic, 755
- cb\_emit\_call, 756
- cb\_emit\_cancel, 758
- cb\_emit\_close, 758
- cb\_emit\_command\_line, 758
- cb\_emit\_commit, 759
- cb\_emit\_continue, 759
- cb\_emit\_corresponding, 759
- cb\_emit\_delete, 759
- cb\_emit\_display, 760
- cb\_emit\_divide, 762
- cb\_emit\_env\_name, 762
- cb\_emit\_env\_value, 762
- cb\_emit\_evaluate, 763
- cb\_emit\_exit, 763
- cb\_emit\_free, 763
- cb\_emit\_get\_environment, 764
- cb\_emit\_goto, 764
- cb\_emit\_if, 765
- cb\_emit\_initialize, 765
- cb\_emit\_inspect, 765
- cb\_emit\_list, 726
- cb\_emit\_move, 766
- cb\_emit\_move\_corresponding, 766
- cb\_emit\_open, 767
- cb\_emit\_perform, 767
- cb\_emit\_read, 767
- cb\_emit\_release, 768
- cb\_emit\_return, 769
- cb\_emit\_rewrite, 770
- cb\_emit\_rollback, 770
- cb\_emit\_search, 771
- cb\_emit\_search\_all, 771
- cb\_emit\_set\_false, 771
- cb\_emit\_set\_on\_off, 772
- cb\_emit\_set\_to, 772
- cb\_emit\_set\_true, 774
- cb\_emit\_set\_up\_down, 774
- cb\_emit\_setenv, 775
- cb\_emit\_sort\_finish, 775
- cb\_emit\_sort\_giving, 775
- cb\_emit\_sort\_init, 775
- cb\_emit\_sort\_input, 776
- cb\_emit\_sort\_output, 777
- cb\_emit\_sort\_using, 777
- cb\_emit\_start, 777
- cb\_emit\_stop\_run, 777
- cb\_emit\_string, 778
- cb\_emit\_unlock, 778
- cb\_emit\_unstring, 779
- cb\_emit\_write, 779
- cb\_encode\_program\_id, 780
- cb\_init\_tarrying, 781
- cb\_validate\_program\_body, 781
- cb\_validate\_program\_data, 782
- cb\_validate\_program\_environment, 784
- COB\_SYSTEM\_GEN, 726
- dpush, 726
- sending\_id, 796
- START\_STACK\_SIZE, 726
- suppress\_warn, 796
- TOKEN, 726
- validate\_move, 788
- VALUE, 726
- ucharptr
  - common.h, 887
- UNARY\_SIGN
  - parser.c, 288, 299
  - parser.h, 382, 393
  - ppparse.c, 444, 456
  - ppparse.h, 479, 491
- UNARY\_SIGN
  - parser.c, 266
  - parser.h, 366
- undefined\_error
  - error.c, 196
  - tree.h, 710

- UNDERLINE
  - parser.c, [266](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- unique
  - cobitem, [111](#)
  - cobsort, [115](#)
- UNIT
  - parser.c, [267](#), [287](#), [298](#)
  - parser.h, [366](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- unlikely
  - common.h, [887](#)
- UNLOCK
  - parser.c, [267](#), [287](#), [298](#)
  - parser.h, [367](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- unput
  - plex.c, [405](#)
  - scanner.c, [500](#)
- uns\_all
  - d1m\_struct, [116](#)
- uns\_d1m
  - d1m\_struct, [116](#)
- UNSIGNED
  - parser.c, [267](#), [287](#), [298](#)
  - parser.h, [367](#), [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- UNSIGNED\_INT
  - parser.c, [287](#), [298](#)
  - parser.h, [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- UNSIGNED\_LONG
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [392](#)
  - ppparse.c, [443](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- UNSIGNED\_SHORT
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [392](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- UNSIGNED\_INT
  - parser.c, [267](#)
  - parser.h, [367](#)
- UNSIGNED\_LONG
  - parser.c, [267](#)
  - parser.h, [367](#)
- UNSIGNED\_SHORT
  - parser.c, [267](#)
  - parser.h, [367](#)
- UNSTRING
  - parser.c, [267](#), [287](#), [299](#)
  - parser.h, [367](#), [381](#), [392](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- UNTIL
  - parser.c, [267](#), [287](#), [299](#)
  - parser.h, [367](#), [381](#), [392](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- until
  - cb\_perform\_varying, [68](#)
- UP
  - parser.c, [267](#), [287](#), [299](#)
  - parser.h, [367](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [478](#), [490](#)
- up\_index
  - cob\_inp\_struct, [107](#)
- UPDATE
  - parser.c, [267](#), [287](#), [299](#)
  - parser.h, [367](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPON
  - parser.c, [268](#), [287](#), [299](#)
  - parser.h, [367](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPON\_ARGUMENT\_NUMBER
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPON\_COMMAND\_LINE
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPON\_ENVIRONMENT\_NAME
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)

- UPON\_ENVIRONMENT\_VALUE
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPON\_ARGUMENT\_NUMBER
  - parser.c, [268](#)
  - parser.h, [368](#)
- UPON\_COMMAND\_LINE
  - parser.c, [268](#)
  - parser.h, [368](#)
- UPON\_ENVIRONMENT\_NAME
  - parser.c, [268](#)
  - parser.h, [368](#)
- UPON\_ENVIRONMENT\_VALUE
  - parser.c, [268](#)
  - parser.h, [368](#)
- UPPER\_CASE\_FUNC
  - parser.c, [287](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- UPPER\_CASE\_FUNC
  - parser.c, [268](#)
  - parser.h, [368](#)
- USAGE
  - parser.c, [268](#), [287](#), [299](#)
  - parser.h, [368](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- usage
  - cb\_field, [38](#)
- USE
  - parser.c, [268](#), [287](#), [299](#)
  - parser.h, [368](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- USE\_DB41
  - config.h, [805](#)
- USE\_LIBDL
  - config.h, [805](#)
- USING
  - parser.c, [268](#), [287](#), [299](#)
  - parser.h, [368](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- val
  - cb\_assign, [20](#)
  - cb\_cast, [24](#)
  - cb\_const, [27](#)
  - cb\_field::cb\_key, [55](#)
  - cb\_initialize, [50](#)
  - cb\_integer, [51](#)
  - config.c, [191](#)
  - option, [130](#)
- validate\_file
  - tree.c, [548](#)
  - tree.h, [710](#)
- validate\_move
  - tree.h, [711](#)
  - typeck.c, [788](#)
- VALUE
  - parser.c, [268](#), [287](#), [299](#)
  - parser.h, [368](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
  - typeck.c, [726](#)
- value
  - \_\_cob\_screen, [15](#)
  - cb\_list, [61](#)
  - cb\_reference, [81](#)
  - cob\_decimal, [93](#)
  - expr\_node, [117](#)
- values
  - cb\_field, [38](#)
- var
  - cb\_assign, [20](#)
  - cb\_initialize, [50](#)
  - cb\_search, [84](#)
  - config.c, [191](#)
- varcnt
  - cb\_funcall, [46](#)
- VARYING
  - parser.c, [269](#), [287](#), [299](#)
  - parser.h, [368](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- varying
  - cb\_perform, [66](#)
- VERSION
  - config.h, [805](#)
- w\_size
  - cobsort, [115](#)
- WAIT
  - parser.c, [269](#), [287](#), [299](#)
  - parser.h, [369](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)



- warningcount
  - cobc.c, [154](#)
  - cobc.h, [171](#)
- WHEN
  - parser.c, [269](#), [288](#), [299](#)
  - parser.h, [369](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- WHEN\_COMPILED\_FUNC
  - parser.c, [288](#), [299](#)
  - parser.h, [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- WHEN\_COMPILED\_FUNC
  - parser.c, [269](#)
  - parser.h, [369](#)
- whens
  - cb\_search, [84](#)
- WITH
  - parser.c, [269](#), [288](#), [299](#)
  - parser.h, [369](#), [381](#), [393](#)
  - ppparse.c, [444](#), [455](#)
  - ppparse.h, [479](#), [490](#)
- WITH\_DB
  - config.h, [805](#)
- WITH\_VARSEQ
  - config.h, [805](#)
- WORD
  - parser.c, [269](#), [288](#), [299](#)
  - parser.h, [369](#), [381](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [490](#)
- word
  - cb\_reference, [81](#)
- word\_table
  - cb\_program, [78](#)
- WORDS
  - parser.c, [269](#), [288](#), [299](#)
  - parser.h, [369](#), [382](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [490](#)
- WORKING\_STORAGE
  - parser.c, [288](#), [299](#)
  - parser.h, [382](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [490](#)
- WORKING\_STORAGE
  - parser.c, [269](#)
  - parser.h, [369](#)
- working\_storage
  - cb\_program, [78](#)
- WRITE
  - parser.c, [269](#), [288](#), [299](#)
  - parser.h, [369](#), [382](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [490](#)
- write
  - cob\_fileio\_funcs, [105](#)
- write\_cursor\_open
  - indexed\_file, [124](#)
- x
  - cb\_binary\_op, [22](#)
- y
  - cb\_binary\_op, [22](#)
- YY\_AT\_BOL
  - pplex.c, [405](#)
  - scanner.c, [500](#)
- yy\_at\_bol
  - yy\_buffer\_state, [134](#)
- yy\_bp
  - pplex.c, [413](#)
  - scanner.c, [507](#)
- YY\_BREAK
  - pplex.c, [405](#)
  - scanner.c, [500](#)
- yy\_buf\_pos
  - yy\_buffer\_state, [134](#)
- YY\_BUF\_SIZE
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- yy\_buf\_size
  - yy\_buffer\_state, [134](#)
- YY\_BUFFER\_EOF\_PENDING
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- YY\_BUFFER\_NEW
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- YY\_BUFFER\_NORMAL
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- YY\_BUFFER\_STATE
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- yy\_buffer\_state, [133](#)
  - yy\_at\_bol, [134](#)
  - yy\_buf\_pos, [134](#)
  - yy\_buf\_size, [134](#)

- yy\_buffer\_status, [134](#)
- yy\_ch\_buf, [134](#)
- yy\_fill\_buffer, [134](#)
- yy\_input\_file, [134](#)
- yy\_is\_interactive, [134](#)
- yy\_is\_our\_buffer, [135](#)
- yy\_n\_chars, [135](#)
- yy\_buffer\_status
  - yy\_buffer\_state, [134](#)
- yy\_ch\_buf
  - yy\_buffer\_state, [134](#)
- YY\_CHAR
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- yy\_create\_buffer
  - pplex.c, [406](#)
- YY\_CURRENT\_BUFFER
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- YY\_DECL
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- yy\_delete\_buffer
  - pplex.c, [406](#)
- YY\_DO\_BEFORE\_ACTION
  - pplex.c, [406](#)
  - scanner.c, [501](#)
- YY\_END\_OF\_BUFFER
  - pplex.c, [407](#)
  - scanner.c, [501](#)
- YY\_END\_OF\_BUFFER\_CHAR
  - pplex.c, [407](#)
  - scanner.c, [501](#)
- YY\_EXIT\_FAILURE
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- YY\_FATAL\_ERROR
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- yy\_fill\_buffer
  - yy\_buffer\_state, [134](#)
- yy\_flex\_debug
  - pplex.c, [407](#)
- YY\_FLEX\_MAJOR\_VERSION
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- YY\_FLEX\_MINOR\_VERSION
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- YY\_FLUSH\_BUFFER
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- yy\_flush\_buffer
  - pplex.c, [407](#)
- yy\_init\_buffer
  - pplex.c, [407](#)
- YY\_INPUT
  - pplex.c, [407](#)
  - scanner.c, [502](#)
- yy\_input\_file
  - yy\_buffer\_state, [134](#)
- yy\_is\_interactive
  - yy\_buffer\_state, [134](#)
- yy\_is\_our\_buffer
  - yy\_buffer\_state, [135](#)
- yy\_load\_buffer\_state
  - pplex.c, [408](#)
- YY\_MORE\_ADJ
  - pplex.c, [408](#)
  - scanner.c, [502](#)
- yy\_n\_chars
  - yy\_buffer\_state, [135](#)
- YY\_NEVER\_INTERACTIVE
  - pplex.c, [408](#)
  - scanner.c, [502](#)
- yy\_new\_buffer
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NEW\_FILE
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NO\_POP\_STATE
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NO\_PUSH\_STATE
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NO\_TOP\_STATE
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NULL
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_NUM\_RULES
  - pplex.c, [408](#)
  - scanner.c, [503](#)
- YY\_PROTO
  - pplex.c, [409](#), [413](#)
  - scanner.c, [503](#), [507](#)
- YY\_READ\_BUF\_SIZE

- pplex.c, [409](#)
- scanner.c, [503](#)
- YY\_REDUCE\_PRINT
  - parser.c, [269](#)
  - ppparse.c, [425](#)
- YY\_RESTORE\_YY\_MORE\_OFFSET
  - pplex.c, [409](#)
  - scanner.c, [503](#)
- YY\_RULE\_SETUP
  - pplex.c, [409](#)
  - scanner.c, [504](#)
- YY\_SC\_TO\_UI
  - pplex.c, [409](#)
  - scanner.c, [504](#)
- yy\_scan\_buffer
  - pplex.c, [409](#)
- yy\_scan\_bytes
  - pplex.c, [409](#)
- yy\_scan\_string
  - pplex.c, [409](#)
- yy\_set\_bol
  - pplex.c, [409](#)
  - scanner.c, [504](#)
- yy\_set\_interactive
  - pplex.c, [410](#)
  - scanner.c, [504](#)
- yy\_size\_t
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- YY\_SKIP\_YYWRAP
  - pplex.c, [410](#)
  - scanner.c, [504](#)
- YY\_STACK\_PRINT
  - parser.c, [270](#)
  - ppparse.c, [425](#)
- YY\_START
  - pplex.c, [410](#)
  - scanner.c, [504](#)
- YY\_START\_STACK\_INCR
  - pplex.c, [410](#)
  - scanner.c, [505](#)
- YY\_STATE\_EOF
  - pplex.c, [410](#)
  - scanner.c, [505](#)
- yy\_state\_type
  - pplex.c, [413](#)
  - scanner.c, [506](#)
- yy\_switch\_to\_buffer
  - pplex.c, [410](#)
- YY\_USER\_ACTION
  - pplex.c, [410](#)
  - scanner.c, [505](#)
- YYABORT
  - parser.c, [270](#)
  - ppparse.c, [426](#)
- YYACCEPT
  - parser.c, [270](#)
  - ppparse.c, [426](#)
- yyalloc, [135](#)
- yyss, [135](#)
- yyvs, [135](#)
- YYBACKUP
  - parser.c, [270](#)
  - ppparse.c, [426](#)
- YYBISON
  - parser.c, [270](#)
  - ppparse.c, [426](#)
- yychar
  - parser.c, [308](#)
  - ppparse.c, [426, 457](#)
- yyclearin
  - parser.c, [271](#)
  - ppparse.c, [426](#)
- yyconst
  - pplex.c, [410](#)
  - scanner.c, [505](#)
- YYCOPY
  - parser.c, [271](#)
  - ppparse.c, [426](#)
- YYDEBUG
  - parser.c, [271](#)
  - ppparse.c, [427](#)
- yydebug
  - parser.c, [308](#)
  - ppparse.c, [427, 457](#)
- YYDPRINTF
  - parser.c, [271](#)
  - ppparse.c, [427](#)
- YYDSYMPRINT
  - parser.c, [271](#)
  - ppparse.c, [427](#)
- YYDSYMPRINTF
  - parser.c, [271](#)
  - ppparse.c, [427](#)
- YYEMPTY
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- YYEOF
  - parser.c, [272](#)
  - ppparse.c, [428](#)

- YYERRCODE
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- yyerrok
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- YYERROR
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- yyerror
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- YYERROR\_VERBOSE
  - parser.c, [272](#)
  - ppparse.c, [428](#)
- YYFAIL
  - parser.c, [273](#)
  - ppparse.c, [428](#)
- YYFINAL
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- YYFPRINTF
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- yyin
  - cobc.h, [171](#)
  - pplex.c, [411](#), [413](#)
  - scanner.c, [507](#)
- YYINITDEPTH
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- YYLAST
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- yyleng
  - pplex.c, [411](#), [414](#)
  - scanner.c, [507](#)
- yyless
  - pplex.c, [411](#)
  - scanner.c, [505](#)
- YYLEX
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- yylex
  - cobc.h, [167](#)
  - pplex.c, [411](#)
  - ppparse.c, [429](#)
- YYLLOC\_DEFAULT
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- YYLSP\_NEEDED
  - parser.c, [273](#)
  - ppparse.c, [429](#)
- yylval
  - parser.c, [308](#)
  - parser.h, [401](#)
  - ppparse.c, [429](#), [457](#)
- YYMAXDEPTH
  - parser.c, [273](#)
  - ppparse.c, [430](#)
- YYMAXUTOK
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- yymore
  - pplex.c, [411](#)
  - scanner.c, [506](#)
- yynerrs
  - parser.c, [308](#)
  - ppparse.c, [430](#), [457](#)
- YYNNTS
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- YYNRULES
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- YYNSTATES
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- YYNTOKENS
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- yyout
  - cobc.h, [171](#)
  - pplex.c, [412](#), [414](#)
  - scanner.c, [507](#)
- YYPACT\_NINF
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- yyparse
  - cobc.h, [167](#)
  - parser.c, [307](#)
  - ppparse.c, [430](#), [457](#)
- YYPOPSTACK
  - parser.c, [274](#)
  - ppparse.c, [430](#)
- YYPURE
  - parser.c, [274](#)
  - ppparse.c, [431](#)
- YYRECOVERING
  - parser.c, [274](#)

- ppparse.c, [431](#)
- yyrestart
  - pplex.c, [412](#)
- yysigned\_char
  - parser.c, [276](#)
  - ppparse.c, [433](#)
- YYSIZE\_T
  - parser.c, [274](#)
  - ppparse.c, [431](#)
- YYSKELETON\_NAME
  - parser.c, [274](#)
  - ppparse.c, [431](#)
- yyss
  - yyalloc, [135](#)
- YYSTACK\_ALLOC
  - parser.c, [275](#)
  - ppparse.c, [431](#)
- YYSTACK\_BYTES
  - parser.c, [275](#)
  - ppparse.c, [431](#)
- YYSTACK\_FREE
  - parser.c, [275](#)
  - ppparse.c, [431](#)
- YYSTACK\_GAP\_MAXIMUM
  - parser.c, [275](#)
  - ppparse.c, [431](#)
- YYSTACK\_RELOCATE
  - parser.c, [275](#)
  - ppparse.c, [431](#)
- YYSTATE
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- YYSTYPE, [136](#)
  - l, [136](#)
  - parser.h, [370](#)
  - ppparse.c, [433](#)
  - ppparse.h, [468](#)
  - r, [136](#)
  - s, [137](#)
  - tree.h, [576](#)
- yystate
  - parser.h, [369](#)
  - ppparse.c, [432](#)
  - ppparse.h, [467](#)
- YYSTYPE\_IS\_DECLARED
  - parser.h, [369](#)
  - ppparse.c, [432](#)
  - ppparse.h, [467](#)
- YYSTYPE\_IS\_TRIVIAL
  - parser.h, [370](#)
- ppparse.c, [432](#)
- ppparse.h, [467](#)
- YYTABLE\_NINF
  - parser.c, [275](#)
  - ppparse.c, [432](#)
- yyterminate
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- YYTERROR
  - parser.c, [275](#)
  - ppparse.c, [432](#)
- yytext
  - pplex.c, [412](#), [414](#)
  - scanner.c, [508](#)
- yytext\_ptr
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- YYTOKENTYPE
  - parser.c, [275](#)
  - ppparse.c, [432](#)
- yytokentype
  - parser.c, [276](#)
  - parser.h, [370](#)
  - ppparse.c, [433](#)
  - ppparse.h, [468](#)
- YYTRANSLATE
  - parser.c, [276](#)
  - ppparse.c, [432](#)
- YYUNDEFTOK
  - parser.c, [276](#)
  - ppparse.c, [432](#)
- yyvs
  - yyalloc, [135](#)
- yywrap
  - pplex.c, [412](#)
  - scanner.c, [506](#)
- YYYYDDD
  - parser.c, [276](#), [288](#), [299](#)
  - parser.h, [370](#), [382](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [491](#)
- YYYYMMDD
  - parser.c, [276](#), [288](#), [299](#)
  - parser.h, [370](#), [382](#), [393](#)
  - ppparse.c, [444](#), [456](#)
  - ppparse.h, [479](#), [491](#)
- ZERO
  - parser.c, [276](#), [288](#), [299](#)
  - parser.h, [370](#), [382](#), [393](#)

ppparse.c, [444](#), [456](#)  
ppparse.h, [479](#), [491](#)