

m17n ライブラリ

1.8.3

構築: Doxygen 1.9.1

Copyright (C) 2001 Information-technology Promotion Agency (IPA)

Copyright (C) 2001-2011 National Institute of Advanced Industrial Science and Technology (AIST)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Section, with no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the appendix entitled "GNU Free Documentation License".

1 m17n ライブラリ	1
1.1 m17n ライブラリとは?	1
1.2 利用方法	1
1.3 外部ライブラリ／データ	2
1.4 連絡先:	2
1.5 謝辞	3
2 モジュール詳解	5
2.1 はじめに	5
2.1.1 詳解	6
2.1.2 マクロ定義詳解	7
2.1.2.1 M17NLIB_MAJOR_VERSION	7
2.1.2.2 M17NLIB_MINOR_VERSION	7
2.1.2.3 M17NLIB_PATCH_LEVEL	7
2.1.2.4 M17NLIB_VERSION_NAME	8
2.1.2.5 M17N_INIT	8
2.1.2.6 M17N_FINI	8
2.1.3 列挙型詳解	8
2.1.3.1 M17N_Status	8
2.1.4 関数詳解	9
2.1.4.1 m17n_status()	9
2.2 コア API	9
2.2.1 詳解	11
2.2.2 マクロ定義詳解	11
2.2.2.1 M17N_FUNC	11
2.2.3 型定義詳解	11
2.2.3.1 M17N_Func	11
2.3 管理下オブジェクト	12
2.3.1 詳解	12
2.3.2 関数詳解	12
2.3.2.1 m17n_object()	13
2.3.2.2 m17n_object_ref()	13
2.3.2.3 m17n_object_unref()	14
2.4 シンボル	14
2.4.1 詳解	15
2.4.2 関数詳解	15
2.4.2.1 msymbol()	15
2.4.2.2 msymbol_as_managing_key()	16
2.4.2.3 msymbol_is_managing_key()	16
2.4.2.4 msymbol_exist()	16

2.4.2.5	<code>msymbol_name()</code>	17
2.4.2.6	<code>msymbol_put()</code>	17
2.4.2.7	<code>msymbol_get()</code>	18
2.4.2.8	<code>msymbol_put_func()</code>	18
2.4.2.9	<code>msymbol_get_func()</code>	18
2.4.3	変数詳解	19
2.4.3.1	<code>Mnil</code>	19
2.4.3.2	<code>Mt</code>	19
2.4.3.3	<code>Mstring</code>	19
2.4.3.4	<code>Msymbol</code>	19
2.5	プロパティリスト	19
2.5.1	詳解	21
2.5.2	関数詳解	21
2.5.2.1	<code>mplist_deserialize()</code>	21
2.5.2.2	<code>mplist()</code>	21
2.5.2.3	<code>mplist_copy()</code>	22
2.5.2.4	<code>mplist_put()</code>	22
2.5.2.5	<code>mplist_get()</code>	23
2.5.2.6	<code>mplist_put_func()</code>	23
2.5.2.7	<code>mplist_get_func()</code>	23
2.5.2.8	<code>mplist_add()</code>	24
2.5.2.9	<code>mplist_push()</code>	24
2.5.2.10	<code>mplist_pop()</code>	24
2.5.2.11	<code>mplist_find_by_key()</code>	25
2.5.2.12	<code>mplist_find_by_value()</code>	25
2.5.2.13	<code>mplist_next()</code>	25
2.5.2.14	<code>mplist_set()</code>	25
2.5.2.15	<code>mplist_length()</code>	26
2.5.2.16	<code>mplist_key()</code>	26
2.5.2.17	<code>mplist_value()</code>	26
2.5.3	変数詳解	26
2.5.3.1	<code>Minteger</code>	26
2.5.3.2	<code>Mplist</code>	26
2.5.3.3	<code>Mtext</code>	27
2.6	文字	27
2.6.1	詳解	28
2.6.2	マクロ定義詳解	28
2.6.2.1	<code>MCHAR_MAX</code>	29
2.6.3	関数詳解	29
2.6.3.1	<code>mchar_define_property()</code>	29

2.6.3.2 mchar_get_prop()	29
2.6.3.3 mchar_put_prop()	30
2.6.3.4 mchar_get_prop_table()	30
2.6.4 変数詳解	30
2.6.4.1 Mscript	31
2.6.4.2 Mname	31
2.6.4.3 Mcategory	31
2.6.4.4 Mcombining_class	31
2.6.4.5 Mbidi_category	32
2.6.4.6 Msimple_case_folding	32
2.6.4.7 Mcomplicated_case_folding	32
2.6.4.8 Mcased	32
2.6.4.9 Msoft_dotted	33
2.6.4.10 Mcase_mapping	33
2.6.4.11 Mblock	33
2.7 文字テーブル	33
2.7.1 詳解	34
2.7.2 型定義詳解	34
2.7.2.1 MCharTable	35
2.7.3 関数詳解	35
2.7.3.1 mchartable()	35
2.7.3.2 mchartable_min_char()	35
2.7.3.3 mchartable_max_char()	35
2.7.3.4 mchartable_lookup()	36
2.7.3.5 mchartable_set()	36
2.7.3.6 mchartable_set_range()	37
2.7.3.7 mchartable_range()	37
2.7.3.8 mchartable_map()	38
2.7.4 変数詳解	38
2.7.4.1 Mchar_table	38
2.8 M-text	38
2.8.1 詳解	41
2.8.2 列挙型詳解	41
2.8.2.1 MTextFormat	41
2.8.2.2 MTextLineBreakOption	42
2.8.3 関数詳解	42
2.8.3.1 mtext_line_break()	42
2.8.3.2 mtext()	42
2.8.3.3 mtext_from_data()	43
2.8.3.4 mtext_data()	43

2.8.3.5 mtext_len()	43
2.8.3.6 mtext_ref_char()	44
2.8.3.7 mtext_set_char()	44
2.8.3.8 mtext_cat_char()	44
2.8.3.9 mtext_dup()	45
2.8.3.10 mtext_cat()	45
2.8.3.11 mtext_ncat()	45
2.8.3.12 mtext_cpy()	46
2.8.3.13 mtext_ncpy()	46
2.8.3.14 mtext_duplicate()	47
2.8.3.15 mtext_copy()	47
2.8.3.16 mtext_del()	48
2.8.3.17 mtext_ins()	48
2.8.3.18 mtext_insert()	49
2.8.3.19 mtext_ins_char()	49
2.8.3.20 mtext_replace()	50
2.8.3.21 mtext_character()	50
2.8.3.22 mtext_chr()	51
2.8.3.23 mtext_rchr()	51
2.8.3.24 mtext_cmp()	52
2.8.3.25 mtext_ncmp()	52
2.8.3.26 mtext_compare()	53
2.8.3.27 mtext_spn()	53
2.8.3.28 mtext_cspn()	53
2.8.3.29 mtext_pbrk()	54
2.8.3.30 mtext_tok()	54
2.8.3.31 mtext_text()	54
2.8.3.32 mtext_search()	55
2.8.3.33 mtext_casecmp()	55
2.8.3.34 mtext_ncasecmp()	55
2.8.3.35 mtext_case_compare()	56
2.8.3.36 mtext_lowercase()	56
2.8.3.37 mtext_titlecase()	57
2.8.3.38 mtext_uppercase()	57
2.8.4 変数詳解	57
2.8.4.1 MTEXT_FORMAT_UTF_16	58
2.8.4.2 MTEXT_FORMAT_UTF_32	58
2.8.4.3 Mlanguage	58
2.9 テキストプロパティ	58
2.9.1 詳解	60

2.9.2 型定義詳解	60
2.9.2.1 MTextPropSerializeFunc	60
2.9.2.2 MTextPropDeserializeFunc	61
2.9.3 列挙型詳解	61
2.9.3.1 MTextPropertyControl	61
2.9.4 関数詳解	61
2.9.4.1 mtext_get_prop()	62
2.9.4.2 mtext_get_prop_values()	63
2.9.4.3 mtext_get_prop_keys()	63
2.9.4.4 mtext_put_prop()	64
2.9.4.5 mtext_put_prop_values()	64
2.9.4.6 mtext_push_prop()	65
2.9.4.7 mtext_pop_prop()	66
2.9.4.8 mtext_prop_range()	66
2.9.4.9 mtext_property()	67
2.9.4.10 mtext_property_mtext()	67
2.9.4.11 mtext_property_key()	67
2.9.4.12 mtext_property_value()	68
2.9.4.13 mtext_property_start()	68
2.9.4.14 mtext_property_end()	68
2.9.4.15 mtext_get_property()	68
2.9.4.16 mtext_get_properties()	69
2.9.4.17 mtext_attach_property()	69
2.9.4.18 mtext_detach_property()	69
2.9.4.19 mtext_push_property()	70
2.9.4.20 mtext_serialize()	70
2.9.4.21 mtext_deserialize()	71
2.9.5 変数詳解	71
2.9.5.1 Mtext_prop_serializer	71
2.9.5.2 Mtext_prop_deserializer	72
2.10 データベース	72
2.10.1 詳解	73
2.10.2 型定義詳解	73
2.10.2.1 MDatabase	74
2.10.3 関数詳解	74
2.10.3.1 mdatabase_find()	74
2.10.3.2 mdatabase_list()	74
2.10.3.3 mdatabase_define()	75
2.10.3.4 mdatabase_load()	75
2.10.3.5 mdatabase_tag()	76

2.10.4 変数詳解	76
2.10.4.1 mdatabase_dir	76
2.11 シェル API	76
2.11.1 詳解	77
2.12 文字セット	77
2.12.1 詳解	79
2.12.2 マクロ定義詳解	79
2.12.2.1 MCHAR_INVALID_CODE	79
2.12.3 関数詳解	79
2.12.3.1 mchar_define_charset()	80
2.12.3.2 mchar_resolve_charset()	80
2.12.3.3 mchar_list_charset()	80
2.12.3.4 mchar_decode()	80
2.12.3.5 mchar_encode()	81
2.12.3.6 mchar_map_charset()	81
2.12.4 変数詳解	81
2.12.4.1 Mcharset_ascii	82
2.12.4.2 Mcharset_iso_8859_1	82
2.12.4.3 Mcharset_unicode	82
2.12.4.4 Mcharset_m17n	82
2.12.4.5 Mcharset_binary	82
2.12.4.6 Mmethod	83
2.12.4.7 Mdimension	83
2.12.4.8 Mmin_range	83
2.12.4.9 Mmax_range	83
2.12.4.10 Mmin_code	83
2.12.4.11 Mmax_code	83
2.12.4.12 Mascii_compatible	83
2.12.4.13 Mfinal_byte	84
2.12.4.14 Mrevision	84
2.12.4.15 Mmin_char	84
2.12.4.16 Mmapfile	84
2.12.4.17 Mparents	84
2.12.4.18 Msubset_offset	84
2.12.4.19 Mdefine_coding	84
2.12.4.20 Malias	85
2.12.4.21 Moffset	85
2.12.4.22 Mmap	85
2.12.4.23 Munify	85
2.12.4.24 Msubset	86

2.12.4.25 Msuperset	86
2.12.4.26 Mcharset	86
2.13 コード変換	86
2.13.1 詳解	90
2.13.2 列挙型詳解	90
2.13.2.1 MConversionResult	90
2.13.2.2 MCodingType	91
2.13.2.3 MCodingFlagISO2022	91
2.13.3 関数詳解	92
2.13.3.1 mconv_define_coding()	92
2.13.3.2 mconv_resolve_coding()	92
2.13.3.3 mconv_list_codings()	93
2.13.3.4 mconv_buffer_converter()	93
2.13.3.5 mconv_stream_converter()	94
2.13.3.6 mconv_reset_converter()	94
2.13.3.7 mconv_free_converter()	94
2.13.3.8 mconv_rebind_buffer()	95
2.13.3.9 mconv_rebind_stream()	95
2.13.3.10 mconv_decode()	96
2.13.3.11 mconv_decode_buffer()	96
2.13.3.12 mconv_decode_stream()	97
2.13.3.13 mconv_encode()	97
2.13.3.14 mconv_encode_range()	98
2.13.3.15 mconv_encode_buffer()	98
2.13.3.16 mconv_encode_stream()	99
2.13.3.17 mconv_getc()	99
2.13.3.18 mconv_ungetc()	100
2.13.3.19 mconv_putc()	100
2.13.3.20 mconv_gets()	101
2.13.4 変数詳解	101
2.13.4.1 Mcoding_us_ascii	101
2.13.4.2 Mcoding_iso_8859_1	101
2.13.4.3 Mcoding_utf_8	102
2.13.4.4 Mcoding_utf_8_full	102
2.13.4.5 Mcoding_utf_16	102
2.13.4.6 Mcoding_utf_16be	102
2.13.4.7 Mcoding_utf_16le	102
2.13.4.8 Mcoding_utf_32	103
2.13.4.9 Mcoding_utf_32be	103
2.13.4.10 Mcoding_utf_32le	103

2.13.4.11 Mcoding_sjis	103
2.13.4.12 Mtype	103
2.13.4.13 Mcharsets	103
2.13.4.14 Mflags	104
2.13.4.15 Mdesignation	104
2.13.4.16 Minvocation	104
2.13.4.17 Mcode_unit	104
2.13.4.18 Mbom	104
2.13.4.19 Mlittle_endian	104
2.13.4.20 Mutf	104
2.13.4.21 Miso_2022	105
2.13.4.22 Mreset_at_eol	105
2.13.4.23 Mreset_at_cnl	105
2.13.4.24 Meight_bit	105
2.13.4.25 Mlong_form	105
2.13.4.26 Mdesignation_g0	105
2.13.4.27 Mdesignation_g1	105
2.13.4.28 Mdesignation_ctxt	106
2.13.4.29 Mdesignation_ctxt_ext	106
2.13.4.30 Mlocking_shift	106
2.13.4.31 Msingle_shift	106
2.13.4.32 Msingle_shift_7	106
2.13.4.33 Meuc_tw_shift	106
2.13.4.34 Miso_6429	106
2.13.4.35 Mrevision_number	107
2.13.4.36 Mfull_support	107
2.13.4.37 Mmaybe	107
2.13.4.38 Mcoding	107
2.14 ロケール	107
2.14.1 詳解	108
2.14.2 型定義詳解	108
2.14.2.1 MLocale	109
2.14.3 関数詳解	109
2.14.3.1 mlanguage_jlist()	109
2.14.3.2 mlanguage_code()	109
2.14.3.3 mlanguage_name_list()	110
2.14.3.4 mlanguage_text()	110
2.14.3.5 mscript_list()	110
2.14.3.6 mscript_language_list()	111
2.14.3.7 mlocale_set()	111

2.14.3.8	<code>mlocale_get_prop()</code>	112
2.14.3.9	<code>mtext_ftime()</code>	112
2.14.3.10	<code>mtext_getenv()</code>	112
2.14.3.11	<code>mtext_putenv()</code>	113
2.14.3.12	<code>mtext_coll()</code>	113
2.14.4	変数詳解	113
2.14.4.1	<code>Miso639_1</code>	113
2.14.4.2	<code>Miso639_2</code>	113
2.14.4.3	<code>Mterritory</code>	114
2.14.4.4	<code>Mmodifier</code>	114
2.14.4.5	<code>Mcodeset</code>	114
2.15	入力メソッド (基本部分)	114
2.15.1	詳解	117
2.15.2	型定義詳解	118
2.15.2.1	<code>MInputCallbackFunc</code>	118
2.15.3	列挙型詳解	118
2.15.3.1	<code>MInputCandidatesChanged</code>	118
2.15.4	関数詳解	119
2.15.4.1	<code>minput_open_im()</code>	119
2.15.4.2	<code>minput_close_im()</code>	119
2.15.4.3	<code>minput_create_ic()</code>	119
2.15.4.4	<code>minput_destroy_ic()</code>	120
2.15.4.5	<code>minput_filter()</code>	120
2.15.4.6	<code>minput_lookup()</code>	120
2.15.4.7	<code>minput_set_spot()</code>	121
2.15.4.8	<code>minput_toggle()</code>	121
2.15.4.9	<code>minput_reset_ic()</code>	121
2.15.4.10	<code>minput_get_title_icon()</code>	122
2.15.4.11	<code>minput_get_description()</code>	122
2.15.4.12	<code>minput_get_command()</code>	123
2.15.4.13	<code>minput_config_command()</code>	124
2.15.4.14	<code>minput_get_variable()</code>	125
2.15.4.15	<code>minput_config_variable()</code>	126
2.15.4.16	<code>minput_config_file()</code>	127
2.15.4.17	<code>minput_save_config()</code>	128
2.15.4.18	<code>minput_list()</code>	128
2.15.4.19	<code>minput_get_variables()</code>	129
2.15.4.20	<code>minput_set_variable()</code>	130
2.15.4.21	<code>minput_get_commands()</code>	130
2.15.4.22	<code>minput_assign_command_keys()</code>	131

2.15.4.23 minput_parse_im_names()	131
2.15.4.24 minput_callback()	131
2.15.5 変数詳解	131
2.15.5.1 Minput_method	131
2.15.5.2 Minput_preedit_start	132
2.15.5.3 Minput_preedit_done	132
2.15.5.4 Minput_preedit_draw	132
2.15.5.5 Minput_status_start	132
2.15.5.6 Minput_status_done	132
2.15.5.7 Minput_status_draw	132
2.15.5.8 Minput_candidates_start	132
2.15.5.9 Minput_candidates_done	133
2.15.5.10 Minput_candidates_draw	133
2.15.5.11 Minput_set_spot	133
2.15.5.12 Minput_toggle	133
2.15.5.13 Minput_reset	133
2.15.5.14 Minput_get_surrounding_text	133
2.15.5.15 Minput_delete_surrounding_text	133
2.15.5.16 Minput_focus_out	134
2.15.5.17 Minput_focus_in	134
2.15.5.18 Minput_focus_move	134
2.15.5.19 Minherited	134
2.15.5.20 Mcustomized	134
2.15.5.21 Mconfigured	134
2.15.5.22 minput_default_driver	135
2.15.5.23 minput_driver	135
2.15.5.24 Minput_driver	135
2.16 FLT API	135
2.16.1 詳解	136
2.16.2 型定義詳解	137
2.16.2.1 MFLT	137
2.16.3 関数詳解	137
2.16.3.1 mflt_get()	137
2.16.3.2 mflt_find()	137
2.16.3.3 mflt_name()	138
2.16.3.4 mflt_coverage()	138
2.16.3.5 mflt_run()	138
2.16.3.6 mdebug_dump_flt()	138
2.16.3.7 mflt_dump_gstring()	139
2.16.4 変数詳解	139

2.16.4.1 mflt_enable_new_feature	139
2.16.4.2 mflt_iterate_otf_feature	139
2.16.4.3 mflt_font_id	139
2.16.4.4 mflt_try_otf	139
2.17 GUI API	140
2.17.1 詳解	140
2.18 フレーム	141
2.18.1 詳解	142
2.18.2 関数詳解	142
2.18.2.1 mframe()	142
2.18.2.2 mframe_get_prop()	144
2.18.3 変数詳解	144
2.18.3.1 Mdevice	144
2.18.3.2 Mdisplay	144
2.18.3.3 Mscreen	145
2.18.3.4 Mdrawable	145
2.18.3.5 Mdepth	145
2.18.3.6 Mcolormap	145
2.18.3.7 Mwidget	145
2.18.3.8 Mgd	145
2.18.3.9 Mfont	145
2.18.3.10 Mfont_width	146
2.18.3.11 Mfont_ascent	146
2.18.3.12 Mfont_descent	146
2.18.3.13 mframe_default	146
2.19 フォント	146
2.19.1 詳解	148
2.19.2 関数詳解	150
2.19.2.1 mfont()	151
2.19.2.2 mfont_parse_name()	151
2.19.2.3 mfont_unparse_name()	151
2.19.2.4 mfont_copy()	152
2.19.2.5 mfont_get_prop()	152
2.19.2.6 mfont_put_prop()	152
2.19.2.7 mfont_selection_priority()	153
2.19.2.8 mfont_set_selection_priority()	153
2.19.2.9 mfont_find()	153
2.19.2.10 mfont_set_encoding()	154
2.19.2.11 mfont_name()	154
2.19.2.12 mfont_from_name()	154

2.19.2.13 mfont_resize_ratio()	154
2.19.2.14 mfont_list()	155
2.19.2.15 mfont_list_family_names()	155
2.19.2.16 mfont_check()	155
2.19.2.17 mfont_match_p()	155
2.19.2.18 mfont_open()	156
2.19.2.19 mfont_encapsulate()	156
2.19.2.20 mfont_close()	156
2.19.3 変数詳解	156
2.19.3.1 Mfoundry	156
2.19.3.2 Mfamily	156
2.19.3.3 Mweight	157
2.19.3.4 Mstyle	157
2.19.3.5 Mstretch	157
2.19.3.6 Madstyle	157
2.19.3.7 Mspacing	157
2.19.3.8 Mregistry	158
2.19.3.9 Msize	158
2.19.3.10 Motf	158
2.19.3.11 Mfontfile	158
2.19.3.12 Mresolution	158
2.19.3.13 Mmax_advance	159
2.19.3.14 Mfontconfig	159
2.19.3.15 Mx	159
2.19.3.16 Mfreetype	159
2.19.3.17 Mxft	159
2.19.3.18 mfont_freetype_path	160
2.20 フォントセット	160
2.20.1 詳解	160
2.20.2 関数詳解	161
2.20.2.1 mfontset()	161
2.20.2.2 mfontset_name()	161
2.20.2.3 mfontset_copy()	161
2.20.2.4 mfontset_modify_entry()	162
2.20.2.5 mfontset_lookup()	163
2.21 フェース	163
2.21.1 詳解	166
2.21.2 型定義詳解	166
2.21.2.1 MFaceHookFunc	167
2.21.3 関数詳解	167

2.21.3.1 mface()	167
2.21.3.2 mface_copy()	167
2.21.3.3 mface_equal()	167
2.21.3.4 mface_merge()	168
2.21.3.5 mface_from_font()	168
2.21.3.6 mface_get_prop()	168
2.21.3.7 mface_get_hook()	169
2.21.3.8 mface_put_prop()	169
2.21.3.9 mface_put_hook()	169
2.21.3.10 mface_update()	170
2.21.4 変数詳解	170
2.21.4.1 Mforeground	170
2.21.4.2 Mbackground	170
2.21.4.3 Mvideomode	170
2.21.4.4 Mratio	171
2.21.4.5 Mhline	171
2.21.4.6 Mbox	171
2.21.4.7 Mfontset	171
2.21.4.8 Mhook_func	172
2.21.4.9 Mhook_arg	172
2.21.4.10 Mnormal	172
2.21.4.11 Mreverse	172
2.21.4.12 mface_normal_video	172
2.21.4.13 mface_reverse_video	173
2.21.4.14 mface_underline	173
2.21.4.15 mface_medium	173
2.21.4.16 mface_bold	173
2.21.4.17 mface_italic	174
2.21.4.18 mface_bold_italic	174
2.21.4.19 mface_xx_small	174
2.21.4.20 mface_x_small	174
2.21.4.21 mface_small	174
2.21.4.22 mface_normalsize	175
2.21.4.23 mface_large	175
2.21.4.24 mface_x_large	175
2.21.4.25 mface_xx_large	175
2.21.4.26 mface_black	175
2.21.4.27 mface_white	176
2.21.4.28 mface_red	176
2.21.4.29 mface_green	176

2.21.4.30 mface_blue	176
2.21.4.31 mface_cyan	176
2.21.4.32 mface_yellow	177
2.21.4.33 mface_magenta	177
2.21.4.34 Mface	177
2.22 表示	177
2.22.1 詳解	179
2.22.2 型定義詳解	179
2.22.2.1 MDrawWindow	179
2.22.2.2 MDrawRegion	179
2.22.3 関数詳解	179
2.22.3.1 mdraw_text()	180
2.22.3.2 mdraw_image_text()	181
2.22.3.3 mdraw_text_with_control()	182
2.22.3.4 mdraw_text_extents()	182
2.22.3.5 mdraw_text_per_char_extents()	183
2.22.3.6 mdraw_coordinates_position()	183
2.22.3.7 mdraw_glyph_info()	184
2.22.3.8 mdraw_glyph_list()	184
2.22.3.9 mdraw_text_items()	185
2.22.3.10 mdraw_default_line_break()	185
2.22.3.11 mdraw_per_char_extents()	186
2.22.3.12 mdraw_clear_cache()	186
2.22.4 変数詳解	186
2.22.4.1 mdraw_line_break_option	186
2.23 入力メソッド (GUI)	187
2.23.1 詳解	187
2.23.2 関数詳解	188
2.23.2.1 minput_event_to_key()	188
2.23.3 変数詳解	188
2.23.3.1 minput_gui_driver	188
2.23.3.2 Mxim	189
2.24 MISC API	189
2.24.1 詳解	189
2.25 エラー処理	189
2.25.1 詳解	190
2.25.2 列挙型詳解	191
2.25.2.1 MErrorCode	191
2.25.3 変数詳解	192
2.25.3.1 merror_code	192

2.25.3.2 m17n_memory_full_handler	192
2.26 デバッグサポート	192
2.26.1 詳解	193
2.26.2 関数詳解	194
2.26.2.1 mdebug_dump_face()	194
2.26.2.2 mdebug_dump_im()	194
2.26.2.3 mdebug_hook()	194
2.26.2.4 mdebug_dump_mtext()	195
2.26.2.5 mdebug_dump_symbol()	195
2.26.2.6 mdebug_dump_all_symbols()	195
3 データ構造詳解	197
3.1 M17NObject 構造体	197
3.1.1 フィールド詳解	197
3.1.1.1 ref_count	198
3.1.1.2 ref_count_extended	198
3.1.1.3 flag	198
3.1.1.4 freer	198
3.1.1.5 record	198
3.1.1.6	198
3.2 M17NObjectArray 構造体	198
3.2.1 フィールド詳解	199
3.2.1.1 name	199
3.2.1.2 count	199
3.2.1.3 size	199
3.2.1.4 inc	199
3.2.1.5 used	199
3.2.1.6 objects	199
3.2.1.7 next	199
3.3 M17NObjectHead 構造体	200
3.3.1 詳解	200
3.3.2 フィールド詳解	200
3.3.2.1 filler	200
3.4 M17NObjectRecord 構造体	200
3.4.1 フィールド詳解	200
3.4.1.1 freer	201
3.4.1.2 size	201
3.4.1.3 inc	201
3.4.1.4 used	201
3.4.1.5 counts	201

3.5 MCharset 構造体	201
3.5.1 フィールド詳解	202
3.5.1.1 ref_count	202
3.5.1.2 name	202
3.5.1.3 dimension	202
3.5.1.4 code_range	203
3.5.1.5 code_range_min_code	203
3.5.1.6 no_code_gap	203
3.5.1.7 code_range_mask	203
3.5.1.8 min_code	203
3.5.1.9 max_code	203
3.5.1.10 ascii_compatible	203
3.5.1.11 min_char	204
3.5.1.12 max_char	204
3.5.1.13 final_byte	204
3.5.1.14 revision	204
3.5.1.15 method	204
3.5.1.16 decoder	204
3.5.1.17 encoder	204
3.5.1.18 unified_max	205
3.5.1.19 parents	205
3.5.1.20 nparents	205
3.5.1.21 subset_min_code	205
3.5.1.22 subset_max_code	205
3.5.1.23 subset_offset	205
3.5.1.24 simple	205
3.5.1.25 fully_loaded	206
3.6 MCharsetISO2022Table 構造体	206
3.6.1 フィールド詳解	206
3.6.1.1 size	206
3.6.1.2 inc	207
3.6.1.3 used	207
3.6.1.4 charsets	207
3.6.1.5 classified	207
3.7 MCodingInfoISO2022 構造体	207
3.7.1 詳解	207
3.7.2 フィールド詳解	207
3.7.2.1 initial_invocation	208
3.7.2.2 designations	208
3.7.2.3 flags	208

3.8 MCodingInfoUTF 構造体	208
3.8.1 詳解	208
3.8.2 フィールド詳解	208
3.8.2.1 code_unit_bits	209
3.8.2.2 bom	209
3.8.2.3 endian	209
3.9 MConverter 構造体	209
3.9.1 詳解	210
3.9.2 フィールド詳解	210
3.9.2.1 lenient	210
3.9.2.2 last_block	210
3.9.2.3 at_most	211
3.9.2.4 nchars	211
3.9.2.5 nbytes	211
3.9.2.6 result	211
3.9.2.7 ptr	211
3.9.2.8 dbl	211
3.9.2.9 c	211
3.9.2.10	212
3.9.2.11 internal_info	212
3.10 MDatabaseInfo 構造体	212
3.10.1 フィールド詳解	213
3.10.1.1 filename	213
3.10.1.2 len	213
3.10.1.3 absolute_filename	213
3.10.1.4 status	213
3.10.1.5 time	213
3.10.1.6 lock_file	213
3.10.1.7 uniq_file	214
3.10.1.8 properties	214
3.11 MDeviceDriver 構造体	214
3.11.1 フィールド詳解	215
3.11.1.1 close	215
3.11.1.2 get_prop	215
3.11.1.3 realize_face	215
3.11.1.4 free_realized_face	215
3.11.1.5 fill_space	216
3.11.1.6 draw_empty_boxes	216
3.11.1.7 draw_hline	216
3.11.1.8 draw_box	216

3.11.1.9 draw_points	216
3.11.1.10 region_from_rect	216
3.11.1.11 union_rect_with_region	216
3.11.1.12 intersect_region	217
3.11.1.13 region_add_rect	217
3.11.1.14 region_to_rect	217
3.11.1.15 free_region	217
3.11.1.16 dump_region	217
3.11.1.17 create_window	217
3.11.1.18 destroy_window	217
3.11.1.19 map_window	218
3.11.1.20 unmap_window	218
3.11.1.21 window_geometry	218
3.11.1.22 adjust_window	218
3.11.1.23 parse_event	218
3.12 MDrawControl 構造体	218
3.12.1 詳解	219
3.12.2 フィールド詳解	219
3.12.2.1 as_image	219
3.12.2.2 align_head	220
3.12.2.3 two_dimensional	220
3.12.2.4 orientation_reversed	220
3.12.2.5 enable_bidi	220
3.12.2.6 ignore_formatting_char	220
3.12.2.7 fixed_width	220
3.12.2.8 anti_alias	220
3.12.2.9 disable_overlapping_adjustment	221
3.12.2.10 min_line_ascent	221
3.12.2.11 min_line_descent	221
3.12.2.12 max_line_ascent	221
3.12.2.13 max_line_descent	221
3.12.2.14 max_line_width	221
3.12.2.15 tab_width	221
3.12.2.16 format	222
3.12.2.17 line_break	222
3.12.2.18 with_cursor	222
3.12.2.19 cursor_pos	222
3.12.2.20 cursor_width	222
3.12.2.21 cursor_bidi	223
3.12.2.22 partial_update	223

3.12.2.23 disable_caching	223
3.12.2.24 clip_region	223
3.13 MDrawGlyph 構造体	223
3.13.1 詳解	224
3.13.2 フィールド詳解	225
3.13.2.1 from	225
3.13.2.2 to	225
3.13.2.3 glyph_code	225
3.13.2.4 x_advance	225
3.13.2.5 y_advance	225
3.13.2.6 x_off	225
3.13.2.7 y_off	226
3.13.2.8 lbearing	226
3.13.2.9 rbearing	226
3.13.2.10 ascent	226
3.13.2.11 descent	226
3.13.2.12 font	226
3.13.2.13 font_type	226
3.13.2.14 fontp	227
3.14 MDrawGlyphInfo 構造体	227
3.14.1 詳解	228
3.14.2 フィールド詳解	228
3.14.2.1 from	228
3.14.2.2 to	228
3.14.2.3 line_from	228
3.14.2.4 line_to	228
3.14.2.5 x	228
3.14.2.6 y	229
3.14.2.7 metrics	229
3.14.2.8 font	229
3.14.2.9 prev_from	229
3.14.2.10 next_to	229
3.14.2.11 left_from	229
3.14.2.12 left_to	229
3.14.2.13 right_from	230
3.14.2.14 right_to	230
3.14.2.15 logical_width	230
3.15 MDrawMetric 構造体	230
3.15.1 詳解	230
3.15.2 フィールド詳解	230

3.15.2.1 x	231
3.15.2.2 y	231
3.15.2.3 width	231
3.15.2.4 height	231
3.16 MDrawPoint 構造体	231
3.16.1 フィールド詳解	231
3.16.1.1 x	231
3.16.1.2 y	232
3.17 MDrawTextItem 構造体	232
3.17.1 詳解	232
3.17.2 フィールド詳解	233
3.17.2.1 mt	233
3.17.2.2 delta	233
3.17.2.3 face	233
3.17.2.4 control	233
3.18 MFace 構造体	233
3.18.1 詳解	234
3.18.2 フィールド詳解	234
3.18.2.1 control	234
3.18.2.2 property	235
3.18.2.3 hook	235
3.18.2.4 frame_list	235
3.19 MFaceBoxProp 構造体	235
3.19.1 詳解	236
3.19.2 フィールド詳解	236
3.19.2.1 width	236
3.19.2.2 color_top	236
3.19.2.3 color_bottom	236
3.19.2.4 color_left	236
3.19.2.5 color_right	236
3.19.2.6 inner_hmargin	237
3.19.2.7 inner_vmargin	237
3.19.2.8 outer_hmargin	237
3.19.2.9 outer_vmargin	237
3.20 MFaceHLineProp 構造体	237
3.20.1 詳解	238
3.20.2 列挙型メンバ詳解	238
3.20.2.1 MFaceHLineType	238
3.20.3 フィールド詳解	238
3.20.3.1 type	239

3.20.3.2 width	239
3.20.3.3 color	239
3.21 MFLTFont 構造体	239
3.21.1 詳解	240
3.21.2 フィールド詳解	240
3.21.2.1 family	240
3.21.2.2 x_ppem	240
3.21.2.3 y_ppem	240
3.21.2.4 get_glyph_id	240
3.21.2.5 get_metrics	241
3.21.2.6 check_otf	241
3.21.2.7 drive_otf	241
3.21.2.8 internal	241
3.22 MFLTFontForRealized 構造体	241
3.22.1 フィールド詳解	242
3.22.1.1 font	242
3.22.1.2 rfont	242
3.23 MFLTGlyph 構造体	242
3.23.1 詳解	243
3.23.2 フィールド詳解	243
3.23.2.1 c	243
3.23.2.2 code	243
3.23.2.3 from	243
3.23.2.4 to	243
3.23.2.5 xadv	243
3.23.2.6 yadv	244
3.23.2.7 ascent	244
3.23.2.8 descent	244
3.23.2.9 lbearing	244
3.23.2.10 rbearing	244
3.23.2.11 xoff	244
3.23.2.12 yoff	244
3.23.2.13 encoded	245
3.23.2.14 measured	245
3.23.2.15 adjusted	245
3.23.2.16 internal	245
3.24 MFLTGlyphAdjustment 構造体	245
3.24.1 詳解	245
3.24.2 フィールド詳解	246
3.24.2.1 xadv	246

3.24.2.2 yadv	246
3.24.2.3 xoff	246
3.24.2.4 yoff	246
3.24.2.5 back	246
3.24.2.6 advance_js_absolute	246
3.24.2.7 set	247
3.25 MFLTGlyphString 構造体	247
3.25.1 詳解	247
3.25.2 フィールド詳解	247
3.25.2.1 glyph_size	248
3.25.2.2 glyphs	248
3.25.2.3 allocated	248
3.25.2.4 used	248
3.25.2.5 r2l	248
3.26 MFLTOtfSpec 構造体	248
3.26.1 詳解	249
3.26.2 フィールド詳解	249
3.26.2.1 sym	249
3.26.2.2 script	249
3.26.2.3 langsys	249
3.26.2.4 features	250
3.27 MFont 構造体	250
3.27.1 詳解	251
3.27.2 フィールド詳解	251
3.27.2.1 property	251
3.27.2.2 type	251
3.27.2.3 source	251
3.27.2.4 spacing	251
3.27.2.5 for_full_width	252
3.27.2.6 multiple_sizes	252
3.27.2.7 size	252
3.27.2.8 file	252
3.27.2.9 capability	252
3.27.2.10 encoding	252
3.28 MFontCapability 構造体	253
3.28.1 フィールド詳解	254
3.28.1.1 control	254
3.28.1.2 language	254
3.28.1.3 script	254
3.28.1.4 otf	254

3.28.1.5 script_tag	254
3.28.1.6 langsys_tag	254
3.28.1.7 str	255
3.28.1.8 nfeatures	255
3.28.1.9 tags	255
3.28.1.10	255
3.29 MFontDriver 構造体	255
3.29.1 フィールド詳解	256
3.29.1.1 select	256
3.29.1.2 open	256
3.29.1.3 find_metric	256
3.29.1.4 has_char	257
3.29.1.5 encode_char	257
3.29.1.6 render	257
3.29.1.7 list	257
3.29.1.8 list_family_names	257
3.29.1.9 check_capability	257
3.29.1.10 encapsulate	257
3.29.1.11 close	258
3.29.1.12 check_otf	258
3.29.1.13 drive_otf	258
3.29.1.14 try_otf	258
3.29.1.15 iterate_otf_feature	258
3.30 MFontList 構造体	259
3.30.1 フィールド詳解	259
3.30.1.1 object	259
3.30.1.2 fonts	260
3.30.1.3 nfonts	260
3.31 MFontPropertyTable 構造体	260
3.31.1 フィールド詳解	260
3.31.1.1 size	261
3.31.1.2 inc	261
3.31.1.3 used	261
3.31.1.4 property	261
3.31.1.5 names	261
3.32 MFontScore 構造体	262
3.32.1 フィールド詳解	262
3.32.1.1 font	262
3.32.1.2 score	262
3.33 MFrame 構造体	263

3.33.1 詳解	264
3.33.2 フィールド詳解	264
3.33.2.1 control	264
3.33.2.2 foreground	264
3.33.2.3 background	264
3.33.2.4 videomode	264
3.33.2.5 font	264
3.33.2.6 face	265
3.33.2.7 rface	265
3.33.2.8 space_width	265
3.33.2.9 average_width	265
3.33.2.10 ascent	265
3.33.2.11 descent	265
3.33.2.12 tick	265
3.33.2.13 device	266
3.33.2.14 device_type	266
3.33.2.15 dpi	266
3.33.2.16 driver	266
3.33.2.17 font_driver_list	266
3.33.2.18 realized_font_list	266
3.33.2.19 realized_face_list	266
3.33.2.20 realized_fontset_list	267
3.34 MGlyph 構造体	267
3.34.1 フィールド詳解	267
3.34.1.1 g	268
3.34.1.2 rface	268
3.34.1.3 left_padding	268
3.34.1.4 right_padding	268
3.34.1.5 enabled	268
3.34.1.6 bidi_level	268
3.34.1.7 category	268
3.34.1.8 type	269
3.34.1.9 libotf_positioning_type	269
3.35 MGlyphString 構造体	269
3.35.1 フィールド詳解	270
3.35.1.1 head	270
3.35.1.2 frame	270
3.35.1.3 tick	270
3.35.1.4 size	270
3.35.1.5 inc	271

3.35.1.6 used	271
3.35.1.7 glyphs	271
3.35.1.8 from	271
3.35.1.9 to	271
3.35.1.10 width	271
3.35.1.11 height	271
3.35.1.12 ascent	272
3.35.1.13 descent	272
3.35.1.14 physical_ascent	272
3.35.1.15 physical_descent	272
3.35.1.16 lbearing	272
3.35.1.17 rbearing	272
3.35.1.18 text_ascent	272
3.35.1.19 text_descent	273
3.35.1.20 line_ascent	273
3.35.1.21 line_descent	273
3.35.1.22 indent	273
3.35.1.23 widthLimit	273
3.35.1.24 anti_alias	273
3.35.1.25 control	273
3.35.1.26 next	274
3.35.1.27 top	274
3.36 MInputContext 構造体	274
3.36.1 詳解	275
3.36.2 フィールド詳解	275
3.36.2.1 im	275
3.36.2.2 produced	276
3.36.2.3 arg	276
3.36.2.4 active	276
3.36.2.5 x	276
3.36.2.6 y	276
3.36.2.7 ascent	276
3.36.2.8 descent	276
3.36.2.9 fontsize	277
3.36.2.10 mt	277
3.36.2.11 pos	277
3.36.2.12	277
3.36.2.13 info	277
3.36.2.14 status	277
3.36.2.15 status_changed	277

3.36.2.16 preedit	278
3.36.2.17 preedit_changed	278
3.36.2.18 cursor_pos	278
3.36.2.19 cursor_pos_changed	278
3.36.2.20 candidate_list	278
3.36.2.21 candidate_index	278
3.36.2.22 candidate_from	278
3.36.2.23 candidate_to	279
3.36.2.24 candidate_show	279
3.36.2.25 candidates_changed	279
3.36.2.26 plist	279
3.37 MInputContextInfo 構造体	280
3.37.1 フィールド詳解	281
3.37.1.1 state	281
3.37.1.2 prev_state	281
3.37.1.3 map	281
3.37.1.4 size	281
3.37.1.5 inc	281
3.37.1.6 used	282
3.37.1.7 keys	282
3.37.1.8 state_key_head	282
3.37.1.9 key_head	282
3.37.1.10 commit_key_head	282
3.37.1.11 preedit_saved	282
3.37.1.12 state_pos	282
3.37.1.13 markers	283
3.37.1.14 vars	283
3.37.1.15 vars_saved	283
3.37.1.16 preceding_text	283
3.37.1.17 following_text	283
3.37.1.18 key_unhandled	283
3.37.1.19 win_info	283
3.37.1.20 state_hook	284
3.37.1.21 tick	284
3.37.1.22 pushing_or_switching	284
3.37.1.23 fallbacks	284
3.37.1.24 stack	284
3.38 MInputDriver 構造体	284
3.38.1 詳解	285
3.38.2 フィールド詳解	286

3.38.2.1 open_lm	286
3.38.2.2 close_lm	286
3.38.2.3 create_ic	286
3.38.2.4 destroy_ic	286
3.38.2.5 filter	287
3.38.2.6 lookup	287
3.38.2.7 callback_list	287
3.39 MInputGUIArgIC 構造体	288
3.39.1 詳解	288
3.39.2 フィールド詳解	288
3.39.2.1 frame	288
3.39.2.2 client	289
3.39.2.3 focus	289
3.40 MInputMethod 構造体	289
3.40.1 詳解	290
3.40.2 フィールド詳解	290
3.40.2.1 language	290
3.40.2.2 name	290
3.40.2.3 driver	290
3.40.2.4 arg	290
3.40.2.5 info	291
3.41 MInputMethodInfo 構造体	291
3.41.1 フィールド詳解	292
3.41.1.1 mdb	292
3.41.1.2 language	292
3.41.1.3 name	292
3.41.1.4 extra	292
3.41.1.5 cmds	293
3.41.1.6 configured_cmds	293
3.41.1.7 bc_cmds	293
3.41.1.8 vars	293
3.41.1.9 configured_vars	293
3.41.1.10 bc_vars	293
3.41.1.11 description	293
3.41.1.12 title	294
3.41.1.13 maps	294
3.41.1.14 states	294
3.41.1.15 macros	294
3.41.1.16 externals	294
3.41.1.17 tick	294

3.42 MInputXIMArgIC 構造体	294
3.42.1 詳解	295
3.42.2 フィールド詳解	295
3.42.2.1 input_style	295
3.42.2.2 client_win	295
3.42.2.3 focus_win	295
3.42.2.4 preedit_attrs	295
3.42.2.5 status_attrs	295
3.43 MInputXIMArgIM 構造体	296
3.43.1 詳解	296
3.43.2 フィールド詳解	296
3.43.2.1 display	296
3.43.2.2 db	296
3.43.2.3 res_class	296
3.43.2.4 res_name	297
3.43.2.5 locale	297
3.43.2.6 modifier_list	297
3.44 MPlist 構造体	297
3.44.1 詳解	298
3.44.2 フィールド詳解	298
3.44.2.1 control	298
3.44.2.2 key	298
3.44.2.3 pointer	298
3.44.2.4 func	298
3.44.2.5	299
3.44.2.6 next	299
3.45 MRealizedFace 構造体	299
3.45.1 フィールド詳解	300
3.45.1.1 frame	300
3.45.1.2 face	300
3.45.1.3 font	300
3.45.1.4 base_face_list	300
3.45.1.5 rfont	300
3.45.1.6 rfontset	301
3.45.1.7 layouter	301
3.45.1.8 hline	301
3.45.1.9 box	301
3.45.1.10 ascii_rface	301
3.45.1.11 non_ascii_list	301
3.45.1.12 ascent	301

3.45.1.13 descent	302
3.45.1.14 space_width	302
3.45.1.15 average_width	302
3.45.1.16 info	302
3.46 MRealizedFont 構造体	302
3.46.1 フィールド詳解	303
3.46.1.1 spec	303
3.46.1.2 id	303
3.46.1.3 frame	303
3.46.1.4 font	303
3.46.1.5 driver	304
3.46.1.6 layouter	304
3.46.1.7 encapsulating	304
3.46.1.8 info	304
3.46.1.9 x_ppem	304
3.46.1.10 y_ppem	304
3.46.1.11 ascent	304
3.46.1.12 descent	305
3.46.1.13 max_advance	305
3.46.1.14 average_width	305
3.46.1.15 baseline_offset	305
3.46.1.16 fontp	305
3.46.1.17 next	305
3.47 MSymbol 構造体	306
3.47.1 詳解	306
3.47.2 フィールド詳解	306
3.47.2.1 managing_key	306
3.47.2.2 name	307
3.47.2.3 length	307
3.47.2.4 plist	307
3.47.2.5 next	307
3.48 MText 構造体	307
3.48.1 詳解	308
3.48.2 フィールド詳解	308
3.48.2.1 control	308
3.48.2.2 format	308
3.48.2.3 coverage	308
3.48.2.4 nchars	308
3.48.2.5 nbytes	309
3.48.2.6 data	309

3.48.2.7 allocated	309
3.48.2.8 plist	309
3.48.2.9 cache_char_pos	309
3.48.2.10 cache_byte_pos	309
3.49 MTextProperty 構造体	310
3.49.1 詳解	310
3.49.2 フィールド詳解	311
3.49.2.1 control	311
3.49.2.2 attach_count	311
3.49.2.3 mt	311
3.49.2.4 start	311
3.49.2.5 end	311
3.49.2.6 key	311
3.49.2.7 val	311
4 ファイル詳解	313
4.1 character.c ファイル	313
4.2 character.h ファイル	314
4.2.1 マクロ定義詳解	315
4.2.1.1 MAX_UTF8_CHAR_BYTES	315
4.2.1.2 MAX_UNICODE_CHAR_BYTES	315
4.2.1.3 USHORT_SIZE	315
4.2.1.4 UINT_SIZE	315
4.2.1.5 UNIT_BYTES	316
4.2.1.6 CHAR_UNITS_ASCII	316
4.2.1.7 CHAR_UNITS_UTF8	316
4.2.1.8 CHAR_UNITS_UTF16	316
4.2.1.9 CHAR_UNITS_UTF32	316
4.2.1.10 CHAR_UNITS	317
4.2.1.11 CHAR_BYTES	317
4.2.1.12 CHAR_UNITS_AT_UTF8	317
4.2.1.13 CHAR_UNITS_AT_UTF16	317
4.2.1.14 CHAR_UNITS_AT	317
4.2.1.15 CHAR_BYTES_AT	318
4.2.1.16 CHAR_UNITS_BY_HEAD_UTF8	318
4.2.1.17 CHAR_UNITS_BY_HEAD_UTF16	318
4.2.1.18 CHAR_UNITS_BY_HEAD	318
4.2.1.19 CHAR_BYTES_BY_HEAD	318
4.2.1.20 STRING_CHAR_UTF8	319
4.2.1.21 STRING_CHAR_UTF16	319

4.2.1.22	STRING_CHAR	319
4.2.1.23	STRING_CHAR_ADVANCE_UTF8	319
4.2.1.24	STRING_CHAR_ADVANCE_UTF16	320
4.2.1.25	STRING_CHAR_ADVANCE	320
4.2.1.26	STRING_CHAR_AND_UNITS_UTF8	320
4.2.1.27	STRING_CHAR_AND_UNITS_UTF16	321
4.2.1.28	STRING_CHAR_AND_UNITS	321
4.2.1.29	STRING_CHAR_AND_BYTES	321
4.2.1.30	CHAR_STRING_UTF8	321
4.2.1.31	CHAR_STRING_UTF16	322
4.2.1.32	CHAR_STRING	322
4.2.1.33	CHAR_HEAD_P_UTF8	322
4.2.1.34	CHAR_HEAD_P_UTF16	322
4.2.1.35	CHAR_HEAD_P	322
4.2.1.36	TOLOWER	322
4.2.1.37	TOUPPER	323
4.2.1.38	ISUPPER	323
4.2.1.39	ISALNUM	323
4.2.2	関数詳解	323
4.2.2.1	mchar__define_prop()	323
4.3	charset.c ファイル	323
4.4	charset.h ファイル	325
4.4.1	マクロ定義詳解	326
4.4.1.1	MCHARSET	326
4.4.1.2	CODE_POINT_TO_INDEX	327
4.4.1.3	INDEX_TO_CODE_POINT	327
4.4.1.4	DECODE_CHAR	327
4.4.1.5	ENCODE_CHAR	328
4.4.1.6	ISO_MAX_DIMENSION	328
4.4.1.7	ISO_MAX_CHARS	328
4.4.1.8	ISO_MAX_FINAL	328
4.4.1.9	MCHARSET_ISO_2022	328
4.4.2	列挙型詳解	328
4.4.2.1	mcharset_method	328
4.4.3	関数詳解	329
4.4.3.1	mcharset_find()	329
4.4.3.2	mcharset__decode_char()	329
4.4.3.3	mcharset__encode_char()	329
4.4.3.4	mcharset__load_from_database()	329
4.4.4	変数詳解	329

4.4.4.1 mcharset__cache	330
4.4.4.2 mcharset__ascii	330
4.4.4.3 mcharset__binary	330
4.4.4.4 mcharset__m17n	330
4.4.4.5 mcharset__unicode	330
4.4.4.6 mcharset__iso_2022_table	330
4.5 chartab.c ファイル	331
4.5.1 関数詳解	331
4.5.1.1 mdebug_dump_chartab()	331
4.6 chartab.h ファイル	332
4.6.1 関数詳解	332
4.6.1.1 mchartable_lookup()	332
4.7 coding.c ファイル	332
4.8 coding.h ファイル	334
4.8.1 関数詳解	335
4.8.1.1 mconv_register_charset_coding()	335
4.8.1.2 mcoding_load_from_database()	335
4.9 database.c ファイル	335
4.10 database.h ファイル	336
4.10.1 マクロ定義詳解	336
4.10.1.1 M17NDIR	337
4.10.1.2 PATH_MAX	337
4.10.1.3 PATH_SEPARATOR	337
4.10.2 列挙型詳解	337
4.10.2.1 MDatabaseStatus	337
4.10.3 関数詳解	337
4.10.3.1 mdatabase_update()	337
4.10.3.2 mdatabase_load_for_keys()	338
4.10.3.3 mdatabase_check()	338
4.10.3.4 mdatabase_find_file()	338
4.10.3.5 mdatabase_file()	338
4.10.3.6 mdatabase_lock()	338
4.10.3.7 mdatabase_save()	338
4.10.3.8 mdatabase_unlock()	339
4.10.3.9 mdatabase_props()	339
4.10.4 変数詳解	339
4.10.4.1 mdatabase_dir_list	339
4.10.4.2 mdatabase_load_charset_func	339
4.11 dbdata.txt ファイル	339
4.12 dbformat.txt ファイル	339

4.13 dbtutorial.txt ファイル	339
4.14 draw.c ファイル	339
4.15 exprog.txt ファイル	340
4.16 face.c ファイル	340
4.17 face.h ファイル	342
4.17.1 列挙型詳解	343
4.17.1.1 MFaceProperty	343
4.17.2 関数詳解	344
4.17.2.1 mface_realize()	344
4.17.2.2 mface_for_chars()	344
4.17.2.3 mface_free_realized()	344
4.17.2.4 mface_update_frame_face()	345
4.17.3 変数詳解	345
4.17.3.1 mface_default	345
4.18 fdl.txt ファイル	345
4.19 font.c ファイル	345
4.19.1 関数詳解	347
4.19.1.1 mdebug_dump_font()	347
4.20 font.h ファイル	347
4.20.1 マクロ定義詳解	349
4.20.1.1 FONT_PROPERTY	349
4.20.1.2 MFONT_INIT	349
4.20.2 型定義詳解	349
4.20.2.1 MFontEncoding	349
4.20.2.2 OTF_Tag	350
4.20.3 列挙型詳解	350
4.20.3.1 MFontProperty	350
4.20.3.2 MFontType	350
4.20.3.3 MFontSource	350
4.20.3.4 MFontSpacing	351
4.20.3.5 MFontOpenTypeTable	351
4.20.4 関数詳解	351
4.20.4.1 mfont_ftl_init()	351
4.20.4.2 mfont_ftl_fini()	352
4.20.4.3 mfont_free_realized()	352
4.20.4.4 mfont_match_p()	352
4.20.4.5 mfont_merge()	352
4.20.4.6 mfont_set_spec_from_face()	352
4.20.4.7 mfont_set_spec_from_plist()	352
4.20.4.8 mfont_has_char()	353

4.20.4.9 mfont_encode_char()	353
4.20.4.10 mfont_get_glyph_id()	353
4.20.4.11 mfont_select()	353
4.20.4.12 mfont_list()	353
4.20.4.13 mfont_open()	354
4.20.4.14 mfont_get_metric()	354
4.20.4.15 mfont_get_metrics()	354
4.20.4.16 mfont_set_property()	354
4.20.4.17 mfont_split_name()	354
4.20.4.18 mfont_parse_name_into_font()	355
4.20.4.19 mfont_encoding_list()	355
4.20.4.20 mfont_get_capability()	355
4.20.4.21 mfont_check_capability()	355
4.20.4.22 mfont_ft_encode_char()	355
4.20.4.23 mfont_ft_run()	355
4.20.5 変数詳解	356
4.20.5.1 mfont_property_table	356
4.20.5.2 Mlayouter	356
4.20.5.3 Miso8859_1	356
4.20.5.4 Miso10646_1	356
4.20.5.5 Municode_bmp	356
4.20.5.6 Municode_full	356
4.20.5.7 Mapple_roman	356
4.21 fontset.c ファイル	357
4.21.1 関数詳解	357
4.21.1.1 mdebug_dump_fontset()	357
4.22 fontset.h ファイル	357
4.22.1 関数詳解	358
4.22.1.1 mfont_realize_fontset()	358
4.22.1.2 mfont_free_realized_fontset()	358
4.22.1.3 mfont_lookup_fontset()	358
4.22.1.4 mfontset_get_font()	358
4.23 input-gui.c ファイル	358
4.24 input.c ファイル	359
4.25 input.h ファイル	361
4.25.1 マクロ定義詳解	361
4.25.1.1 MINPUT_KEY_SHIFT_MODIFIER	362
4.25.1.2 MINPUT_KEY_CONTROL_MODIFIER	362
4.25.1.3 MINPUT_KEY_META_MODIFIER	362
4.25.1.4 MINPUT_KEY_ALT_MODIFIER	362

4.25.1.5 MINPUT_KEY_SUPER_MODIFIER	362
4.25.1.6 MINPUT_KEY_HYPER_MODIFIER	362
4.25.1.7 MINPUT_KEY_ALTGR_MODIFIER	362
4.25.2 型定義詳解	362
4.25.2.1 MIMState	363
4.25.2.2 MIMMap	363
4.25.2.3 MIMInputStack	363
4.25.3 関数詳解	363
4.25.3.1 minput_char_to_key()	363
4.26 internal-flt.h ファイル	363
4.26.1 マクロ定義詳解	364
4.26.1.1 MAKE_COMBINING_CODE	364
4.26.1.2 COMBINING_CODE_OFF_Y	364
4.26.1.3 COMBINING_CODE_OFF_X	364
4.26.1.4 COMBINING_CODE_BASE_X	364
4.26.1.5 COMBINING_CODE_BASE_Y	364
4.26.1.6 COMBINING_CODE_ADD_X	365
4.26.1.7 COMBINING_CODE_ADD_Y	365
4.26.1.8 PACK_OTF_TAG	365
4.26.2 変数詳解	365
4.26.2.1 Mcombining	365
4.27 internal-gui.h ファイル	365
4.27.1 マクロ定義詳解	367
4.27.1.1 M_CHECK_WRITABLE	367
4.27.1.2 M_CHECK_READABLE	367
4.27.1.3 MGLYPH	367
4.27.1.4 GLYPH_INDEX	367
4.27.1.5 INIT_GLYPH	368
4.27.1.6 APPEND_GLYPH	368
4.27.1.7 INSERT_GLYPH	368
4.27.1.8 DELETE_GLYPH	368
4.27.1.9 REPLACE_GLYPHS	368
4.27.2 型定義詳解	369
4.27.2.1 MRealizedFontset	369
4.27.3 列挙型詳解	369
4.27.3.1 MDeviceType	369
4.27.3.2 glyph_type	369
4.27.3.3 glyph_category	369
4.27.4 関数詳解	370
4.27.4.1 mfont_init()	370

4.27.4.2 mfont_fini()	370
4.27.4.3 mface_init()	370
4.27.4.4 mface_fini()	370
4.27.4.5 mdraw_init()	370
4.27.4.6 mdraw_fini()	371
4.27.4.7 mfont_fontset_init()	371
4.27.4.8 mfont_fontset_fini()	371
4.27.4.9 minput_win_init()	371
4.27.4.10 minput_win_fini()	371
4.27.5 変数詳解	371
4.27.5.1 Mlatin	371
4.28 internal.h ファイル	372
4.28.1 マクロ定義詳解	375
4.28.1.1 _	375
4.28.1.2 MERROR	375
4.28.1.3 MERROR_GOTO	375
4.28.1.4 MWARNING	375
4.28.1.5 MFATAL	376
4.28.1.6 MFAILP	376
4.28.1.7 M_CHECK_CHAR	376
4.28.1.8 MEMORY_FULL	376
4.28.1.9 MTABLE_MALLOC	376
4.28.1.10 MTABLE_CALLOC	377
4.28.1.11 MTABLE_CALLOC_SAFE	377
4.28.1.12 MTABLE_REALLOC	377
4.28.1.13 MTABLE_ALLOCA	377
4.28.1.14 MSTRUCT_MALLOC	378
4.28.1.15 MSTRUCT_CALLOC	378
4.28.1.16 MSTRUCT_CALLOC_SAFE	378
4.28.1.17 USE_SAFE_ALLOCA	378
4.28.1.18 SAFE_ALLOCA	378
4.28.1.19 SAFE_FREE	379
4.28.1.20 MLIST_RESET	379
4.28.1.21 MLIST_INIT1	379
4.28.1.22 MLIST_APPEND1	379
4.28.1.23 MLIST_PREPEND1	380
4.28.1.24 MLIST_INSERT1	380
4.28.1.25 MLIST_DELETE1	380
4.28.1.26 MLIST_COPY1	381
4.28.1.27 MLIST_FREE1	381

4.28.1.28 M17N_OBJECT	381
4.28.1.29 M17N_OBJECT_REF	382
4.28.1.30 M17N_OBJECT_REF_NTICES	382
4.28.1.31 M17N_OBJECT_UNREF	382
4.28.1.32 M17N_OBJECT_ADD_ARRAY	383
4.28.1.33 M17N_OBJECT_REGISTER	383
4.28.1.34 M17N_OBJECT_UNREGISTER	383
4.28.1.35 M_CHECK_POS	384
4.28.1.36 M_CHECK_POS_X	384
4.28.1.37 M_CHECK_RANGE	384
4.28.1.38 M_CHECK_RANGE_X	384
4.28.1.39 M_CHECK_POS_NCHARS	385
4.28.1.40 MTEXT_READ_ONLY_P	385
4.28.1.41 M_CHECK_READONLY	385
4.28.1.42 mtext_nchars	385
4.28.1.43 mtext_nbytes	385
4.28.1.44 mtext_allocated	386
4.28.1.45 mtext_reset	386
4.28.1.46 MDEBUG_FLAG	386
4.28.1.47 MDEBUG_PRINT0	386
4.28.1.48 MDEBUG_PRINT	386
4.28.1.49 MDEBUG_PRINT1	386
4.28.1.50 MDEBUG_PRINT2	387
4.28.1.51 MDEBUG_PRINT3	387
4.28.1.52 MDEBUG_PRINT4	387
4.28.1.53 MDEBUG_PRINT5	387
4.28.1.54 MDEBUG_DUMP	387
4.28.1.55 MDEBUG_PUSH_TIME	388
4.28.1.56 MDEBUG_POP_TIME	388
4.28.1.57 MDEBUG_PRINT_TIME	388
4.28.1.58 SWAP_16	388
4.28.1.59 SWAP_32	388
4.28.2 列挙型詳解	389
4.28.2.1 MTextCoverage	389
4.28.2.2 MDebugFlag	389
4.28.3 関数詳解	389
4.28.3.1 mdebug_add_object_array()	389
4.28.3.2 mdebug_register_object()	390
4.28.3.3 mdebug_unregister_object()	390
4.28.3.4 mdebug_push_time()	390

4.28.3.5 mdebug_pop_time()	390
4.28.3.6 mdebug_print_time()	390
4.28.3.7 msymbol_init()	390
4.28.3.8 msymbol_fini()	390
4.28.3.9 mplist_init()	391
4.28.3.10 mplist_fini()	391
4.28.3.11 mtext_init()	391
4.28.3.12 mtext_fini()	391
4.28.3.13 mtext_prop_init()	391
4.28.3.14 mtext_prop_fini()	391
4.28.3.15 mchartable_init()	391
4.28.3.16 mchartable_fini()	392
4.28.3.17 mcharset_init()	392
4.28.3.18 mcharset_fini()	392
4.28.3.19 mcoding_init()	392
4.28.3.20 mcoding_fini()	392
4.28.3.21 mdatabase_init()	392
4.28.3.22 mdatabase_fini()	392
4.28.3.23 mchar_init()	393
4.28.3.24 mchar_fini()	393
4.28.3.25 mlang_init()	393
4.28.3.26 mlang_fini()	393
4.28.3.27 mlocale_init()	393
4.28.3.28 mlocale_fini()	393
4.28.3.29 minput_init()	393
4.28.3.30 minput_fini()	394
4.28.4 変数詳解	394
4.28.4.1 m17n_core_initialized	394
4.28.4.2 m17n_shell_initialized	394
4.28.4.3 m17n_gui_initialized	394
4.28.4.4 mdebug_flags	394
4.28.4.5 mdebug_output	394
4.29 language.c ファイル	394
4.29.1 関数詳解	395
4.29.1.1 mlanguage_name()	395
4.30 language.h ファイル	396
4.30.1 関数詳解	396
4.30.1.1 mscript_char_list()	396
4.30.1.2 mscript_otf_tag()	396
4.30.1.3 mscript_from_otf_tag()	396

4.31 locale.c ファイル	397
4.32 m17n-config.txt ファイル	397
4.33 m17n-core.c ファイル	397
4.34 m17n-core.h ファイル	398
4.34.1 マクロ定義詳解	404
4.34.1.1 M17N_BEGIN_HEADER	405
4.34.1.2 M17N_END_HEADER	405
4.34.2 変数詳解	405
4.34.2.1 Minteger	405
4.34.2.2 Msoft_dotted	405
4.34.2.3 Mcase_mapping	405
4.35 m17n-db.txt ファイル	405
4.36 m17n-flt.c ファイル	405
4.37 m17n-flt.h ファイル	406
4.37.1 変数詳解	407
4.37.1.1 mflt_font_id	407
4.37.1.2 mflt_iterate_off_feature	407
4.38 m17n-gd.c ファイル	408
4.39 m17n-gui.c ファイル	408
4.40 m17n-gui.h ファイル	408
4.40.1 型定義詳解	414
4.40.1.1 MFontset	414
4.40.2 関数詳解	414
4.40.2.1 mdebug_dump_font()	414
4.40.2.2 mdebug_dump_fontset()	415
4.40.3 変数詳解	415
4.40.3.1 Mfreetype	415
4.40.3.2 Mxft	415
4.41 m17n-misc.h ファイル	415
4.41.1 関数詳解	417
4.41.1.1 mdebug_dump_plist()	417
4.41.1.2 mdebug_dump_chartab()	417
4.42 m17n-X.c ファイル	418
4.42.1 関数詳解	418
4.42.1.1 device_open()	418
4.43 m17n-X.h ファイル	418
4.43.1 変数詳解	419
4.43.1.1 minput_xim_driver	419
4.44 m17n.c ファイル	419
4.45 m17n.h ファイル	419

4.45.1 関数詳解	426
4.45.1.1 mlanguage_name()	426
4.45.2 変数詳解	426
4.45.2.1 Miso639_2	426
4.46 mainpage.txt ファイル	426
4.47 mlocale.h ファイル	426
4.47.1 変数詳解	427
4.47.1.1 mlocale_collate	427
4.47.1.2 mlocale_ctype	427
4.47.1.3 mlocale_messages	427
4.47.1.4 mlocale_time	427
4.48 mtext-lbrk.c ファイル	427
4.49 mtext-wseg.c ファイル	427
4.50 mtext.c ファイル	427
4.51 mtext.h ファイル	430
4.51.1 マクロ定義詳解	430
4.51.1.1 POS_CHAR_TO_BYTE	430
4.51.1.2 POS_BYTE_TO_CHAR	431
4.51.1.3 MTEXT_DATA	431
4.51.1.4 MTEXT_CAT_ASCII	431
4.51.2 関数詳解	431
4.51.2.1 mtext_char_to_byte()	431
4.51.2.2 mtext_byte_to_char()	431
4.51.2.3 mtext_enlarge()	432
4.51.2.4 mtext_takein()	432
4.51.2.5 mtext_cat_data()	432
4.51.2.6 mtext_from_data()	432
4.51.2.7 mtext_adjust_format()	432
4.51.2.8 mtext_bol()	432
4.51.2.9 mtext_eol()	433
4.51.2.10 mtext_wseg_fini()	433
4.51.2.11 mtext_word_segment()	433
4.52 plist.c ファイル	433
4.52.1 関数詳解	434
4.52.1.1 mdebug_dump_plist()	434
4.53 plist.h ファイル	435
4.53.1 マクロ定義詳解	436
4.53.1.1 MPLIST_KEY	436
4.53.1.2 MPLIST_VAL	436
4.53.1.3 MPLIST_FUNC	436

4.53.1.4 MPLIST_NEXT	437
4.53.1.5 MPLIST_TAIL_P	437
4.53.1.6 MPLIST_SYMBOL_P	437
4.53.1.7 MPLIST_STRING_P	437
4.53.1.8 MPLIST_MTEXT_P	437
4.53.1.9 MPLIST_INTEGER_P	437
4.53.1.10 MPLIST_PLIST_P	437
4.53.1.11 MPLIST_NESTED_P	438
4.53.1.12 MPLIST_SET_NESTED_P	438
4.53.1.13 MPLIST_VAL_FUNC_P	438
4.53.1.14 MPLIST_SET_VAL_FUNC_P	438
4.53.1.15 MPLIST_SYMBOL	438
4.53.1.16 MPLIST_STRING	438
4.53.1.17 MPLIST_MTEXT	438
4.53.1.18 MPLIST_INTEGER	439
4.53.1.19 MPLIST_PLIST	439
4.53.1.20 MPLIST_FIND	439
4.53.1.21 MPLIST_DO	439
4.53.1.22 MPLIST_LENGTH	439
4.53.1.23 MPLIST_ADD_PLIST	440
4.53.1.24 MPLIST_PUSH_PLIST	440
4.53.1.25 MPLIST_PUT_PLIST	440
4.53.2 関数詳解	440
4.53.2.1 mplist_from_file()	440
4.53.2.2 mplist_from_plist()	440
4.53.2.3 mplist_from_alist()	441
4.53.2.4 mplist_from_string()	441
4.53.2.5 mplist_serialize()	441
4.53.2.6 mplist_conc()	441
4.53.2.7 mplist_pop_unref()	441
4.53.2.8 mplist_assq()	441
4.53.3 変数詳解	441
4.53.3.1 hex_mnemonic	442
4.53.3.2 escape_mnemonic	442
4.54 symbol.c ファイル	442
4.55 symbol.h ファイル	443
4.55.1 マクロ定義詳解	444
4.55.1.1 MSYMBOL_NAME	444
4.55.1.2 MSYMBOL_NAMELEN	444
4.55.2 関数詳解	444

4.55.2.1	<code>msymbol_free_table()</code>	444
4.55.2.2	<code>msymbol_with_len()</code>	445
4.55.2.3	<code>msymbol_list()</code>	445
4.55.2.4	<code>msymbol_canonicalize()</code>	445
4.55.3	変数詳解	445
4.55.3.1	<code>msymbol_serializer</code>	445
4.55.3.2	<code>msymbol_deserializer</code>	445
4.56	<code>textprop.c</code> ファイル	446
4.57	<code>textprop.h</code> ファイル	447
4.57.1	マクロ定義詳解	447
4.57.1.1	<code>MTEXTPROP_START</code>	447
4.57.1.2	<code>MTEXTPROP_END</code>	447
4.57.1.3	<code>MTEXTPROP_KEY</code>	447
4.57.1.4	<code>MTEXTPROP_VAL</code>	448
4.57.2	関数詳解	448
4.57.2.1	<code>mtext_copy_plist()</code>	448
4.57.2.2	<code>mtext_free_plist()</code>	448
4.57.2.3	<code>mtext_adjust_plist_for_delete()</code>	448
4.57.2.4	<code>mtext_adjust_plist_for_insert()</code>	448
4.57.2.5	<code>mtext_adjust_plist_for_change()</code>	449
4.57.2.6	<code>dump_textplist()</code>	449
Index		451
Index		451

Chapter 1

m17n ライブラリ

1.1 m17n ライブラリとは？

m17n ライブラリ は C 言語用の多言語文書処理ライブラリです。

- 自由公開ソフトウェアです。
- GNU/Linux と Unix のアプリケーションやライブラリから利用できます。
- アプリケーションやライブラリのさまざまな側面で、多言語化を実現します。

"m17n" とは "multilingualization" の省略形です。

m17n ライブラリは多言語を扱うため、以下の機能を提供します。

- **M-text**: 多言語テキスト用のデータ構造。基本的には文字列であるが、テキストプロパティと呼ばれる属性が付いており、C の文字列の代わりになるよう設計されている。m17n ライブラリで最も重要なオブジェクト。
- **M-text** を作ったり取り扱ったりするための関数。
- **M-text** と既存のフォーマットでコード化された文字列との間の変換を行う関数。
- 巨大な文字空間。Unicode 文字すべてとそれ以上の数の非 Unicode 文字を含むことができる。
- 文字テーブル: 文字毎の情報を効率的に保持するデータ構造。
- **M-text** をウィンドウシステム上で入力／表示する関数。

1.2 利用方法

<m17n.h> をプログラムに include し、-lm17n で m17n ライブラリ とリンクしてください。はじめにを参照。

1.3 外部ライブラリ／データ

m17n ライブラリは以下の外部ライブラリを利用しています。必須ではありませんが、m17n ライブラリの幾つかの関数はこれらに依存しています。

- m17n-db – <http://download.savannah.nongnu.org/releases/m17n/m17n-db-1.8.3.tar.gz>
m17n ライブラリに種々の情報を提供します。
- libxml2 – <http://xmlsoft.org/>
関数 `mtext_serialize()` と `mtext_deserialize()` が使います。libxml2 が利用できない時には、これらの関数は NULL を返します。
- fribidi – <http://fribidi.sourceforge.net/>
BIDI 処理に使います。利用できない時は、m17n ライブラリの表示エンジンは Arabic や Hebrew などのスクリプトを正しく処理できません。
- freetype – <http://www.freetype.org/>
ローカルフォントの処理に使います。
- fontconfig – <http://www.fontconfig.org/>
Xft と共に、ローカルフォントの検索に使います。
- xft – <http://freedesktop.org/Software/Xft>
fontconfig と共に X サーバの XRender 拡張を利用してテキストをローカルフォントで表示するために使います。
- GD テキストをローカルフォントで bitmap/pixmap 上に表示するのに使います。
- libotf – <http://www.m17n.org/libotf/>
freetype と共に OpenType フォントの処理に使います。
- anthy – <http://anthy.sourceforge.jp/>
日本語入力メソッド ja-anthy.mim が使います。
- wordcut – <http://thaiwordseg.sourceforge.net/>
プログラム例 example/linebreak.c 中でタイ語の語の境界を見つけるために使っています。

1.4 連絡先:

独立行政法人 産業技術総合研究所
情報技術研究部門
グローバル IT セキュリティグループ

Web: <https://savannah.nongnu.org/projects/m17n/>

バグレポート: <https://savannah.nongnu.org/bugs/?group=m17n>

メイリングリスト: <http://lists.nongnu.org/mailman/listinfo/m17n-list>

1.5 謝辞

Special thanks to:

- Dimitri van Heesch doxygen@gmail.com
Author of Doxygen <https://www.doxygen.nl/>. Without this tool, it would have been impossible to create this documentation.
- Information-technology Promotion Agency (IPA), Japan
Writing this documentation was partially funded by Information-technology Promotion Agency (IPA) <https://www.ipa.go.jp/en/index.html> in fiscal year 2001.

Chapter 2

モジュール詳解

2.1 はじめに

m17n ライブラリ イントロダクション.

マクロ定義

- `#define M17NLIB_MAJOR_VERSION`
- `#define M17NLIB_MINOR_VERSION`
- `#define M17NLIB_PATCH_LEVEL`
- `#define M17NLIB_VERSION_NAME`
- `#define M17N_INIT()`
m17n ライブラリを初期化する.
- `#define M17N_FINI()`
m17n ライブラリを終了する.

列挙型

- `enum M17NStatus {`
 `M17N_NOT_INITIALIZED ,`
 `M17N_CORE_INITIALIZED ,`
 `M17N_SHELL_INITIALIZED ,`
 `M17N_GUI_INITIALIZED }`
m17n ライブラリの状態を示す列挙型.

関数

- `enum M17NStatus m17n_status (void)`
m17n ライブラリのどの部分が初期化されたか報告する.

構築: Doxygen

2.1.1 詳解

m17n ライブラリ イントロダクション.

API のレベル

m17n ライブラリの API は以下の 4 種に分類されている。

1. コア API

M-text を扱うための基本的なモジュールを提供する。この API を利用するためには、アプリケーションプログラムは `<m17n-core.h>` を `include` し、`-lm17n-core` でリンクされなくてはならない。

2. シェル API

文字プロパティ、文字集合操作、コード変換等のためのモジュールを提供する。これらのモジュールは、データベースから必要に応じて多様なデータをロードする。この API を利用するためには、アプリケーションプログラムは `<m17n.h>` を `include` し、`-lm17n-core -lm17n` でリンクされなくてはならない。

この API を使用すれば、コア API も自動的に使用可能となる。

3. FLT API

文字列表示に [フォントレイアウトテーブル](#) を用いるモジュールを提供する。この API を利用するためには、アプリケーションプログラムは `<m17n.h>` を `include` し、`-lm17n-core -lm17n-flt` でリンクされなくてはならない。

この API を使用すれば、コア API も自動的に使用可能となる。

4. GUI API

グラフィックデバイス上で M-text を表示したり入力したりするための GUI モジュールを提供する。この API 自体はグラフィックデバイスとは独立であるが、多くの関数は特定のグラフィックデバイス用に作成された [MFrame](#) を引数に取る。現時点でサポートされているグラフィックデバイスは、ヌルデバイス、X ウィンドウシステム、および GD ライブラリのイメージデータ (`gdImagePtr`) である。ヌルデバイスのフレーム上では表示も入力もできない。ただし `mdraw-glyph-list()` などの関数は使用可能である。

X ウィンドウシステムのフレーム上ではすべての GUI API が使用できる。

GD ライブラリのフレーム上では、描画用の API はすべて使用できるが、入力にはできない。

この API を使用するためには、アプリケーションプログラムは `<m17n-gui.h>` を `include` し、`-lm17n-core -lm17n -lm17n-gui` でリンクされなくてはならない。

この API を使用すれば、コア API、シェル API、および FLT API も自動的に使用可能となる。

5. その他の API

エラー処理、デバッグ用のその他の関数を提供する。この API はそれだけでは使用できず、上記の他の API と共に使う。利用するためには、上記のいずれかの `include` ファイルに加えて、`<m17n-misc.h>` を `include` しなくてはならない。

[m17n-config\(1\)](#) 節も参照。

環境変数

m17n ライブラリは以下の環境変数を参照する。

- M17NDIR

m17n データベースのデータを格納したディレクトリの名前。詳細は [データベース](#) 参照。

- MDEBUG_XXX

"MDEBUG_" で始まる名前を持つ環境変数はデバッグ情報の出力を制御する。詳細は [デバッグサポート](#) 参照。

API の命名規則

m17n ライブラリは、関数、変数、マクロ、型を export する。それらは 'm' または 'M' のあとにオブジェクト名 ("symbol", "plist" など) またはモジュール名 (draw, input など) を続けたものである。M-text オブジェクトの名前は "mmtxt" ではなくて "mtext" で始まることに注意。

- 関数 – mobject() または mobject_xxx()
'm' のあとに小文字でオブジェクト名が続く。単語間は ':' で区切られる。たとえば, msymbol(), mtext_ref_char(), mdraw_text() など。
- シンボルでない変数 – mobject, または mobject_xxx
関数と同じ命名規則に従う。たとえば mface_large など。
- シンボル変数 – Mname
MSymbol 型変数は、'M' の後に名前が続く。単語間は ':' で区切られる。たとえば Mlanguage (名前は "language"), Miso_2022 (名前は "iso-2022") など。
- マクロ – MOBJECT_XXX
'M' の後に大文字でオブジェクト名が続く。単語間は ':' で区切られる。
- タイプ – MObject または MObjectXxx
'M' の後に大文字で始まるオブジェクト名が続く。単語は連続して書かれ、':' は用いられない。たとえば MConverter, MInputDriver など。

2.1.2 マクロ定義詳解

2.1.2.1 M17NLIB_MAJOR_VERSION

```
#define M17NLIB_MAJOR_VERSION
```

マクロ [M17NLIB_MAJOR_VERSION](#) は m17n ライブラリのメジャーバージョン番号を与える。

2.1.2.2 M17NLIB_MINOR_VERSION

```
#define M17NLIB_MINOR_VERSION
```

マクロ [M17NLIB_MINOR_VERSION](#) は m17n ライブラリのマイナーバージョン番号を与える。

2.1.2.3 M17NLIB_PATCH_LEVEL

```
#define M17NLIB_PATCH_LEVEL
```

マクロ [M17NLIB_PATCH_LEVEL](#) は m17n ライブラリのパッチレベル番号を与える。

2.1.2.4 M17NLIB_VERSION_NAME

```
#define M17NLIB_VERSION_NAME
```

マクロ `M17NLIB_VERSION_NAME` は m17n ライブラリのバージョン名を文字列として与える。

2.1.2.5 M17N_INIT

```
#define M17N_INIT( )
```

m17n ライブラリを初期化する。

マクロ `M17N_INIT()` は m17n ライブラリを初期化する。m17n の関数を利用する前に、このマクロをまず呼ばなくてはならない。

このマクロを複数回呼んでも安全であるが、その場合メモリを解放するためにマクロ `M17N_FINI()` を同じ回数呼ぶ必要がある。

外部変数 `merror_code` は、初期化が成功すれば 0 に、そうでなければ -1 に設定される。

参照:

`M17N_FINI()`, `m17n_status()`

2.1.2.6 M17N_FINI

```
#define M17N_FINI( )
```

m17n ライブラリを終了する。

マクロ `M17N_FINI()` は m17n ライブラリを終了する。m17n ライブラリが使った全てのメモリ領域は解放される。一度このマクロが呼ばれたら、マクロ `M17N_INIT()` が再度呼ばれるまで m17n 関数は使うべきでない。

マクロ `M17N_INIT()` が N 回呼ばれていた場合には、このマクロが N 回呼ばれて初めてメモリが解放される。

参照:

`M17N_INIT()`, `m17n_status()`

2.1.3 列挙型詳解

2.1.3.1 M17NStatus

```
enum M17NStatus
```

m17n ライブラリの状態を示す列挙型。

列挙型 `M17NStatus` は関数 `m17n_status()` の戻り値として用いられる。

列挙値

M17N_NOT_INITIALIZED	
M17N_CORE_INITIALIZED	
M17N_SHELL_INITIALIZED	
M17N_GUI_INITIALIZED	

2.1.4 関数詳解

2.1.4.1 m17n_status()

```
enum M17NStatus m17n_status (  
    void )
```

m17n ライブラリのどの部分が初期化されたか報告する。

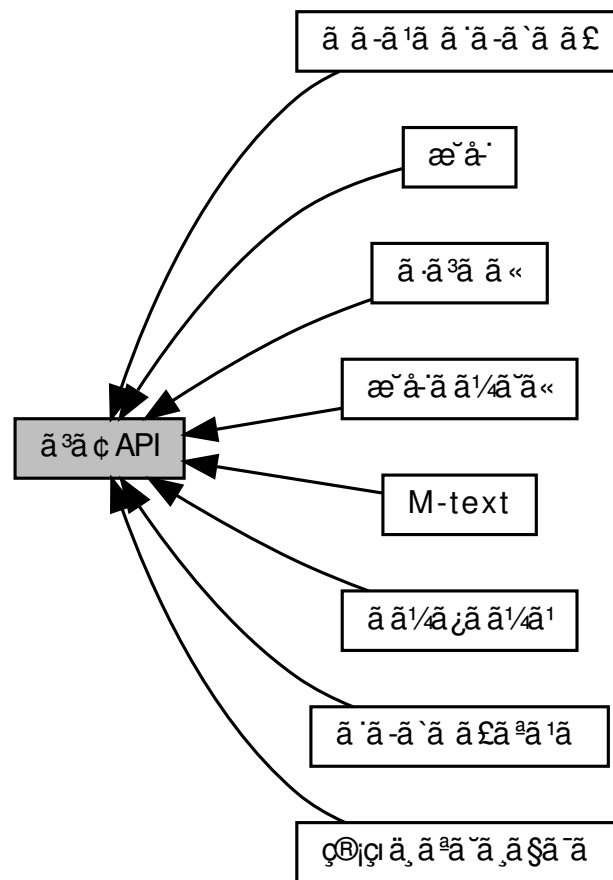
関数 [m17n_status\(\)](#) は m17n ライブラリのどの部分が初期化されたかに応じて、以下の値のいずれかを返す。

[M17N_NOT_INITIALIZED](#), [M17N_CORE_INITIALIZED](#), [M17N_SHELL_INITIALIZED](#), [M17N_GUI_INITIALIZED](#)

2.2 コア API

libm17n-core.so が提供する API

コア API 連携図



モジュール

- **管理下オブジェクト**
参照回数で管理されるオブジェクト
- **シンボル**
シンボルオブジェクトとそれに関する API.
- **プロパティリスト**
プロパティリストオブジェクトとそれに関する API.
- **文字**
文字オブジェクトとそれに関する API.
- **文字テーブル**
文字テーブルとそれに関する API.
- **M-text**
M-text オブジェクトとそれに関する API.
- **テキストプロパティ**
テキストプロパティを操作するための関数.
- **データベース**
m17n データベースにとそれに関する API.

マクロ定義

- `#define M17N_FUNC(func) ((M17NFunc) (func))`
汎関数型へのラップ。

型定義

- `typedef void(* M17NFunc) (void)`
汎関数型。

2.2.1 詳解

libm17n-core.so が提供する API

2.2.2 マクロ定義詳解

2.2.2.1 M17N_FUNC

```
#define M17N_FUNC(  
    func ) ((M17NFunc) (func))
```

汎関数型へのラップ。

マクロ `M17N_FUNC()` は関数を `M17NFunc` 型へキャストする。

2.2.3 型定義詳解

2.2.3.1 M17NFunc

```
typedef void(* M17NFunc) (void)
```

汎関数型。

`M17NFunc` は汎関数型であり、関数ポインタを `MSymbol` プロパティや `MPList` の値として設定する際用いる。

参照:

[msymbol_put_func\(\)](#), [msymbol_get_func\(\)](#), [mplist_put_func\(\)](#), [mplist_get_func\(\)](#).

2.3.2.1 m17n_object()

```
void* m17n_object (
    int size,
    void(*) (void *) freer )
```

@brief 管理下オブジェクトを割り当てる.

関数 `m17n_object()` は @b size バイトの新しい管理下オブジェクトを割り当て、その参照数を 1 とする。@b freer は参照数が 0 になった際にそのオブジェクトを解放するために用いられる関数である。@b freer が NULL ならば、オブジェクトは関数 `free()` によって解放される。

割り当てられたオブジェクト冒頭のバイトは、`#M17NObjectHead` が占める。この領域は `m17n` ライブラリが使用するので、アプリケーションプログラムは触れてはならない。

@par 戻り値:
この関数は新しく割り当てられたオブジェクトを返す。

@par エラー:
この関数は失敗しない。

例:

```
typedef struct
{
    M17NObjectHead head;
    int mem1;
    char *mem2;
} MYStruct;
void
my_freer (void *obj)
{
    free ((MYStruct *) obj->mem2);
    free (obj);
}
void
my_func (MText *mt, MSymbol key, int num, char *str)
{
    MYStruct *st = m17n_object (sizeof (MYStruct), my_freer);
    st->mem1 = num;
    st->mem2 = strdup (str);
    /* KEY must be a managing key. */
    mtext.put_prop (mt, 0, mtext.len (mt), key, st);
    /* This sets the reference count of ST back to 1. */
    m17n_object_unref (st);
}
```

2.3.2.2 m17n_object_ref()

```
int m17n_object_ref (
    void * object )
```

管理下オブジェクトの参照数を 1 増やす。

関数 `m17n_object_ref()` は `object` で指される管理下オブジェクトの参照数を 1 増やす。

戻り値:
この関数は、増やした参照数が 16 ビットの符号無し整数値 (すなわち 0x10000 未満) におさまれば、それを返す。そうでなければ -1 を返す。

エラー:
この関数は失敗しない。

2.3.2.3 m17n_object_unref()

```
int m17n_object_unref (
    void * object )
```

管理下オブジェクトの参照数を 1 減らす。

関数 `m17n_object_unref()` は `object` で指される管理下オブジェクトの参照数を 1 減らす。参照数が 0 になれば、オブジェクトは解放関数によって解放される。

戻り値:

この関数は、減らした参照数が 16 ビットの符号無し整数値 (すなわち 0x10000 未満) におさまれば、それを返す。そうでなければ -1 を返す。つまり、0 が返って来た場合は `object` は解放されている。

エラー:

この関数は失敗しない。

2.4 シンボル

シンボルオブジェクトとそれに関する API.

シンボル 連携図



関数

- `MSymbol msymbol` (`const char *name`)
シンボルを得る.
- `MSymbol msymbol_as_managing_key` (`const char *name`)
管理キーを作る.
- `int msymbol_is_managing_key` (`MSymbol symbol`)
- `MSymbol msymbol_exist` (`const char *name`)
指定された名前を持つシンボルを探す.
- `char * msymbol_name` (`MSymbol symbol`)
シンボルの名前を得る.
- `int msymbol_put` (`MSymbol symbol`, `MSymbol key`, `void *val`)
シンボルプロパティに値を設定する.
- `void * msymbol_get` (`MSymbol symbol`, `MSymbol key`)
シンボルプロパティの値を得る.
- `int msymbol_put_func` (`MSymbol symbol`, `MSymbol key`, `M17NFunc func`)
シンボルプロパティの値 (関数ポインタ) を設定する.
- `M17NFunc msymbol_get_func` (`MSymbol symbol`, `MSymbol key`)
シンボルプロパティの値 (関数ポインタ) を得る.

変数

- [MSymbol Mnil](#)
"nil" を名前として持つシンボル.
- [MSymbol Mt](#)
"t" を名前として持つシンボル.
- [MSymbol Mstring](#)
"string" を名前として持つシンボル.
- [MSymbol Msymbol](#)
"symbol" を名前として持つシンボル.

2.4.1 詳解

シンボルオブジェクトとそれに関する API.

m17n ライブラリは一意に決まる識別子としてシンボル と呼ぶオブジェクトを用いる。シンボルは X ライブラリのアトムと似ているが、0 個以上の シンボルプロパティ を持つことができる。シンボルプロパティは キー と 値 からなる。キーはそれ自体シンボルであり、値は (void *) 型にキャストできるものなら何でもよい。「シンボル S が持つシンボルプロパティのうちキーが K のもの」を簡単に「S の K プロパティ」と呼ぶことがある。

シンボルの用途は主に以下の 3 通りである。

- シンボルプロパティおよび他のプロパティのキーを表す。
- 文字セット、コード系、フォントセットなどの各種オブジェクトを表す。
- m17n ライブラリ関数の引数となり、関数の挙動を制御する。

管理キー と呼ばれる特別なシンボルがあり、管理キーをキーとして持つプロパティの値は 管理下オブジェクト でなくてはならない。詳細は [管理下オブジェクト](#) 参照。

2.4.2 関数詳解

2.4.2.1 msymbol()

```
MSymbol msymbol (
    const char * name )
```

シンボルを得る。

関数 [msymbol\(\)](#) は name という名前を持つ正規化されたシンボルを返す。そのようなシンボルが存在しない場合には、生成する。生成されたシンボルは管理キーではない。

空白文字二つで始まるシンボルは m17n ライブラリ用であり、内部的にのみ用いられる。

戻り値:

この関数は見つけたか生成したかしたシンボルを返す。

エラー:

この関数は決して失敗しない。

参照:

[msymbol_as_managing_key\(\)](#), [msymbol_name\(\)](#), [msymbol_exist\(\)](#)

2.4.2.2 msymbol_as_managing_key()

```
MSymbol msymbol_as_managing_key (
    const char * name )
```

管理キーを作る。

関数 `msymbol_as_managing_key()` は名前 `name` を持つ新しく作られた管理キーを返す。すでに名前 `name` を持つシンボルがあれば、`Mnil` を返す。

空白文字二つで始まるシンボルは `m17n` ライブラリ用であり、内部的にのみ用いられる。

戻り値:

処理に成功すれば、この関数は生成したシンボルを返す。そうでなければ `Mnil` を返す。

エラー:

`MERROR_SYMBOL`

参照:

`msymbol()`, `msymbol_exist()`

2.4.2.3 msymbol_is_managing_key()

```
int msymbol_is_managing_key (
    MSymbol symbol )
```

2.4.2.4 msymbol_exist()

```
MSymbol msymbol_exist (
    const char * name )
```

指定された名前を持つシンボルを探す。

関数 `msymbol_exist()` は `name` という名前を持つシンボルを探す。

戻り値:

もしそのようなシンボルが存在するならばそのシンボルを返す。そうでなければ、定義済みシンボル `Mnil` を返す。

エラー:

この関数は決して失敗しない。

参照:

`msymbol_name()`, `msymbol()`

2.4.2.5 msymbol_name()

```
char* msymbol_name (
    MSymbol symbol )
```

シンボルの名前を得る.

関数 [msymbol_name\(\)](#) は指定されたシンボル `symbol` の名前を含む文字列へのポインタを返す。

エラー:

この関数は決して失敗しない。

参照:

[msymbol\(\)](#), [msymbol_exist\(\)](#)

2.4.2.6 msymbol_put()

```
int msymbol_put (
    MSymbol symbol,
    MSymbol key,
    void * val )
```

シンボルプロパティに値を設定する.

関数 [msymbol_put\(\)](#) は、シンボル `symbol` 中でキーが `key` であるシンボルプロパティの値を `val` に設定する。そのシンボルプロパティにすでに値があれば上書きする。 `symbol`, `key` とも `Mnil` であってはならない。

`key` が管理キーならば、`val` は管理下オブジェクトでなくてはならない。この場合、古い値の参照数は `NULL` でなければ 1 減らされ、`val` の参照数は 1 増やされる。

戻り値:

処理が成功すれば、[msymbol_put\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

`MERROR_SYMBOL`

参照:

[msymbol_get\(\)](#)

2.4.2.7 msymbol_get()

```
void* msymbol_get (
    MSymbol symbol,
    MSymbol key )
```

シンボルプロパティの値を得る。

関数 [msymbol_get\(\)](#) は、シンボル `symbol` が持つシンボルプロパティのうち、キーが `key` であるものを探す。もし該当するシンボルプロパティが存在すれば、その値を返す。そうでなければ `NULL` を返す。

戻り値:

エラーが検出された場合、[msymbol_get\(\)](#) は `NULL` を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

`MERROR_SYMBOL`

参照:

[msymbol_put\(\)](#)

2.4.2.8 msymbol_put_func()

```
int msymbol_put_func (
    MSymbol symbol,
    MSymbol key,
    M17NFunc func )
```

シンボルプロパティの値 (関数ポインタ) を設定する。

関数 [msymbol_put_func\(\)](#) は、関数 [msymbol_put\(\)](#) と同様に、シンボル `symbol` のキーが `key` であるシンボルプロパティの値を設定する。但し その値は関数ポインタ `func` である。

参照:

[msymbol_put\(\)](#), [M17N_FUNC\(\)](#)

2.4.2.9 msymbol_get_func()

```
M17NFunc msymbol_get_func (
    MSymbol symbol,
    MSymbol key )
```

シンボルプロパティの値 (関数ポインタ) を得る。

関数 [msymbol_get_func\(\)](#) は、関数 [msymbol_get\(\)](#) と同様に、シンボル `symbol` が持つシンボルプロパティのうち、キーが `key` であるものを得る。但し その値は関数ポインタである。

参照:

[msymbol_get\(\)](#)

2.4.3 変数詳解

2.4.3.1 Mnil

`MSymbol` `Mnil`

"nil" を名前として持つシンボル。

シンボル `Mnil` は "nil" という名前を持ち、一般に「偽」または「否定」を意味する。"int" に変換された場合、値は 0 である。 `Mnil` 自身はいかなるシンボルプロパティも持たない。

2.4.3.2 Mt

`MSymbol` `Mt`

"t" を名前として持つシンボル。

シンボル `Mt` は "t" という名前を持ち、一般に「真」または「肯定」を意味する。

2.4.3.3 Mstring

`MSymbol` `Mstring`

"string" を名前として持つシンボル。

シンボル `Mstring` は "string" という名前を持ち、関数 `mchar_define_property()` などの引数として用いられる。

2.4.3.4 Msymbol

`MSymbol` `Msymbol`

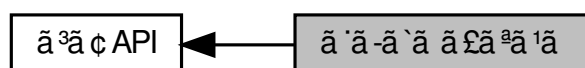
"symbol" を名前として持つシンボル。

定義済みシンボル `Msymbol` は "symbol" という名前を持ち、関数 `mchar_define_property()` などの引数として使われる。

2.5 プロパティリスト

プロパティリストオブジェクトとそれに関する API。

プロパティリスト 連携図



関数

- `MPlist * mplist_deserialize (MText *mt)`
M-text をデシリアライズしてプロパティリストを作る.
- `MPlist * mplist (void)`
プロパティリストオブジェクトを作る.
- `MPlist * mplist_copy (MPlist *plist)`
プロパティリストをコピーする.
- `MPlist * mplist_put (MPlist *plist, MSymbol key, void *val)`
プロパティリスト中のプロパティの値を設定する.
- `void * mplist_get (MPlist *plist, MSymbol key)`
プロパティリスト中のプロパティの値を得る.
- `MPlist * mplist_put_func (MPlist *plist, MSymbol key, M17NFunc func)`
プロパティリスト中のプロパティに関数ポインタである値を設定する.
- `M17NFunc mplist_get_func (MPlist *plist, MSymbol key)`
プロパティリストからプロパティの関数ポインタである値を得る.
- `MPlist * mplist_add (MPlist *plist, MSymbol key, void *val)`
プロパティリスト末尾にプロパティを追加する.
- `MPlist * mplist_push (MPlist *plist, MSymbol key, void *val)`
プロパティリストの先頭にプロパティを挿入する.
- `void * mplist_pop (MPlist *plist)`
プロパティリストの先頭からプロパティを削除する.
- `MPlist * mplist_find_by_key (MPlist *plist, MSymbol key)`
プロパティリスト中から指定のキーを持つプロパティを探す.
- `MPlist * mplist_find_by_value (MPlist *plist, void *val)`
プロパティリスト中から指定の値を持つプロパティを探す.
- `MPlist * mplist_next (MPlist *plist)`
プロパティリストの次の部分リストを返す.
- `MPlist * mplist_set (MPlist *plist, MSymbol key, void *val)`
プロパティリストの最初のプロパティを設定する.
- `int mplist_length (MPlist *plist)`
プロパティリストの長さを返す.
- `MSymbol mplist_key (MPlist *plist)`
プロパティリスト中の最初のプロパティのキーを返す.
- `void * mplist_value (MPlist *plist)`
プロパティリスト中の最初のプロパティの値を返す.

変数

- `MSymbol Minteger`
"integer" を名前として持つシンボル.
- `MSymbol Mplist`
"plist" を名前として持つシンボル.
- `MSymbol Mtext`
"mtext" を名前として持つシンボル.

2.5.1 詳解

プロパティリストオブジェクトとそれに関する API.

プロパティリスト (または `plist`) は 0 個以上のプロパティのリストである。プロパティは キー と 値 からなる。キーはシンボルであり、値は (`void *`) にキャストできるものならば何でも良い。

あるプロパティのキーが管理キー ならば、その 値 は 管理下 オブジェクト である。プロパティリスト自体も管理下オブジェクトである。

2.5.2 関数詳解

2.5.2.1 `mplist_deserialize()`

```
MPlist * mplist_deserialize (
    MText * mt )
```

M-text をデシリアライズしてプロパティリストを作る。

関数 `mplist_deserialize()` は M-text `mt` を解析してプロパティリストを返す。

`mt` のシンタックスは以下の通り。

`MT ::= '(' ELEMENT * ')'`

`ELEMENT ::= SYMBOL | INTEGER | M-TEXT | PLIST`

`SYMBOL ::=` アスキー文字列

`INTEGER ::= '-' ? ['0' | .. | '9']+ | '0x' ['0' | .. | '9' | 'A' | .. | 'F' | 'a' | .. | 'f']+`

`M-TEXT ::= "" character-sequence ""`

`ELEMENT` の各選択肢はキー: `Msymbol`, `Minteger`, `Mtext`, `Mplist` のいずれかを割り当てられている。

アスキー文字列内では、バックスラッシュ `()` がエスケープ文字として用いられる。たとえば `abc\ def` は 4 文字目が空白文字であり長さが 7 である持つ名前を持つシンボルを生成する。

2.5.2.2 `mplist()`

```
MPlist* mplist (
    void )
```

プロパティリストオブジェクトを作る。

関数 `mplist()` は長さ 0 のプロパティリストオブジェクトを新しく作って返す。

戻り値:

この関数は新しく作られたプロパティリストオブジェクトを返す。

エラー:

この関数は決して失敗しない。

2.5.2.3 mplist_copy()

```
MPlist* mplist_copy (
    MPlist * plist )
```

プロパティリストをコピーする。

関数 `mplist_copy()` はプロパティリスト `plist` をコピーする。コピーのすべての値はコピー元 `plist` の値と同じである。

戻り値:

この関数は新しく作られた、`plist` のコピーであるプロパティリストを返す。

エラー:

この関数は決して失敗しない。

2.5.2.4 mplist_put()

```
MPlist* mplist_put (
    MPlist * plist,
    MSymbol key,
    void * val )
```

プロパティリスト中のプロパティの値を設定する。

関数 `mplist_put()` はプロパティリスト `plist` を始めから探して、キーが `key` であるプロパティを見つける。見つければ、その値を `value` に変更する。見つからなければ、キーが `key` で値が `value` である新しいプロパティが `plist` の末尾に追加される。`key` と `val` に対する制限については、`mplist_add()` の説明を参照。

`key` が管理キーならば、`val` は管理下オブジェクトでなくてはならない。この場合、古い値の参照数は `NULL` でなければ 1 減らされ、`val` の参照数は 1 増やされる。

戻り値:

処理が成功すれば `mplist_put()` は変更されたか追加された要素から始まる `plist` の部分リストを返す。そうでなければ `NULL` を返す。

2.5.2.5 mplist_get()

```
void* mplist_get (
    MPlist * plist,
    MSymbol key )
```

プロパティリスト中のプロパティの値を得る。

関数 `mplist_get()` は、プロパティリスト `plist` を始めから探して、キーが `key` であるプロパティを見つける。見つければ、その値を `(void *)` 型で返す。見つからなければ `NULL` を返す。

`NULL` が返った際には二つの可能性がある: 上記のようにプロパティが見つからなかった場合と、プロパティが見つかり、その値が `NULL` である場合である。これらを区別する必要がある場合には関数 `mplist_find_by_key()` を使うこと。

参照:

[mplist_find_by_key\(\)](#)

2.5.2.6 mplist_put_func()

```
MPlist* mplist_put_func (
    MPlist * plist,
    MSymbol key,
    M17NFunc func )
```

プロパティリスト中のプロパティに関数ポインタである値を設定する。

関数 `mplist_put_func()` は関数 `mplist_put()` 同様、プロパティリスト `plist` 中でキーが `key` であるプロパティに値を設定する。但しその値は関数ポインタ `func` である。`key` は管理キーであってはならない。

参照:

[mplist_put\(\)](#), [M17N_FUNC\(\)](#)

2.5.2.7 mplist_get_func()

```
M17NFunc mplist_get_func (
    MPlist * plist,
    MSymbol key )
```

プロパティリストからプロパティの関数ポインタである値を得る。

関数 `mplist_get_func()` は関数 `mplist_get()` と同様に、プロパティリスト `plist` 中でキーが `key` であるプロパティの値、但し関数ポインタ、を得る。

参照:

[mplist_get\(\)](#)

2.5.2.8 mplist_add()

```
MPlist* mplist_add (
    MPlist * plist,
    MSymbol key,
    void * val )
```

プロパティリスト末尾にプロパティを追加する。

関数 `mplist_add()` は、プロパティリスト `plist` の末尾にキーが `key` で値が `val` であるプロパティを追加する。`key` は、`Mnil` 以外の任意のシンボルでよい。

`key` が管理キーならば、`val` は管理下オブジェクトでなくてはならない。この場合、`val` の参照数は 1 増やされる。

戻り値:

処理が成功すれば `mplist_add()` は追加された要素から始まる `plist` の部分リストを返す。そうでなければ `NULL` を返す。

2.5.2.9 mplist_push()

```
MPlist* mplist_push (
    MPlist * plist,
    MSymbol key,
    void * val )
```

プロパティリストの先頭にプロパティを挿入する。

関数 `mplist_push()` はプロパティリスト `plist` の先頭にキーが `key` で値が `val` であるオブジェクトを挿入する。

`key` が管理キーならば、`val` は管理下オブジェクトでなくてはならない。この場合、`val` の参照数は 1 増やされる。

戻り値:

処理が成功すればこの関数は `plist` を返し、そうでなければ `NULL` を返す。

2.5.2.10 mplist_pop()

```
void* mplist_pop (
    MPlist * plist )
```

プロパティリストの先頭からプロパティを削除する。

関数 `mplist_pop()` はプロパティリスト `plist` の先頭のプロパティを削除する。結果として、元の 2 番目のキーと値が先頭のキーと値になる。

戻り値:

処理に成功すれば、この関数は削除されたプロパティの値を返す。そうでなければ `NULL` を返す。

2.5.2.11 `mplist_find_by_key()`

```
MPLIST* mplist_find_by_key (
    MPLIST * plist,
    MSymbol key )
```

プロパティリストの中から指定のキーを持つプロパティを探す。

関数 `mplist_find_by_key()` はプロパティリスト `plist` を始めから探して、キーが `key` であるプロパティを見つける。見つければ、そのプロパティから始まる `plist` の部分リストを返す。そうでなければ `NULL` を返す。

`key` が `Mnil` ならば、`plist` の最後の要素から始まる部分リストを返す。

2.5.2.12 `mplist_find_by_value()`

```
MPLIST* mplist_find_by_value (
    MPLIST * plist,
    void * val )
```

プロパティリストの中から指定の値を持つプロパティを探す。

関数 `mplist_find_by_value()` はプロパティリスト `plist` を始めから探して、値が `val` であるプロパティを見つける。見つければ、そのプロパティから始まる `plist` の部分リストを返す。そうでなければ `NULL` を返す。

2.5.2.13 `mplist_next()`

```
MPLIST* mplist_next (
    MPLIST * plist )
```

プロパティリストの次の部分リストを返す。

関数 `mplist_next()` はプロパティリスト `plist` の 2 番目の要素から始まる部分リストへのポインタを返す。`plist` の長さが 0 ならば `NULL` を返す。

2.5.2.14 `mplist_set()`

```
MPLIST* mplist_set (
    MPLIST * plist,
    MSymbol key,
    void * val )
```

プロパティリストの最初のプロパティを設定する。

関数 `mplist_set()` はプロパティリスト `plist` の最初のプロパティのキーと値をそれぞれ `key` と `value` に設定する。`key` と `val` に対する制限については、`mplist_add()` の説明を参照。

戻り値:

処理に成功すれば `mplist_set()` は `plist` を返す。そうでなければ `NULL` を返す。

2.5.2.15 mplist_length()

```
int mplist_length (
    MPlist * plist )
```

プロパティリストの長さを返す。

関数 `mplist_length()` はプロパティリスト `plist` 中のプロパティの数を返す。

2.5.2.16 mplist_key()

```
MSymbol mplist_key (
    MPlist * plist )
```

プロパティリスト中の最初のプロパティのキーを返す。

関数 `mplist_key()` は、プロパティリスト `plist` 中の最初のプロパティのキーを返す。`plist` の長さが 0 ならば、`Mnil` を返す。

2.5.2.17 mplist_value()

```
void* mplist_value (
    MPlist * plist )
```

プロパティリスト中の最初のプロパティの値を返す。

関数 `mplist_value()` は、プロパティリスト `plist` 中の最初のプロパティの値を返す。`plist` の長さが 0 ならば、`Mnil` を返す。

2.5.3 変数詳解

2.5.3.1 Minteger

```
MSymbol Minteger
```

"integer" を名前として持つシンボル。

シンボル `Minteger` は "integer" という名前を持つ。キーが `Minteger` であるプロパティの値は整数値でなくてはならない。

2.5.3.2 Mplist

```
MSymbol Mplist
```

"plist" を名前として持つシンボル。

シンボル `Mplist` は "plist" という名前を持つ。これは管理キーである。キーが `Mplist` であるプロパティの値は `plist` でなくてはならない。

2.5.3.3 Mtext

MSymbol Mtext

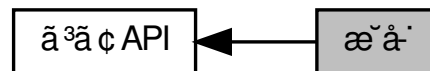
"mtext" を名前として持つシンボル.

シンボル Mtext は "mtext" という名前を持つ管理キーである。キーが Mtext であるプロパティの値は M-text でなくてはならない。

2.6 文字

文字オブジェクトとそれに関する API.

文字 連携図



マクロ定義

- `#define MCHAR_MAX`
文字コードの最大値.

関数

- `MSymbol mchar_define_property` (const char *name, MSymbol type)
文字プロパティを定義する.
- `void * mchar_get_prop` (int c, MSymbol key)
文字プロパティの値を得る.
- `int mchar_put_prop` (int c, MSymbol key, void *val)
文字プロパティの値を設定する.
- `MCharTable * mchar_get_prop_table` (MSymbol key, MSymbol *type)
文字プロパティの文字テーブルを得る.

変数: 文字プロパティのキー

これらのシンボルは文字プロパティのキーとして使われる。

- [MSymbol Mscript](#)
スクリプトを表わすキー.
- [MSymbol Mname](#)
名前を表わすキー.
- [MSymbol Mcategory](#)
一般カテゴリを表わすキー.
- [MSymbol Mcombining_class](#)
標準結合クラスを表わすキー.
- [MSymbol Mbidi_category](#)
双方向カテゴリを表わすキー.
- [MSymbol Msimple_case_folding](#)
対応する小文字一文字を表わすキー.
- [MSymbol Mcomplicated_case_folding](#)
対応する小文字の列を表わすキー.
- [MSymbol Mcased](#)
Case 処理に用いられる値のキー.
- [MSymbol Msoft_dotted](#)
Case 処理に用いられる値のキー.
- [MSymbol Mcase_mapping](#)
Case 処理に用いられる値のキー.
- [MSymbol Mblock](#)
スクリプトブロック名を表すキー.

2.6.1 詳解

文字オブジェクトとそれに関する API.

m17n ライブラリは文字を文字コード（整数）で表現する。最小の文字コードは 0 であり、最大の文字コードはマクロ [MCHAR_MAX](#) によって定義されている。::MCHAR_MAX は 0x3FFFFFF (22 ビット) 以上であることが保証されている。

0 から 0x10FFFF までの文字は、それと同じ値を持つ Unicode の文字に割り当てられている。

各文字は文字プロパティと呼ぶプロパティを 0 個以上持つことができる。文字プロパティはキーと値からなる。キーはシンボルであり、値は (void *) 型にキャストできるものなら何でもよい。「文字 C の文字プロパティのうちキーが K であるもの」を簡単に「文字 C の K プロパティ」と呼ぶことがある。

2.6.2 マクロ定義詳解

2.6.2.1 MCHAR_MAX

```
#define MCHAR_MAX
```

文字コードの最大値.

マクロ `MCHAR_MAX` は文字コードの最大値を表す。

2.6.3 関数詳解

2.6.3.1 mchar_define_property()

```
MSymbol mchar_define_property (
    const char * name,
    MSymbol type )
```

文字プロパティを定義する.

関数 `mchar_define_property()` は、`<Mchar_table, type, sym >` というタグを持ったデータベースを m17n 言語情報ベースから探す。

ここで `sym` は `name` という名前のシンボルである。 `type` は `::Mstring`, `Mtext`, `MSymbol`, `Minteger`, `Mplist` のいずれかでなければならない。

戻り値:

処理に成功すれば `mchar_define_property()` は `sym` を返す。 失敗した場合は `Mnil` を返す。

エラー:

`MERROR_DB`

参照:

`mchar_get_prop()`, `mchar_put_prop()`

2.6.3.2 mchar_get_prop()

```
void* mchar_get_prop (
    int c,
    MSymbol key )
```

文字プロパティの値を得る.

関数 `mchar_get_prop()` は、文字 `c` の文字プロパティのうちキーが `key` であるものを探す。

戻り値:

処理が成功すれば `mchar_get_prop()` は見つかったプロパティの値を返す。 失敗した場合は `NULL` を返す。

エラー:

`MERROR_SYMBOL`, `MERROR_DB`

参照:

`mchar_define_property()`, `mchar_put_prop()`

2.6.3.3 mchar_put_prop()

```
int mchar_put_prop (
    int c,
    MSymbol key,
    void * val )
```

文字プロパティの値を設定する。

関数 `mchar_put_prop()` は、文字 `c` の文字プロパティのうちキーが `key` であるものを探し、その値として `val` を設定する。

戻り値:

処理が成功すれば `mchar_put_prop()` は 0 を返す。失敗した場合は -1 を返す。

エラー:

`MERROR_SYMBOL, MERROR_DB`

参照:

`mchar_define_property()`, `mchar_get_prop()`

2.6.3.4 mchar_get_prop_table()

```
MCharTable* mchar_get_prop_table (
    MSymbol key,
    MSymbol * type )
```

文字プロパティの文字テーブルを得る。

関数 `mchar_get_prop_table()` は、キーが `key` である文字プロパティを含む文字テーブルを返す。もし `type` が `NULL` でなければ、`type` で指される場所にその文字のプロパティを格納する。文字プロパティの種類に関しては `mchar_define_property()` を見よ。

戻り値:

もし `key` が正当な文字プロパティのキーであれば、文字テーブルが返される。そうでない場合は `NULL` が返される。

2.6.4 変数詳解

2.6.4.1 Mscript

`MSymbol Mscript`

スクリプトを表わすキー。

シンボル `Mscript` は "script" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、この文字の属するスクリプトを表わすシンボルである。

スクリプトを表わすシンボルの名前は、Unicode Technical Report #24 にリストされているもののいずれかである。

2.6.4.2 Mname

`MSymbol Mname`

名前を表わすキー。

シンボル `Mname` は "name" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値はその文字の名前を表わす C の文字列である。

2.6.4.3 Mcategory

`MSymbol Mcategory`

一般カテゴリを表わすキー。

シンボル `Mcategory` は "category" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、対応する 一般カテゴリ を表わすシンボルである。

一般カテゴリを表わすシンボルの名前は、General Category の省略形として Unicode に定義されているものである。

2.6.4.4 Mcombining_class

`MSymbol Mcombining_class`

標準結合クラスを表わすキー。

シンボル `Mcombining_class` は "combining-class" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、対応する 標準結合クラス を表わす整数である。

標準結合クラスを表わす整数の意味は、Unicode に定義されているものと同じである。

2.6.4.5 Mbidi.category

MSymbol Mbidi.category

双方向カテゴリを表わすキー。

シンボル **Mbidi.category** は "bidi-category" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、対応する 双方向カテゴリ を表わすシンボルである。

双方向カテゴリを表わすシンボルの名前は、**Bidirectional Category** の型として **Unicode** に定義されているものである。

2.6.4.6 Msimple.case.folding

MSymbol Msimple.case.folding

対応する小文字一文字を表わすキー。

シンボル **Msimple.case.folding** は "simple-case-folding" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、対応する小文字一文字であり、大文字／小文字の区別を無視した文字列比較の際に使われる。

複雑な比較方法を必要とする文字であった場合（別の一文字と対応付けることによって比較できない場合）、このプロパティの値は 0xFFFF になる。この場合その文字は、**::Mcomplicated.case.folding** というキーの文字プロパティを持つ。

2.6.4.7 Mcomplicated.case.folding

MSymbol Mcomplicated.case.folding

対応する小文字の列を表わすキー。

シンボル **Mcomplicated.case.folding** は "complicated-case-folding" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は、対応する小文字列からなる **M-text** であり、大文字／小文字の区別を無視した文字列比較の際に使われる。

2.6.4.8 Mcased

MSymbol Mcased

Case 処理に用いられる値のキー。

シンボル **Mcased** は、"cased" という名前を持ち、文字プロパティのキーとして使われる。このプロパティの値は整数値 1, 2, 3 のいずれかであり、それぞれ "cased", "case-ignorable", その両方を意味する。詳細については、the Unicode Standard 5.0 (Section 3.13 Default Case Algorithm) 参照。

関数

- `MCharTable * mchartable (MSymbol key, void *default_value)`
新しい文字テーブルを作る.
- `int mchartable_min_char (MCharTable *table)`
- `int mchartable_max_char (MCharTable *table)`
- `void * mchartable_lookup (MCharTable *table, int c)`
文字テーブル中で文字に割り当てられた値を返す.
- `int mchartable_set (MCharTable *table, int c, void *val)`
文字テーブル中での文字の値を設定する.
- `int mchartable_set_range (MCharTable *table, int from, int to, void *val)`
指定範囲の文字に値を設定する.
- `void mchartable_range (MCharTable *table, int *from, int *to)`
値がデフォルトと異なる文字を探す.
- `int mchartable_map (MCharTable *table, void *ignore, void(*func)(int, int, void *, void *), void *func_arg)`
文字テーブル中の文字に対して指定の関数を呼ぶ.

変数

- `MSymbol Mchar_table`

2.7.1 詳解

文字テーブルとそれに関する API.

"char-table" という名前を持つシンボル.

m17n ライブラリが扱う文字の空間は広大であるため、文字毎の情報を単純な配列に格納しようとする、その配列は巨大になりすぎ、非実用的である。しかし通常必要となる文字についての情報は、ある特定の範囲の文字にのみ付いていることが多い。全文字に関して情報がある場合にも、連続した文字コードを持つ文字は同じ情報を持つことが多い。

このような傾向を利用して文字とその付加情報を効率的に格納するために、m17n ライブラリは文字テーブル (chartable) と呼ぶオブジェクトを用いる。文字テーブルは配列ではないが、アプリケーションプログラムは文字テーブルを配列の一種として扱うことができる。ある文字についての特定の情報は、その情報を持つ文字テーブルをその文字のコードで引くことで得られる。

文字テーブルは管理下オブジェクトである。

シンボル Mchar_table は名前 "char-table" を持つ。

2.7.2 型定義詳解

2.7.2.1 MCharTable

```
typedef struct MCharTable MCharTable
```

文字テーブルの型宣言.

MCharTable は 文字テーブル (chartable) オブジェクトの型である。内部構造はアプリケーションプログラムからは見えない。

2.7.3 関数詳解

2.7.3.1 mchartable()

```
MCharTable* mchartable (
    MSymbol key,
    void * default_value )
```

新しい文字テーブルを作る.

関数 **mchartable()** はキーが **key** で要素のデフォルト値が **default_value** である新しい文字テーブルを作る。もし **key** が管理キーであれば、このテーブルの要素は（デフォルト値を含めて）管理下オブジェクトか **NULL** のいずれかである。

戻り値:

処理が成功すれば **mchartable()** は作成された文字テーブルへのポインタを返す。失敗した場合は **NULL** を返し、外部変数 **merror_code** にエラーコードを設定する。

2.7.3.2 mchartable_min_char()

```
int mchartable_min_char (
    MCharTable * table )
```

2.7.3.3 mchartable_max_char()

```
int mchartable_max_char (
    MCharTable * table )
```

2.7.3.4 mchartable_lookup()

```
void* mchartable_lookup (
    MCharTable * table,
    int c )
```

文字テーブル中で文字に割り当てられた値を返す。

関数 `mchartable_lookup()` は文字テーブル `table` 中で文字 `c` に割り当てられた値を返す。`c` に対する明示的な値がなければ、`table` のデフォルト値を返す。`c` が妥当な文字でなければ、`mchartable_lookup()` は `NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_CHAR`

参照:

[mchartable_set\(\)](#)

2.7.3.5 mchartable_set()

```
int mchartable_set (
    MCharTable * table,
    int c,
    void * val )
```

文字テーブル中での文字の値を設定する。

関数 `mchartable_set()` は、文字テーブル `table` 中の文字 `c` に値 `val` を割り当てる。

戻り値:

処理が成功すれば、`mchartable_set()` は 0 を返す。そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_CHAR`

参照:

[mchartable_lookup\(\)](#), [mchartable_set_range\(\)](#)

2.7.3.6 mchartable_set_range()

```
int mchartable_set_range (
    MCharTable * table,
    int from,
    int to,
    void * val )
```

指定範囲の文字に値を設定する。

関数 [mchartable_set_range\(\)](#) は、文字テーブル `table` 中の `from` から `to` まで（両端を含む）の文字に、値として `val` を設定する。

戻り値:

処理が成功すれば [mchartable_set_range\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。`from` が `to` より大きいときには、[mchartable_set_range\(\)](#) は何もせず、エラーも起こさない。

エラー:

`MERROR_CHAR`

参照:

[mchartable_set\(\)](#)

2.7.3.7 mchartable_range()

```
void mchartable_range (
    MCharTable * table,
    int * from,
    int * to )
```

値がデフォルトと異なる文字を探す。

関数 [mchartable_range\(\)](#) は文字テーブル `table` 中で、`table` のデフォルト値以外の値を持つ最初と最後の文字を探し、それぞれを `from` と `to` に設定する。すべての文字が値としてデフォルト値をとっている場合には `from` と `to` を -1 に設定する。

2.7.3.8 mchartable_map()

```
int mchartable_map (
    MCharTable * table,
    void * ignore,
    void(*) (int, int, void *, void *) func,
    void * func_arg )
```

文字テーブル中の文字に対して指定の関数を呼ぶ。

関数 `mchartable_map()` は、文字テーブル `table` 中の文字に対して関数 `func` を呼ぶ。ただし `table` 中でも値が `ignore` である文字については関数呼び出しを行なわない。`ignore` と文字の値の比較は `==` で行なうので、文字列リテラルやポインタを使う際には注意を要する。

`mchartable_map()` は、一文字ごとに `func` を呼ぶのではなく、関数呼び出しの回数を最適化しようとする。すなわち、連続した文字が同じ値を持っていた場合には、その文字のまとまり全体について一度の関数呼び出ししか行なわない。

文字のまとまりの大きさにかかわらず、`func` は `from, to, val, arg` の 4 引数で呼ばれる。`from` と `to`（両端を含む）は `val` を値として持つ文字の範囲を示し、`arg` は `func_arg` そのものである。

戻り値:

この関数は常に 0 を返す。

2.7.4 変数詳解

2.7.4.1 Mchar_table

`MSymbol` Mchar_table

2.8 M-text

M-text オブジェクトとそれに関する API.

M-text 連携図



列挙型

- enum `MTextFormat` {
`MTEXT_FORMAT_US_ASCII` ,
`MTEXT_FORMAT_UTF_8` ,
`MTEXT_FORMAT_UTF_16LE` ,
`MTEXT_FORMAT_UTF_16BE` ,
`MTEXT_FORMAT_UTF_32LE` ,
`MTEXT_FORMAT_UTF_32BE` ,
`MTEXT_FORMAT_MAX` }
- M-text のフォーマットを指定する列挙型.
- enum `MTextLineBreakOption` {
`MTEXT_LBO_SP_CM` = 1 ,
`MTEXT_LBO_KOREAN_SP` = 2 ,
`MTEXT_LBO_AI_AS_ID` = 4 ,
`MTEXT_LBO_MAX` }

関数

- int `mtext_line_break` (`MText` *mt, int pos, int option, int *after)
- `MText` * `mtext` ()
 新しいM-text を割り当てる.
- `MText` * `mtext_from_data` (const void *data, int nitems, enum `MTextFormat` format)
 指定のデータを元に新しい M-text を割り当てる.
- void * `mtext_data` (`MText` *mt, enum `MTextFormat` *fmt, int *nunits, int *pos_idx, int *unit_idx)
- int `mtext_len` (`MText` *mt)
 M-text 中の文字の数.
- int `mtext_ref_char` (`MText` *mt, int pos)
 M-text 中の指定された位置の文字を返す.
- int `mtext_set_char` (`MText` *mt, int pos, int c)
 M-text に一文字を設定する.
- `MText` * `mtext_cat_char` (`MText` *mt, int c)
 M-text に一文字追加する.
- `MText` * `mtext_dup` (`MText` *mt)
 M-text のコピーを作る.
- `MText` * `mtext_cat` (`MText` *mt1, `MText` *mt2)
 2 個の M-text を連結する.
- `MText` * `mtext_ncat` (`MText` *mt1, `MText` *mt2, int n)
 M-text の一部を別の M-text に付加する.
- `MText` * `mtext_cpy` (`MText` *mt1, `MText` *mt2)
 M-text を別の M-text にコピーする.
- `MText` * `mtext_ncpy` (`MText` *mt1, `MText` *mt2, int n)
 M-text に含まれる最初の何文字かをコピーする.
- `MText` * `mtext_duplicate` (`MText` *mt, int from, int to)
 既存の M-text の一部から新しい M-text をつくる.
- `MText` * `mtext_copy` (`MText` *mt1, int pos, `MText` *mt2, int from, int to)
 M-text に指定範囲の文字をコピーする.
- int `mtext_del` (`MText` *mt, int from, int to)
 指定範囲の文字を破壊的に取り除く.
- int `mtext_ins` (`MText` *mt1, int pos, `MText` *mt2)

- M-text を別の M-text に挿入する.
- `int mtext_insert (MText *mt1, int pos, MText *mt2, int from, int to)`
M-text の一部を別の M-text に挿入する.
- `int mtext_ins_char (MText *mt, int pos, int c, int n)`
M-text に文字を挿入する.
- `int mtext_replace (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
M-text の一部を別の M-text の一部で置換する.
- `int mtext_character (MText *mt, int from, int to, int c)`
M-text 中で文字を探す.
- `int mtext_chr (MText *mt, int c)`
M-text 中で指定された文字が最初に現れる位置を返す.
- `int mtext_rchr (MText *mt, int c)`
M-text 中で指定された文字が最後に現れる位置を返す.
- `int mtext_cmp (MText *mt1, MText *mt2)`
二つの M-text を文字単位で比較する.
- `int mtext_ncmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を文字単位で比較する.
- `int mtext_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域同士を比較する.
- `int mtext_spn (MText *mt, MText *accept)`
ある集合の文字を M-text の中で探す.
- `int mtext_cspn (MText *mt, MText *reject)`
ある集合に属さない文字を M-text の中で探す.
- `int mtext_pbrk (MText *mt, MText *accept)`
ある集合に属す文字を M-text の中から探す.
- `MText * mtext_tok (MText *mt, MText *delim, int *pos)`
M-text 中のトークンを探す.
- `int mtext_text (MText *mt1, int pos, MText *mt2)`
M-text 中で別の M-text を探す.
- `int mtext_search (MText *mt1, int from, int to, MText *mt2)`
M-text 中の特定の領域で別の M-text を探す.
- `int mtext_casecmp (MText *mt1, MText *mt2)`
二つの M-text を大文字／小文字の区別を無視して比較する.
- `int mtext_ncasecmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を大文字／小文字の区別を無視して比較する.
- `int mtext_case_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域を、大文字／小文字の区別を無視して比較する.
- `int mtext_lowercase (MText *mt)`
M-text を小文字にする.
- `int mtext_titlecase (MText *mt)`
M-text をタイトルケースにする.
- `int mtext_uppercase (MText *mt)`
M-text を大文字にする.

変数

- `MSymbol Mlanguage`

変数: UTF-16 と UTF-32 のデフォルトのエンディアン

- enum `MTextFormat MTEXT_FORMAT_UTF_16`
値が `MTEXT_FORMAT_UTF_16LE` か `MTEXT_FORMAT_UTF_16BE` である変数
- const int `MTEXT_FORMAT_UTF_32`
値が `MTEXT_FORMAT_UTF_32LE` か `MTEXT_FORMAT_UTF_32BE` である変数

2.8.1 詳解

M-text オブジェクトとそれに関する API.

m17n ライブラリは、C-string (`char *` や `unsigned char *`) ではなく M-text と呼ぶオブジェクトでテキストを表現する。M-text は長さ 0 以上の文字列であり、種々の文字ソース（たとえば C-string、ファイル、文字コード等）から作成できる。

M-text には、C-string がない以下の特徴がある。

- M-text は非常に多くの種類の文字を、同時に、混在させて、同等に扱うことができる。Unicode の全ての文字はもちろん、より多くの文字までも扱うことができる。これは多言語テキストを扱う上では必須の機能である。
- M-text 内の各文字は、テキストプロパティ と呼ばれるプロパティを持ち、テキストプロパティによって、テキストの各部位に関する様々な情報を M-text 内に保持することが可能になる。そのため、それらの情報をアプリケーションプログラム内で統一的に扱うことが可能になる。また、M-text 自体が豊富な情報を持つため、アプリケーションプログラム中の各関数を簡素化することができる。

さらに m17n ライブラリは、C-string を操作するために提供される種々の関数と同等のものを M-text を操作するためにサポートしている。

2.8.2 列挙型詳解

2.8.2.1 MTextFormat

enum `MTextFormat`

M-text のフォーマットを指定する列挙型.

列挙型 `MTextFormat` は関数 `mtext_from_data()` の引数として用いられ、M-text を生成する元となるデータのフォーマットを指定する。

列挙値

<code>MTEXT_FORMAT_US_ASCII</code>	US-ASCII encoding
<code>MTEXT_FORMAT_UTF_8</code>	UTF-8 encoding
<code>MTEXT_FORMAT_UTF_16LE</code>	UTF-16LE encoding
<code>MTEXT_FORMAT_UTF_16BE</code>	UTF-16BE encoding
<code>MTEXT_FORMAT_UTF_32LE</code>	UTF-32LE encoding
<code>MTEXT_FORMAT_UTF_32BE</code>	UTF-32BE encoding

2.8.2.2 MTextLineBreakOption

enum [MTextLineBreakOption](#)

列挙値

MTEXT_LBO_SP_CM	
MTEXT_LBO_KOREAN_SP	
MTEXT_LBO_ALAS_ID	
MTEXT_LBO_MAX	

2.8.3 関数詳解

2.8.3.1 mtext_line_break()

```
int mtext_line_break (
    MText * mt,
    int pos,
    int option,
    int * after )
```

2.8.3.2 mtext()

[MText](#)* mtext ()

新しいM-text を割り当てる.

関数 [mtext\(\)](#) は、長さ 0 の新しい M-text を割り当て、それへのポインタを返す。割り当てられた M-text は、関数 [m17n_object_unref\(\)](#) によってユーザが明示的に行なわない限り、解放されない。

参照:

[m17n_object_unref\(\)](#)

2.8.3.3 mtext_from_data()

```
MText* mtext_from_data (
    const void * data,
    int nitems,
    enum MTextFormat format )
```

指定のデータを元に新しい M-text を割り当てる。

関数 `mtext_from_data()` は、要素数 `nitems` の配列 `data` で指定された文字列を持つ新しい M-text を割り当てる。`format` は `data` のフォーマットを示す。

`format` が `MTEXT_FORMAT_US_ASCII` か `MTEXT_FORMAT_UTF_8` ならば、`data` の内容は `unsigned char` 型であり、`nitems` はバイト単位で表されている。

`format` が `MTEXT_FORMAT_UTF_16LE` か `MTEXT_FORMAT_UTF_16BE` ならば、`data` の内容は `unsigned short` 型であり、`nitems` は `unsigned short` 単位である。

`format` が `MTEXT_FORMAT_UTF_32LE` か `MTEXT_FORMAT_UTF_32BE` ならば、`data` の内容は `unsigned` 型であり、`nitems` は `unsigned` 単位である。

割り当てられた M-text の文字列は変更できない。`data` の内容は M-text が有効な間に変更してはならない。

割り当てられた M-text は、関数 `m17n_object_unref()` によってユーザが明示的に行なわない限り、解放されない。その場合でも `data` は解放されない。

戻り値:

処理が成功すれば、`mtext_from_data()` は割り当てられた M-text へのポインタを返す。そうでなければ `NULL` を返し外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_MTEXT`

2.8.3.4 mtext_data()

```
void* mtext_data (
    MText * mt,
    enum MTextFormat * fmt,
    int * nunits,
    int * pos_idx,
    int * unit_idx )
```

2.8.3.5 mtext_len()

```
int mtext_len (
    MText * mt )
```

M-text 中の文字の数。

関数 `mtext_len()` は M-text `mt` 中の文字の数を返す。

2.8.3.6 mtext_ref_char()

```
int mtext_ref_char (
    MText * mt,
    int pos )
```

M-text 中の指定された位置の文字を返す。

関数 `mtext_ref_char()` は、M-text `mt` の位置 `pos` の文字を返す。エラーが検出された場合は -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_RANGE

2.8.3.7 mtext_set_char()

```
int mtext_set_char (
    MText * mt,
    int pos,
    int c )
```

M-text に一文字を設定する。

関数 `mtext_set_char()` は、テキストプロパティ無しの文字 `c` を M-text `mt` の位置 `pos` に設定する。

戻り値:

処理に成功すれば `mtext_set_char()` は 0 を返す。失敗すれば -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_RANGE

2.8.3.8 mtext_cat_char()

```
MText* mtext_cat_char (
    MText * mt,
    int c )
```

M-text に一文字追加する。

関数 `mtext_cat_char()` は、テキストプロパティ無しの文字 `c` を M-text `mt` の末尾に追加する。

戻り値:

この関数は変更された M-text `mt` へのポインタを返す。c が正しい文字でない場合には NULL を返す。

参照:

`mtext_cat()`, `mtext_ncat()`

2.8.3.9 mtext_dup()

```
MText* mtext_dup (
    MText * mt )
```

M-text のコピーを作る。

関数 [mtext_dup\(\)](#) は、M-text *mt* のコピーを作る。*mt* のテキストプロパティはすべて継承される。

戻り値:

この関数は作られたコピーへのポインタを返す。

参照:

[mtext_duplicate\(\)](#)

2.8.3.10 mtext_cat()

```
MText* mtext_cat (
    MText * mt1,
    MText * mt2 )
```

2 個の M-text を連結する。

関数 [mtext_cat\(\)](#) は、M-text *mt2* を M-text *mt1* の末尾に付け加える。*mt2* のテキストプロパティはすべて継承される。*mt2* は変更されない。

戻り値:

この関数は変更された M-text *mt1* へのポインタを返す。

参照:

[mtext_ncat\(\)](#), [mtext_cat_char\(\)](#)

2.8.3.11 mtext_ncat()

```
MText* mtext_ncat (
    MText * mt1,
    MText * mt2,
    int n )
```

M-text の一部を別の M-text に付加する。

関数 [mtext_ncat\(\)](#) は、M-text *mt2* の最初の *n* 文字を M-text *mt1* の末尾に付け加える。*mt2* のテキストプロパティはすべて継承される。*mt2* の長さが *n* 以下ならば、*mt2* のすべての文字が付加される。*mt2* は変更されない。

戻り値:

処理が成功した場合、[mtext_ncat\(\)](#) は変更された M-text *mt1* へのポインタを返す。エラーが検出された場合は NULL を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_cat\(\)](#), [mtext_cat_char\(\)](#)

2.8.3.12 mtext_cpy()

```
MText* mtext_cpy (
    MText * mt1,
    MText * mt2 )
```

M-text を別の M-text にコピーする。

関数 [mtext_cpy\(\)](#) は M-text mt2 を M-text mt1 に上書きコピーする。mt2 のテキストプロパティはすべて継承される。mt1 の長さは必要に応じて伸ばされる。mt2 は変更されない。

戻り値:

この関数は変更された M-text mt1 へのポインタを返す。

参照:

[mtext_ncpy\(\)](#), [mtext_copy\(\)](#)

2.8.3.13 mtext_ncpy()

```
MText* mtext_ncpy (
    MText * mt1,
    MText * mt2,
    int n )
```

M-text に含まれる最初の何文字かをコピーする。

関数 [mtext_ncpy\(\)](#) は、M-text mt2 の最初の n 文字を M-text mt1 に上書きコピーする。mt2 のテキストプロパティはすべて継承される。もし mt2 の長さが n よりも小さければ mt2 のすべての文字をコピーする。mt1 の長さは必要に応じて伸ばされる。mt2 は変更されない。

戻り値:

処理が成功した場合、[mtext_ncpy\(\)](#) は変更された M-text mt1 へのポインタを返す。エラーが検出された場合は NULL を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_cpy\(\)](#), [mtext_copy\(\)](#)

2.8.3.14 mtext_duplicate()

```
MText* mtext_duplicate (
    MText * mt,
    int from,
    int to )
```

既存の M-text の一部から新しい M-text をつくる。

関数 [mtext_duplicate\(\)](#) は、M-text *mt* の *from* (*from* 自体も含む) から *to* (*to* 自体は含まない) までの部分のコピーを作る。このとき *mt* のテキストプロパティはすべて継承される。*mt* そのものは変更されない。

戻り値:

処理が成功すれば、[mtext_duplicate\(\)](#) は作られた M-text へのポインタを返す。エラーが検出された場合は NULL を返し、外部変数 [merror.code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_dup\(\)](#)

2.8.3.15 mtext_copy()

```
MText* mtext_copy (
    MText * mt1,
    int pos,
    MText * mt2,
    int from,
    int to )
```

M-text に指定範囲の文字をコピーする。

関数 [mtext_copy\(\)](#) は、M-text *mt2* の *from* (*from* 自体も含む) から *to* (*to* 自体は含まない) までの範囲のテキストを M-text *mt1* の位置 *pos* から上書きコピーする。*mt2* のテキストプロパティはすべて継承される。*mt1* の長さは必要に応じて伸ばされる。*mt2* は変更されない。

戻り値:

処理が成功した場合、[mtext_copy\(\)](#) は変更された *mt1* へのポインタを返す。そうでなければ NULL を返し、外部変数 [merror.code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_cpy\(\)](#), [mtext_ncpy\(\)](#)

2.8.3.16 mtext_del()

```
int mtext_del (
    MText * mt,
    int from,
    int to )
```

指定範囲の文字を破壊的に取り除く。

関数 [mtext_del\(\)](#) は、M-text *mt* の *from* (*from* 自体も含む) から *to* (*to* 自体は含まない) までの文字を破壊的に取り除く。結果的に *mt* は長さが (*to* - *from*) だけ縮むことになる。

戻り値:

処理が成功すれば [mtext_del\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_ins\(\)](#)

2.8.3.17 mtext_ins()

```
int mtext_ins (
    MText * mt1,
    int pos,
    MText * mt2 )
```

M-text を別の M-text に挿入する。

関数 [mtext_ins\(\)](#) は M-text *mt1* の *pos* の位置に別の M-text *mt2* を挿入する。この結果 *mt1* の長さは *mt2* の長さ分だけ増える。挿入の際、*mt2* のテキストプロパティはすべて継承される。*mt2* そのものは変更されない。

戻り値:

処理が成功すれば [mtext_ins\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE, MERROR_MTEXT

参照:

[mtext_del\(\)](#), [mtext_insert\(\)](#)

2.8.3.18 mtext_insert()

```
int mtext_insert (
    MText * mt1,
    int pos,
    MText * mt2,
    int from,
    int to )
```

M-text の一部を別の M-text に挿入する。

関数 [mtext_insert\(\)](#) は M-text mt1 中の pos の位置に、別の M-text mt2 の from (from 自体も含む) から to (to 自体は含まない) までの文字を挿入する。結果的に mt1 は長さが (to - from) だけ伸びる。挿入の際、mt2 中のテキストプロパティはすべて継承される。

戻り値:

処理が成功すれば、[mtext_insert\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_MTEXT , MERROR_RANGE

参照:

[mtext_ins\(\)](#)

2.8.3.19 mtext_ins_char()

```
int mtext_ins_char (
    MText * mt,
    int pos,
    int c,
    int n )
```

M-text に文字を挿入する。

関数 [mtext_ins_char\(\)](#) は M-text mt の pos の位置に文字 c のコピーを n 個挿入する。この結果 mt1 の長さは n だけ増える。

戻り値:

処理が成功すれば [mtext_ins_char\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_RANGE

参照:

[mtext_ins](#), [mtext_del\(\)](#)

2.8.3.20 mtext_replace()

```
int mtext_replace (
    MText * mt1,
    int from1,
    int to1,
    MText * mt2,
    int from2,
    int to2 )
```

M-text の一部を別の M-text の一部で置換する。

関数 [mtext_replace\(\)](#) は、M-text mt1 の from1 (from1 自体も含む) から to1 (to1 自体は含まない) までを、M-text mt2 の from2 (from2 自体も含む) から to2 (to2 自体は含まない) で置き換える。新しく挿入された部分は、置き換える前のテキストプロパティすべてを継承する。

戻り値:

処理が成功すれば、[mtext_replace\(\)](#) は 0 を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

MERROR_MTEXT, MERROR_RANGE

参照:

[mtext_insert\(\)](#)

2.8.3.21 mtext_character()

```
int mtext_character (
    MText * mt,
    int from,
    int to,
    int c )
```

M-text 中で文字を探す。

関数 [mtext_character\(\)](#) は M-text mt 中で文字 c を探す。もし from が to より小さければ、探索は位置 from から末尾方向へ、最大 (to - 1) まで進む。そうでなければ位置 (from - 1) から先頭方向へ、最大 to まで進む。位置の指定に誤りがある場合は、from と to の両方に 0 が指定されたものとみなす。

戻り値:

もし c が見つければ、[mtext_character\(\)](#) はその最初の出現位置を返す。見つからなかった場合は外部変数 [merror_code](#) を変更せずに -1 を返す。エラーが検出された場合は -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

参照:

[mtext_chr\(\)](#), [mtext_rchr\(\)](#)

2.8.3.22 mtext_chr()

```
int mtext_chr (
    MText * mt,
    int c )
```

M-text 中で指定された文字が最初に現れる位置を返す。

関数 [mtext_chr\(\)](#) は M-text mt 中で文字 c を探す。探索は mt の先頭から末尾方向に進む。

戻り値:

もし c が見つければ、[mtext_chr\(\)](#) はその出現位置を返す。見つからなかった場合は -1 を返す。

エラー:

MERROR_RANGE

参照:

[mtext_rchr\(\)](#), [mtext_character\(\)](#)

2.8.3.23 mtext_rchr()

```
int mtext_rchr (
    MText * mt,
    int c )
```

M-text 中で指定された文字が最後に現れる位置を返す。

関数 [mtext_rchr\(\)](#) は M-text mt 中で文字 c を探す。探索は mt の最後から先頭方向へと後向きに進む。

戻り値:

もし c が見つければ、[mtext_rchr\(\)](#) はその出現位置を返す。見つからなかった場合は -1 を返す。

エラー:

MERROR_RANGE

参照:

[mtext_chr\(\)](#), [mtext_character\(\)](#)

2.8.3.24 mtext_cmp()

```
int mtext_cmp (
    MText * mt1,
    MText * mt2 )
```

二つの M-text を文字単位で比較する。

関数 [mtext_cmp\(\)](#) は、M-text mt1 と mt2 を文字単位で比較する。

戻り値:

この関数は、mt1 と mt2 が等しければ 0、mt1 が mt2 より大きければ 1、mt1 が mt2 より小さければ -1 を返す。比較は文字コードに基づく。

参照:

[mtext_ncmp\(\)](#), [mtext_casecmp\(\)](#), [mtext_ncasecmp\(\)](#), [mtext_compare\(\)](#), [mtext_case_compare\(\)](#)

2.8.3.25 mtext_ncmp()

```
int mtext_ncmp (
    MText * mt1,
    MText * mt2,
    int n )
```

二つの M-text の先頭部分を文字単位で比較する。

関数 [mtext_ncmp\(\)](#) は、関数 [mtext_cmp\(\)](#) 同様の M-text 同士の比較を先頭から最大 n 文字までに関して行なう。

戻り値:

この関数は、mt1 と mt2 が等しければ 0、mt1 が mt2 より大きければ 1、mt1 が mt2 より小さければ -1 を返す。

参照:

[mtext_cmp\(\)](#), [mtext_casecmp\(\)](#), [mtext_ncasecmp\(\)](#), [mtext_compare\(\)](#), [mtext_case_compare\(\)](#)

2.8.3.26 mtext_compare()

```
int mtext_compare (
    MText * mt1,
    int from1,
    int to1,
    MText * mt2,
    int from2,
    int to2 )
```

二つの M-text の指定した領域同士を比較する.

関数 [mtext_compare\(\)](#) は二つの M-text `mt1` と `mt2` を文字単位で比較する。比較の対象は `mt1` のうち `from1` から `to1` までと、`mt2` のうち `from2` から `to2` までである。`from1` と `from2` は含まれ、`to1` と `to2` は含まれない。`from1` と `to1` (あるいは `from2` と `to2`) が等しい場合は長さゼロの M-text を意味する。範囲指定に誤りがある場合は、`from1` と `to1` (あるいは `from2` と `to2`) 両方に 0 が指定されたものとみなす。

戻り値:

この関数は、`mt1` と `mt2` が等しければ 0、`mt1` が `mt2` より大きければ 1、`mt1` が `mt2` より小さければ -1 を返す。比較は文字コードに基づく。

参照:

[mtext_cmp\(\)](#), [mtext_ncmp\(\)](#), [mtext_casecmp\(\)](#), [mtext_ncasecmp\(\)](#), [mtext_case_compare\(\)](#)

2.8.3.27 mtext_spn()

```
int mtext_spn (
    MText * mt,
    MText * accept )
```

ある集合の文字を M-text の中で探す.

関数 [mtext_spn\(\)](#) は、M-text `mt1` の先頭から M-text `mt2` に含まれる文字だけでできている部分の長さを返す。

参照:

[mtext_cspn\(\)](#)

2.8.3.28 mtext_cspn()

```
int mtext_cspn (
    MText * mt,
    MText * reject )
```

ある集合に属さない文字を M-text の中で探す.

関数 [mtext_cspn\(\)](#) は、M-text `mt1` の先頭部分で M-text `mt2` に含まれない文字だけでできている部分の長さを返す。

参照:

[mtext_spn\(\)](#)

2.8.3.29 mtext_pbrk()

```
int mtext_pbrk (
    MText * mt,
    MText * accept )
```

ある集合に属す文字を M-text の中から探す。

関数 **mtext_pbrk()** は、M-text mt1 中で M-text mt2 の文字のどれかが最初に現れる位置を調べる。

戻り値:

見つかった文字の、mt1 内における出現位置を返す。もしそのような文字がなければ -1 を返す。

2.8.3.30 mtext_tok()

```
MText* mtext_tok (
    MText * mt,
    MText * delim,
    int * pos )
```

M-text 中のトークンを探す。

関数 **mtext_tok()** は、M-text mt の中で位置 pos 以降最初に現れるトークンを探す。ここでトークンとは M-text delim の中に現れない文字だけからなる部分文字列である。pos の型が int ではなくて int へのポインタであることに注意。

戻り値:

もしトークンが見つければ **mtext_tok()** はそのトークンに相当する部分の mt をコピーし、そのコピーへのポインタを返す。この場合、pos は見つかったトークンの終端にセットされる。トークンが見つからなかった場合は外部変数 **merror_code** を変えずに NULL を返す。エラーが検出された場合は NULL を返し、変部変数 **merror_code** にエラーコードを設定する。

エラー:

MERROR_RANGE

2.8.3.31 mtext_text()

```
int mtext_text (
    MText * mt1,
    int pos,
    MText * mt2 )
```

M-text 中で別の M-text を探す。

関数 **mtext_text()** は、M-text mt1 中で位置 pos 以降に現れる M-text mt2 の最初の位置を調べる。テキストプロパティの違いは無視される。

戻り値:

mt1 中に mt2 が見つければ、**mtext_text()** はその最初の出現位置を返す。見つからない場合は -1 を返す。もし mt2 が空ならば 0 を返す。

2.8.3.32 mtext_search()

```
int mtext_search (
    MText * mt1,
    int from,
    int to,
    MText * mt2 )
```

M-text 中の特定の領域で別の M-text を探す。

関数 [mtext_search\(\)](#) は、M-text mt1 中の from から to までの間の領域で M-text mt2 が最初に現われる位置を調べる。テキストプロパティの違いは無視される。もし from が to より小さければ探索は位置 from から末尾方向へ、そうでなければ to から先頭方向へ進む。

戻り値:

mt1 中に mt2 が見つければ、[mtext_search\(\)](#) はその最初の出現位置を返す。見つからない場合は -1 を返す。もし mt2 が空ならば 0 を返す。

2.8.3.33 mtext_casecmp()

```
int mtext_casecmp (
    MText * mt1,
    MText * mt2 )
```

二つの M-text を大文字／小文字の区別を無視して比較する。

関数 [mtext_casecmp\(\)](#) は、関数 [mtext_cmp\(\)](#) 同様の M-text 同士の比較を、大文字／小文字の区別を無視して行なう。

戻り値:

この関数は、mt1 と mt2 が等しければ 0、mt1 が mt2 より大きければ 1、mt1 が mt2 より小さければ -1 を返す。

参照:

[mtext_cmp\(\)](#), [mtext_ncmp\(\)](#), [mtext_ncasecmp\(\)](#) [mtext_compare\(\)](#), [mtext_case_compare\(\)](#)

2.8.3.34 mtext_ncasecmp()

```
int mtext_ncasecmp (
    MText * mt1,
    MText * mt2,
    int n )
```

二つの M-text の先頭部分を大文字／小文字の区別を無視して比較する。

関数 [mtext_ncasecmp\(\)](#) は、関数 [mtext_casecmp\(\)](#) 同様の M-text 同士の比較を先頭から最大 n 文字までに行なう。

戻り値:

この関数は、mt1 と mt2 が等しければ 0、mt1 が mt2 より大きければ 1、mt1 が mt2 より小さければ -1 を返す。

参照:

[mtext_cmp\(\)](#), [mtext_casecmp\(\)](#), [mtext_ncasecmp\(\)](#) [mtext_compare\(\)](#), [mtext_case_compare\(\)](#)

2.8.3.35 mtext_case_compare()

```
int mtext_case_compare (
    MText * mt1,
    int from1,
    int to1,
    MText * mt2,
    int from2,
    int to2 )
```

二つの M-text の指定した領域を、大文字／小文字の区別を無視して比較する。

関数 [mtext_compare\(\)](#) は二つの M-text `mt1` と `mt2` を、大文字／小文字の区別を無視して文字単位で比較する。比較の対象は `mt1` の `from1` から `to1` まで、`mt2` の `from2` から `to2` までである。`from1` と `from2` は含まれ、`to1` と `to2` は含まれない。`from1` と `to1` (あるいは `from2` と `to2`) が等しい場合は長さゼロの M-text を意味する。範囲指定に誤りがある場合は、`from1` と `to1` (あるいは `from2` と `to2`) 両方に 0 が指定されたものと見なす。

戻り値:

この関数は、`mt1` と `mt2` が等しければ 0、`mt1` が `mt2` より大きければ 1、`mt1` が `mt2` より小さければ -1 を返す。比較は文字コードに基づく。

参照:

[mtext_cmp\(\)](#), [mtext_ncmp\(\)](#), [mtext_casecmp\(\)](#), [mtext_ncasecmp\(\)](#), [mtext_compare\(\)](#)

2.8.3.36 mtext_lowercase()

```
int mtext_lowercase (
    MText * mt )
```

M-text を小文字にする。

関数 [mtext_lowercase\(\)](#) は M-text `mt` 中の各文字を破壊的に小文字に変換する。変換に際して隣接する文字の影響を受けることがある。`mt` にテキストプロパティ `Mlanguage` が付いている場合は、それも変換に影響を与える。`mt` の長さは変わることがある。小文字に変換できなかった文字はそのまま残る。テキストプロパティはすべて継承される。

戻り値:

この関数は更新後の `mt` の長さを返す。

参照:

[mtext_titlecase\(\)](#), [mtext_uppercase\(\)](#)

2.8.3.37 mtext_titlecase()

```
int mtext_titlecase (
    MText * mt )
```

M-text をタイトルケースにする。

関数 [mtext_titlecase\(\)](#) は M-text `mt` 中で `cased` プロパティを持つ 最初の文字をタイトルケースに、そしてそれ以降の文字を小文字に破壊的に変換する。`mt` の長さは変わることがある。タイトルケースにに変換できなかった場合はそのまま変わらない。テキストプロパティはすべて継承される。

戻り値:

この関数は更新後の `mt` の長さを返す。

参照:

[mtext_lowercase\(\)](#), [mtext_uppercase\(\)](#)

2.8.3.38 mtext_uppercase()

```
int mtext_uppercase (
    MText * mt )
```

M-text を大文字にする。

関数 [mtext_uppercase\(\)](#) は M-text `mt` 中の各文字を破壊的に大文字に変換する。変換に際して隣接する文字の影響を受けることがある。`mt` にテキストプロパティ `Mlanguage` が付いている場合は、それも変換に影響を与える。`mt` の長さは変わることがある。大文字に変換できなかった文字はそのまま残る。テキストプロパティはすべて継承される。

戻り値:

この関数は更新後の `mt` の長さを返す。

参照:

[mtext_lowercase\(\)](#), [mtext_titlecase\(\)](#)

2.8.4 変数詳解

型定義

- typedef `MPlist` `*(MTextPropSerializeFunc)` (`void *val`)
シリアライズ関数の型宣言.
- typedef `void` `*(MTextPropDeserializeFunc)` (`MPlist *plist`)
デシリアライズ関数の型宣言.

列挙型

- enum `MTextPropertyControl` {
 `MTEXTPROP_FRONT_STICKY` = 0x01 ,
 `MTEXTPROP_REAR_STICKY` = 0x02 ,
 `MTEXTPROP_VOLATILE_WEAK` = 0x04 ,
 `MTEXTPROP_VOLATILE_STRONG` = 0x08 ,
 `MTEXTPROP_NO_MERGE` = 0x10 ,
 `MTEXTPROP_CONTROL_MAX` = 0x1F }
テキストプロパティを制御するフラグビット.

関数

- `void * mtext_get_prop` (`MText *mt`, `int pos`, `MSymbol key`)
テキストプロパティの一番上の値を得る.
- `int mtext_get_prop_values` (`MText *mt`, `int pos`, `MSymbol key`, `void **values`, `int num`)
テキストプロパティの値を複数個得る.
- `int mtext_get_prop_keys` (`MText *mt`, `int pos`, `MSymbol **keys`)
M-text の指定した位置のテキストプロパティのキーのリストを得る.
- `int mtext_put_prop` (`MText *mt`, `int from`, `int to`, `MSymbol key`, `void *val`)
- `int mtext_put_prop_values` (`MText *mt`, `int from`, `int to`, `MSymbol key`, `void **values`, `int num`)
同じキーのテキストプロパティを複数設定する.
- `int mtext_push_prop` (`MText *mt`, `int from`, `int to`, `MSymbol key`, `void *val`)
- `int mtext_pop_prop` (`MText *mt`, `int from`, `int to`, `MSymbol key`)
- `int mtext_prop_range` (`MText *mt`, `MSymbol key`, `int pos`, `int *from`, `int *to`, `int deeper`)
テキストプロパティが同じ値をとる範囲を調べる.
- `MTextProperty * mtext_property` (`MSymbol key`, `void *val`, `int control_bits`)
テキストプロパティを生成する.
- `MText * mtext_property_mtext` (`MTextProperty *prop`)
あるテキストプロパティを持つ M-text を返す.
- `MSymbol mtext_property_key` (`MTextProperty *prop`)
テキストプロパティのキーを返す.
- `void * mtext_property_value` (`MTextProperty *prop`)
テキストプロパティの値を返す.
- `int mtext_property_start` (`MTextProperty *prop`)
テキストプロパティの開始位置を返す.
- `int mtext_property_end` (`MTextProperty *prop`)
テキストプロパティの終了位置を返す.
- `MTextProperty * mtext_get_property` (`MText *mt`, `int pos`, `MSymbol key`)
一番上のテキストプロパティを得る.
- `int mtext_get_properties` (`MText *mt`, `int pos`, `MSymbol key`, `MTextProperty **props`, `int num`)

複数のテキストプロパティを得る.

- `int mtext_attach_property (MText *mt, int from, int to, MTextProperty *prop)`
M-text にテキストプロパティを付加する.
- `int mtext_detach_property (MTextProperty *prop)`
M-text からテキストプロパティを分離する.
- `int mtext_push_property (MText *mt, int from, int to, MTextProperty *prop)`
M-text にテキストプロパティをプッシュする.
- `MText * mtext_serialize (MText *mt, int from, int to, MPlist *property_list)`
- `MText * mtext_deserialize (MText *mt)`

変数

- `MSymbol Mtext_prop_serializer`
シリアライザ関数を指定するシンボル.
- `MSymbol Mtext_prop_deserializer`
デシリアライザ関数を指定するシンボル.

2.9.1 詳解

テキストプロパティを操作するための関数.

M-text 内の各文字は、テキストプロパティ と呼ばれるプロパティを持つことができる。テキストプロパティは、M-text の各部位に付加されたさまざまな情報を保持しており、アプリケーションプログラムはそれらの情報を統一的に扱うことができる。M-text 自体が豊富な情報を持つため、アプリケーションプログラム中の関数を簡素化することができる。

テキストプロパティはキーと値からなる。キーはシンボルであり、値は (void *) 型にキャストできるものなら何でもよい。他のタイプのプロパティと異なり、一つのテキストプロパティが複数の値を持つことが許される。「キーが K であるテキストプロパティ」のことを簡単に「K プロパティ」と呼ぶことがある。

2.9.2 型定義詳解

2.9.2.1 MTextPropSerializeFunc

```
typedef MPlist* (* MTextPropSerializeFunc) (void *val)
```

シリアライザ関数の型宣言.

シリアライザ関数の型である。あるシンボルのプロパティのキーが `Mtext_prop_serializer` であるとき、値はこの型でなくてはならない。

参照:

`mtext_serialize()`, `Mtext_prop_serializer`

2.9.2.2 MTextPropDeserializeFunc

```
typedef void*(* MTextPropDeserializeFunc) (MPList *plist)
```

デシリアライザ関数の型宣言.

デシリアライザ関数の型である。あるシンボルのプロパティのキーが `Mtext_prop_deserializer` であるとき、値はこの型でなくてはならない。

参照:

`Mtext_prop_deserialize()`, `Mtext_prop_deserializer`

2.9.3 列挙型詳解

2.9.3.1 MTextPropertyControl

```
enum MTextPropertyControl
```

テキストプロパティを制御するフラグビット.

関数 `mtext_property()` は以下のフラグビットの論理 OR を引数としてとることができる。フラグビットは生成されたテキストプロパティの振舞いを制御する。詳細は各フラグビットの説明を参照。

列挙値

MTEXTPROP_FRONT_STICKY	このビットが on ならば、このテキストプロパティの始まる点あるいは中間に挿入された M-text はこのテキストプロパティを継承する。
MTEXTPROP_REAR_STICKY	このビットが on ならば、このテキストプロパティの終わる点あるいは中間に挿入された M-text はこのテキストプロパティを継承する。
MTEXTPROP_VOLATILE_WEAK	このビットが on ならば、このテキストプロパティの範囲内のテキストが変更された場合テキストプロパティは取り除かれる。
MTEXTPROP_VOLATILE_STRONG	このビットが on ならば、このテキストプロパティの範囲内のテキストあるいは別のテキストプロパティが変更された場合このテキストプロパティは取り除かれる。
MTEXTPROP_NO_MERGE	このビットが on ならば、このテキストプロパティは他のプロパティと自動的にマージされない。
MTEXTPROP_CONTROL_MAX	

2.9.4 関数詳解

2.9.4.1 mtext_get_prop()

```
void* mtext_get_prop (
    MText * mt,
    int pos,
    MSymbol key )
```

テキストプロパティの一番上の値を得る。

関数 [mtext_get_prop\(\)](#) は、M-text *mt* 内の位置 *pos* にある文字のテキストプロパティのうち、キーが *key* であるものを探す。

戻り値:

テキストプロパティがみつければ、[mtext_get_prop\(\)](#) はそのプロパティの値を返す。値が複数存在するときは、一番上の値を返す。見つからなければ外部変数 [merror_code](#) を変更することなく NULL を返す。

エラーが検出された場合 [mtext_get_prop\(\)](#) は NULL を返し、外部変数 [merror_code](#) にエラーコードを設定する。

覚え書き

エラーなしで NULL が返された場合には二つの可能性がある。

- *pos* の位置の文字は *key* をキーとするプロパティを持たない。
- その文字はそのようなプロパティを持ち、その値が NULL である。

この二つを区別する必要がある場合には、関数 [mtext_get_prop_values\(\)](#) を代わりに使用すること。

エラー:

[MERROR_RANGE](#), [MERROR_SYMBOL](#)

参照:

[mtext_get_prop_values\(\)](#), [mtext_put_prop\(\)](#), [mtext_put_prop_values\(\)](#), [mtext_push_prop\(\)](#), [mtext_pop_prop\(\)](#),
[mtext_prop_range\(\)](#)

2.9.4.2 mtext_get_prop_values()

```
int mtext_get_prop_values (
    MText * mt,
    int pos,
    MSymbol key,
    void ** values,
    int num )
```

テキストプロパティの値を複数個得る。

関数 `mtext_get_prop_values()` は、M-text `mt` 内で `pos` という位置にある文字のプロパティのうち、キーが `key` であるものを探す。もしそのようなプロパティが見つければ、それが持つ値 (複数可) を `values` の指すメモリ領域に格納する。`num` は格納する値の数の上限である。

戻り値:

処理が成功すれば、`mtext_get_prop_values()` は実際にメモリに格納された値の数を返す。`pos` の位置の文字が `key` をキーとするプロパティを持たなければ 0 を返す。エラーが検出された場合は -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`, `MERROR_SYMBOL`

参照:

`mtext_get_prop()`, `mtext_put_prop()`, `mtext_put_prop_values()`, `mtext_push_prop()`, `mtext_pop_prop()`,
`mtext_prop_range()`

2.9.4.3 mtext_get_prop_keys()

```
int mtext_get_prop_keys (
    MText * mt,
    int pos,
    MSymbol ** keys )
```

M-text の指定した位置のテキストプロパティのキーのリストを得る。

関数 `mtext_get_prop_keys()` は、M-text `mt` 内で `pos` の位置にあるすべてのテキストプロパティのキーを要素とする配列を作り、その配列のアドレスを `*keys` に設定する。この配列のために確保されたメモリを解放するのはユーザの責任である。

戻り値:

処理が成功すれば `mtext_get_prop_keys()` は得られたリストの長さを返す。そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`

参照:

`mtext_get_prop()`, `mtext_put_prop()`, `mtext_put_prop_values()`, `mtext_get_prop_values()`, `mtext_push_prop()`,
`mtext_pop_prop()`

2.9.4.4 mtext_put_prop()

```
int mtext_put_prop (
    MText * mt,
    int from,
    int to,
    MSymbol key,
    void * val )
```

@brief テキストプロパティを設定する。

関数 `mtext_put_prop()` は、M-text `@b mt` の `@b from` (含まれる) から `@b to` (含まれない) の範囲の文字に、キーが `@b key` で値が `@b val` であるようなテキストプロパティを設定する。この関数によって

```

                                FROM                TO
M-text:      |<-----|----- MT -----|----->|
PROP:        <----- OLD_VAL ----->
```

は次のようになる。

```

                                FROM                TO
M-text:      |<-----|----- MT -----|----->|
PROP:        <-- OLD_VAL-><----- VAL -----><-- OLD_VAL-->
```

@par 戻り値:

処理が成功すれば `mtext_put_prop()` は 0 を返す。そうでなければ -1 を返し、外部変数 `#merror_code` にエラーコードを設定する。

```
@latexonly \IPAlabel{mtext_put_prop} @endlatexonly
```

@par エラー:

```
@c MERROR_RANGE, @c MERROR_SYMBOL
```

@par 参照:

```
mtext_put_prop_values(), mtext_get_prop(),
mtext_get_prop_values(), mtext_push_prop(),
mtext_pop_prop(), mtext_prop_range()
```

2.9.4.5 mtext_put_prop_values()

```
int mtext_put_prop_values (
    MText * mt,
    int from,
    int to,
    MSymbol key,
    void ** values,
    int num )
```

同じキーのテキストプロパティを複数設定する。

関数 `mtext_put_prop_values()` は、M-Text `mt` の `from` (含まれる) から `to` (含まれない) の範囲の文字に、テキストプロパティを設定する。テキストプロパティのキーは `key` によって、値 (複数可) は `values` によって指定される。`num` は設定される値の個数である。

戻り値:

処理が成功すれば、`mtext_put_prop_values()` は 0 を返す。そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`, `MERROR_SYMBOL`

参照:

`mtext_put_prop()`, `mtext_get_prop()`, `mtext_get_prop_values()`, `mtext_push_prop()`, `mtext_pop_prop()`, `mtext_prop_range()`

2.9.4.6 mtext_push_prop()

```
int mtext_push_prop (
    MText * mt,
    int from,
    int to,
    MSymbol key,
    void * val )
```

@brief テキストプロパティをプッシュする。

関数 `mtext_push_prop()` は、キーが `@b key` で値が `@b val` であるテキストプロパティを、M-text `@b mt` 中の `@b from` (含まれる) から `@b to` (含まれない) の範囲の文字にプッシュする。この関数によって

```

                FROM                      TO
M-text: |<-----|----- MT -----|----->|
PROP   : <----- OLD_VAL ----->
```

は次のようになる。

```

                FROM                      TO
M-text: |<-----|----- MT -----|----->|
PROP   : <----- OLD_VAL ----->
PROP   : <----- VAL ----->
```

@par 戻り値:

処理が成功すれば、`mtext_push_prop()` は 0 を返す。そうでなければ -1 を返し、外部変数 `#merror_code` にエラーコードを設定する。

@latexonly \IPAlabel{mtext_push_prop} @endlatexonly

@par エラー:

@c MERROR_RANGE, @c MERROR_SYMBOL

@par 参照:

`mtext_put_prop()`, `mtext_put_prop_values()`,
`mtext_get_prop()`, `mtext_get_prop_values()`,
`mtext_pop_prop()`, `mtext_prop_range()`

2.9.4.7 mtext_pop_prop()

```
int mtext_pop_prop (
    MText * mt,
    int from,
    int to,
    MSymbol key )
```

@brief テキストプロパティをポップする。

関数 `mtext_pop_prop()` は、キーが `@b key` であるテキストプロパティのうち一番上のものを、M-text `@b mt` の `@b from` (含まれる) から `@b to` (含まれない) の範囲の文字から取り除く。

指定範囲の文字がそのようなプロパティを持たないならば、この関数は何もしない。この関数によって、

```

                FROM                      TO
M-text: |<-----|----- MT -----|----->|
PROP   : <----- OLD_VAL ----->
```

は以下のようになる。

```

                FROM                      TO
M-text: |<-----|----- MT -----|----->|
PROP   : <--OLD_VAL-->|                  |<--OLD_VAL-->|
```

@par 戻り値:

処理が成功すれば、`mtext_pop_prop()` は 0 を返す。そうでなければ -1 を返し、外部変数 `#merror_code` にエラーコードを設定する。

@latexonly \IPAlabel{mtext_pop_prop} @endlatexonly

@par エラー:

@c MERROR_RANGE, @c MERROR_SYMBOL

@par 参照:

`mtext_put_prop()`, `mtext_put_prop_values()`,
`mtext_get_prop()`, `mtext_get_prop_values()`,
`mtext_push_prop()`, `mtext_prop_range()`

2.9.4.8 mtext_prop_range()

```
int mtext_prop_range (
    MText * mt,
    MSymbol key,
    int pos,
    int * from,
    int * to,
    int deeper )
```

テキストプロパティが同じ値をとる範囲を調べる。

関数 `mtext_prop_range()` は、指定したテキストプロパティの値が同じである連続した文字の範囲を調べる。まず M-text `mt` の `pos` の位置にある文字のプロパティのうち、キー `key` で指定されたものの値を見つける。そして前後の文字も `key` のプロパティの値が同じであるかどうかを調べる。見つけた範囲の最初と最後を、それぞれ `from` と `to` にポイントされる変数に保存する。`from` に保存される文字の位置は見つけた範囲に含まれるが、`to` は含まれない。(`to` の前で同じ値をとる範囲は終わる。) この範囲指定法は、関数 `mtext_put_prop()` などと共通である。

`deeper` が 0 でなければ、`key` というキーを持つプロパティのうち一番上のものだけでなく、スタック中のすべてのものが比較される。

`from` が NULL ならば、範囲の始まりは探索しない。`to` が NULL ならば、範囲の終りは探索しない。

戻り値:

処理が成功すれば、`mtext_prop_range()` は key プロパティの値の数を返す。そうでなければ-1を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`, `MERROR_SYMBOL`

参照:

`mtext_put_prop()`, `mtext_put_prop_values()`, `mtext_get_prop()`, `mtext_get_prop_values()`, `mtext_pop_prop()`, `mtext_push_prop()`

2.9.4.9 mtext_property()

```
MTextProperty* mtext_property (
    MSymbol key,
    void * val,
    int control_bits )
```

テキストプロパティを生成する。

関数 `mtext_property()` は key をキー、val を値とする新しく割り当てられたテキストプロパティを返す。生成したテキストプロパティはいかなる M-text にも付加されていない、すなわち分離して (detached) いる。

control_bits は 0 であるか enum MTextPropertyControl の論理 OR でなくてはならない。

2.9.4.10 mtext_property_mtext()

```
MText* mtext_property_mtext (
    MTextProperty * prop )
```

あるテキストプロパティを持つ M-text を返す。

関数 `mtext_property_mtext()` は、テキストプロパティ prop が付加されている M-text を返す。その時点で prop が分離していれば NULL を返す。

2.9.4.11 mtext_property_key()

```
MSymbol mtext_property_key (
    MTextProperty * prop )
```

テキストプロパティのキーを返す。

関数 `mtext_property_key()` は、テキストプロパティ prop のキー (シンボル) を返す。

2.9.4.12 mtext_property_value()

```
void* mtext_property_value (
    MTextProperty * prop )
```

テキストプロパティの値を返す。

関数 `mtext_property_value()` は、テキストプロパティ `prop` の値を返す。

2.9.4.13 mtext_property_start()

```
int mtext_property_start (
    MTextProperty * prop )
```

テキストプロパティの開始位置を返す。

関数 `mtext_property_start()` は、テキストプロパティ `prop` の開始位置を返す。開始位置とは M-text 中で `prop` が始まる文字位置である。`prop` が分離されていれば、-1 を返す。

2.9.4.14 mtext_property_end()

```
int mtext_property_end (
    MTextProperty * prop )
```

テキストプロパティの終了位置を返す。

関数 `mtext_property_end()` は、テキストプロパティ `prop` の終了位置を返す。終了位置とは M-text 中で `prop` が終る文字位置である。`prop` が分離されていれば、-1 を返す。

2.9.4.15 mtext_get_property()

```
MTextProperty* mtext_get_property (
    MText * mt,
    int pos,
    MSymbol key )
```

一番上のテキストプロパティを得る。

関数 `mtext_get_property()` は M-text `mt` の位置 `pos` の文字がキーが `key` であるテキストプロパティを持つかどうかを調べる。

戻り値:

テキストプロパティが見つければ、`mtext_get_property()` はそれを返す。複数ある場合には、一番上のものを返す。見つからなければ、外部変数 `merror_code` を変えることなく NULL を返す。

エラーが検出された場合 `mtext_get_property()` は NULL を返し、外部変数 `merror_code` にエラーコードを設定する。

2.9.4.16 mtext_get_properties()

```
int mtext_get_properties (
    MText * mt,
    int pos,
    MSymbol key,
    MTextProperty ** props,
    int num )
```

複数のテキストプロパティを得る。

関数 `mtext_get_properties()` は M-text `mt` の位置 `pos` の文字がキーが `key` であるテキストプロパティを持つかどうかを調べる。そのようなプロパティがみつければ、`props` が指すメモリ領域に保存する。`num` は保存されるプロパティの数の上限である。

戻り値:

処理が成功すれば、`mtext_get_properties()` は実際に保存したプロパティ の数を返す。`pos` の位置の文字がキーが `key` であるプロパティを持た なければ、0 が返る。エラーが検出された場合には、`mtext_get_properties()` は -1 を返し、外部変数 `merror_code` にエラー コードを設定する。

2.9.4.17 mtext_attach_property()

```
int mtext_attach_property (
    MText * mt,
    int from,
    int to,
    MTextProperty * prop )
```

M-text にテキストプロパティを付加する。

関数 `mtext_attach_property()` は、M-text `mt` の `from` から `to` ま での領域にテキストプロパティ `prop` を付加する。もし `prop` が既に M-text に付加されていれば、`mt` に付加する前に分離される。

戻り値:

処理に成功すれば、`mtext_attach_property()` は 0 を返す。そうでなければ -1 を返して外部変数 `::merror_code` にエラーコードを設定する。

2.9.4.18 mtext_detach_property()

```
int mtext_detach_property (
    MTextProperty * prop )
```

M-text からテキストプロパティを分離する。

関数 `mtext_detach_property()` はテキストプロパティ `prop` を分離する。

戻り値:

この関数は常に 0 を返す。

2.9.4.19 mtext_push_property()

```
int mtext_push_property (
    MText * mt,
    int from,
    int to,
    MTextProperty * prop )
```

M-text にテキストプロパティをプッシュする。

関数 `mtext_push_property()` は、テキストプロパティ `prop` を、M-text `mt` 中の `from` (含まれる) から `to` (含まれない) の範囲の文字にプッシュする。

戻り値:

処理に成功すれば、`mtext_push_property()` は 0 を返す。そうでなければ -1 を返して外部変数 `::merror_code` にエラーコードを設定する。

2.9.4.20 mtext_serialize()

```
MText* mtext_serialize (
    MText * mt,
    int from,
    int to,
    MPlist * property_list )
```

@brief M-text 中のテキストプロパティをシリアル化する。

関数 `mtext_serialize()` は M-text `@b mt` の `@b from` から `@b to` までのテキストをシリアル化する。シリアル化した結果は XML 形式の M-text である。 `@b property_list` はシリアル化されるテキストプロパティを限定する。対象となるテキストプロパティは、そのキーが

@li `@b property_list` の要素の値として現われ、かつ
 @li シンボルプロパティ `#Mtext_prop_serializer` を持つ

もののみである。この条件を満たすテキストプロパティは、生成される XML 表現中で "property" 要素にシリアル化される。

生成される XML の DTD は以下の通り:

```
<!DOCTYPE mtext [
    <!ELEMENT mtext (property*,body+)>
    <!ELEMENT property EMPTY>
    <!ELEMENT body (#PCDATA)>
    <!ATTLIST property key CDATA #REQUIRED>
    <!ATTLIST property value CDATA #REQUIRED>
    <!ATTLIST property from CDATA #REQUIRED>
    <!ATTLIST property to CDATA #REQUIRED>
    <!ATTLIST property control CDATA #REQUIRED>
]>
```

この関数は `libxml2` ライブラリに依存する。`m17n` ライブラリが `libxml2` 無しに設定されている場合、この関数は常に失敗する。

@par 戻り値:

処理に成功すれば、`mtext_serialize()` は XML 形式で M-text を返す。そうでなければ `@c NULL` を返して外部変数 `#merror_code` にエラーコードを設定する。

@par 参照:

`mtext_deserialize()`, `#Mtext_prop_serializer`

2.9.4.21 mtext_deserialize()

```
MText* mtext_deserialize (
    MText * mt )
```

@brief M-text 中のテキストプロパティをデシリアライズする。

関数 `mtext_deserialize()` は M-text @b mt をデシリアライズする。@b mt は次の DTD を持つ XML でなくてはならない。

```
<!DOCTYPE mtext [
  <!ELEMENT mtext (property*,body+)>
  <!ELEMENT property EMPTY>
  <!ELEMENT body (#PCDATA)>
  <!ATTLIST property key CDATA #REQUIRED>
  <!ATTLIST property value CDATA #REQUIRED>
  <!ATTLIST property from CDATA #REQUIRED>
  <!ATTLIST property to CDATA #REQUIRED>
  <!ATTLIST property control CDATA #REQUIRED>
]>
```

この関数は `libxml2` ライブラリに依存する。`m17n` ライブラリが `libxml2` 無しに設定されている場合、この関数は常に失敗する。

@par 戻り値:
処理に成功すれば、`mtext_serialize()` は得られた M-text を返す。そうでなければ `@c NULL` を返して外部変数 `#merror_code` にエラーコードを設定する。

@par 参照:
`mtext_serialize()`, `#Mtext_prop_deserializer`

2.9.5 変数詳解

2.9.5.1 Mtext_prop_serializer

```
MSymbol Mtext_prop_serializer
```

シリアライザ関数を指定するシンボル。

テキストプロパティをシリアライズするためには、そのテキストプロパティ用のシリアライザ関数を与えてはならない。具体的には、`Mtext_prop_serializer` をキーとし、適切なシリアライズ関数へのポインタを値とするシンボルプロパティを指定する。

参照:
`mtext_serialize()`, `MTextPropSerializeFunc`

2.9.5.2 Mtext_prop_deserializer

`MSymbol Mtext_prop_deserializer`

デシリアライザ関数を指定するシンボル。

テキストプロパティをデシリアライズするためには、そのテキストプロパティ用のデシリアライザ関数を与えてはならない。具体的には、`Mtext_prop_deserializer` をキーとし、適切なデシリアライズ関数へのポインタを値とするシンボルプロパティを指定する。

参照:

`mtext_deserialize()`, `MTextPropSerializeFunc`

2.10 データベース

m17n データベースにとそれに関する API.

データベース 連携図



型定義

- `typedef struct MDatabase MDatabase`
データベースの型宣言.

関数

- `MDatabase * mdatabase_find (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3)`
データベース中のデータを探す.
- `MPlist * mdatabase_list (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3)`
m17n データベースのデータリストを返す.
- `MDatabase * mdatabase_define (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3, void (*loader)(MSymbol *, void *), void *extra_info)`
m17n データベースのデータを定義する.
- `void * mdatabase_load (MDatabase *mdb)`
データベースからデータをロードする.
- `MSymbol * mdatabase_tag (MDatabase *mdb)`
データのタグを得る.

変数

- `char * mdatabase_dir`

2.10.1 詳解

m17n データベースにとそれに関する API.

アプリケーション固有のデータ用ディレクトリ.

m17n ライブラリは必要に応じて動的に m17n データベース から情報を取得する。またアプリケーション プログラムも、独自のデータを m17n データベースに追加し、それを動的に取得することができる。アプリケーションプログラムが独自のデータを追加・取得するには、変数 `mdatabase_dir` にそのアプリケーション固有のディレクトリをセットし、その中にデータを格納する。ユーザがそのデータをオーバーライトしたいときは、環境変数 "M17NDIR" で指定されるディレクトリ（指定されていないときは `~/m17n.d` というディレクトリ）に別のデータを置く。

m17n データベースには複数の多様なデータが含まれており、各データは TAG0, TAG1, TAG2, TAG3（すべてシンボル）の 4 つのタグによって識別される。

TAG0 によって、データベース内のデータのタイプは次のように指定される。

- TAG0 が `Mchar.table` であるデータは `chartable` タイプと呼ばれ、各文字に関する情報を提供する。この場合 TAG1 は情報の種類を指定するシンボルであり、`::Msymbol`, `Minteger`, `Mstring`, `Mtext`, `Mplist` のいずれかである。TAG2 と TAG3 は任意のシンボルでよい。
- TAG0 が `Mcharset` であるデータは `charset` タイプと呼ばれ、文字セット用のデコード／エンコードマップを提供する。この場合 TAG1 は文字セットのシンボルでなければならない。TAG2 と TAG3 は任意のシンボルでよい。
- TAG0 が `Mchar.table` でも `Mcharset` でもない場合、そのデータは `plist` タイプである。詳細に関しては関数 `mdatabaseLoad()` の説明を参照のこと。この場合 TAG1, TAG2, TAG3 は任意のシンボルでよい。

特定のタグを持つデータベースを `<TAG0, TAG1, TAG2, TAG3>` という形式で表す。

アプリケーションプログラムは、まず関数 `mdatabase.find()` を使ってデータベースに関する情報を保持するオブジェクト (`::MDatabase` 型) へのポインタを得る。それに成功したら、`mdatabaseLoad()` によって実際にデータベースをロードする。構造体 `MDatabase` 自身がどう実装されているかは、アプリケーションプログラムからは見えない。

アプリケーションプログラムが、そのプログラム固有のデータや m17n データベースを上書きするデータを提供する場合には、マクロ `M17N_JNIT()` を呼ぶ前にこの変数をデータファイルを含むディレクトリ名にセットしなくてはならない。ディレクトリには "mdb.dir" ファイルをおくことができる。その "mdb.dir" ファイルには、`mdbDir(5)` で説明されているフォーマットでデータ定義のリストを記述する。

デフォルトの値は NULL である。

2.10.2 型定義詳解

2.10.2.1 MDatabase

```
typedef struct MDatabase MDatabase
```

データベースの型宣言.

MDatabase 型はデータベースオブジェクト用の構造体である。内部構造はアプリケーションプログラムからは見えない。

2.10.3 関数詳解

2.10.3.1 mdatabase_find()

```
MDatabase* mdatabase_find (
    MSymbol tag0,
    MSymbol tag1,
    MSymbol tag2,
    MSymbol tag3 )
```

データベース中のデータを探す.

関数 **mdatabase_find()** は、m17n 言語情報ベース中で **tag0** から **tag3** までのタグを持つデータを探し、それへのポインタを返す。そのようなデータがなければ NULL を返す。

2.10.3.2 mdatabase_list()

```
MList* mdatabase_list (
    MSymbol tag0,
    MSymbol tag1,
    MSymbol tag2,
    MSymbol tag3 )
```

m17n データベースのデータリストを返す.

関数 **mdatabase_list()** は m17n データベース中から **tag0** から **tag3** までのタグを持つデータを探し、そのリストを **plist** として返す。tag_n が **Mnil** であった場合には、任意のタグにマッチするワイルドカードとして取り扱われる。返される **plist** の各要素はキーとして **Mt** を、値として **MDatabase** 型へのポインタを持つ。

2.10.3.3 mdatabase_define()

```
MDatabase* mdatabase_define (
    MSymbol tag0,
    MSymbol tag1,
    MSymbol tag2,
    MSymbol tag3,
    void (*)(MSymbol *, void *) loader,
    void * extra_info )
```

m17n データベースのデータを定義する。

関数 [mdatabase_define\(\)](#) は tag0 から tag3 までのタグおよび付加情報 `extra_info` を持つデータを定義する。

`loader` はそのデータのロードに用いられる関数へのポインタである。この関数は [mdatabase_load\(\)](#) から `tags` と `extra_info` という二つの引数付きで呼び出される。ここで `tags` は tag0 から tag3 までの配列である。

もし `loader` が NULL なら、m17n ライブラリ標準のローダが使われる。この場合には `extra_info` はデータを含むファイル名でなくてはならない。

戻り値:

処理に成功すれば [mdatabase_define\(\)](#) は定義されたデータベースへのポインタを返す。このポインタは関数 [mdatabase_load\(\)](#) の引数として用いることができる。そうでなければ NULL を返す。

参照:

[mdatabase_load\(\)](#), [mdatabase_define\(\)](#)

2.10.3.4 mdatabase_load()

```
void* mdatabase_load (
    MDatabase * mdb )
```

データベースからデータをロードする。

関数 [mdatabase_load\(\)](#) は `mdb` が指すデータをロードし、その中身を返す。返されるものはデータのタイプによって異なる。

データが `plist` タイプならば、`plist` へのポインタを返す。

データが `chartable` タイプならば文字テーブルを返す。文字テーブルのデフォルト値は、データの第 2 タグによって以下のように決まる。

- タグが `MSymbol` なら、デフォルト値は `Mnil`
- タグが `Minteger` なら、デフォルト値は -1
- それ以外なら、デフォルト値は NULL

データが `charset` タイプならば長さ 2 の `plist` を返す（キーは共に `::Mt`）。最初の要素の値はコードポイントを対応する文字コードにマップする整数の配列である。2 番目の要素の値は逆のマップをする文字テーブルである。この文字セットは予め定義されていなければならない。

参照:

[mdatabase_load\(\)](#), [mdatabase_define\(\)](#)

2.10.3.5 mdatabase_tag()

```
MSymbol* mdatabase_tag (
    MDatabase * mdb )
```

データのタグを得る.

関数 `mdatabase_tag()` は、データ `mdb` のタグ（シンボル）の配列を返す。配列の長さは 4 である。

2.10.4 変数詳解

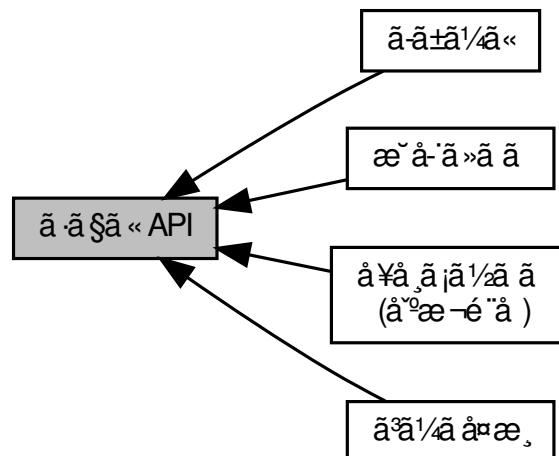
2.10.4.1 mdatabase_dir

```
char* mdatabase_dir
```

2.11 シェル API

libm17n.so が提供する API

シェル API 連携図



モジュール

- [文字セット](#)
文字セットオブジェクトとそれに関する API.
- [コード変換](#)
コード系オブジェクトとそれに関する API.
- [ロケール](#)
ロケールオブジェクトとそれに関する API.
- [入力メソッド \(基本部分\)](#)
入力メソッド用API.

2.11.1 詳解

libm17n.so が提供する API

2.12 文字セット

文字セットオブジェクトとそれに関する API.

文字セット 連携図



マクロ定義

- `#define MCHAR_INVALID_CODE`
無効なコードポイント.

関数

- `MSymbol mchar_define_charset` (const char *name, MPlist *plist)
- `MSymbol mchar_resolve_charset` (MSymbol symbol)
文字セット名を解決する.
- `int mchar_list_charset` (MSymbol **symbols)
文字セットを表わすシンボルを列挙する.
- `int mchar_decode` (MSymbol charset_name, unsigned code)
コードポイントをデコードする.
- `unsigned mchar_encode` (MSymbol charset_name, int c)
文字コードをエンコードする.
- `int mchar_map_charset` (MSymbol charset_name, void(*func)(int from, int to, void *arg), void *func_arg)
指定した文字セットのすべての文字に対して関数を呼ぶ.

変数

- `MSymbol Mcharset`

変数: 文字セットを表現する定義済みシンボル.

以下の各シンボルは、定義済み文字セットを表現する。

- [MSymbol Mcharset_ascii](#)
ASCII 文字セットを表現するシンボル.
- [MSymbol Mcharset_iso_8859_1](#)
ISO/IEC 8859-1:1998 文字セットを表現するシンボル.
- [MSymbol Mcharset_unicode](#)
Unicode 文字セットを表現するシンボル.
- [MSymbol Mcharset_m17n](#)
全文字を含む文字セットを表現するシンボル.
- [MSymbol Mcharset_binary](#)
正しくデコードできない文字の文字セットを表現するシンボル.

変数: `mchar_define_charset` 用のパラメータ・キー

これらは、関数 `mchar_define_charset()` 用のパラメータ・キーとして使われるシンボルである。詳しくはこの関数の解説を参照のこと。

- [MSymbol Mmethod](#)
- [MSymbol Mdimension](#)
- [MSymbol Mmin_range](#)
- [MSymbol Mmax_range](#)
- [MSymbol Mmin_code](#)
- [MSymbol Mmax_code](#)
- [MSymbol Mascii_compatible](#)
- [MSymbol Mfinal_byte](#)
- [MSymbol Mrevision](#)
- [MSymbol Mmin_char](#)
- [MSymbol Mmapfile](#)
- [MSymbol Mparents](#)
- [MSymbol Msubset_offset](#)
- [MSymbol Mdefine_coding](#)
- [MSymbol Malias](#)

変数: 文字セットのメソッド指定に使われるシンボル

これらは、文字セットのメソッドを指定するための定義済みシンボルであり、文字セットの `Mmethod` パラメータの値となることができる。この値は関数 `mchar_define_charset()` の引数として使われる。

メソッドとは、コードポイントと文字コードを相互変換する際の方式のことである。詳しくは関数 `mchar_define_charset()` の解説を参照のこと。

- [MSymbol Moffset](#)
- [MSymbol Mmap](#)
マップ型のメソッドを示すシンボル.
- [MSymbol Munify](#)
ユニファイ型のメソッドを示すシンボル.
- [MSymbol Msubset](#)
サブセット型のメソッドを示すシンボル.
- [MSymbol Msuperset](#)
スーパーセット型のメソッドを示すシンボル.

2.12.1 詳解

文字セットオブジェクトとそれに関する API.

シンボル `Mcharset`.

`m17n` ライブラリは、符号化文字集合 (CCS) を文字セットと呼ぶオブジェクトで表現する。`m17n` ライブラリは多くの符号化文字集合をあらかじめサポートしているし、アプリケーションプログラムが独自に文字セットを追加することも可能である。一つの文字は複数の文字セットに属してもよい。

`m17n` ライブラリは、以下の概念を区別している:

- コードポイント とは、CCS がその中の個々の文字に対して定義する数値である。コードポイントは連続しているとは限らない。コードポイントは `unsigned` 型によって表される。無効なコードポイントはマクロ `MCHAR_INVALID_CODE` で表される。
- 文字インデックス とは、CCS 内で各文字に割り当てられる正規化されたインデックスである。文字インデックスが `N` の文字は、CCS 中の全文字をコードポイント順に並べたときに `N` 番目に現われる。CCS 中の文字インデックスは連続しており、0 から始まる。
- 文字コード とは、`m17n` ライブラリ内における文字の内部表現であり、21 ビット以上の長さを持つ符号付き整数である。

各文字セットオブジェクトは、その文字セットに属する文字のコードポイントと文字コードとの間の変換を規定する。コードポイントから文字コードへの変換をデコードと呼び、文字コードからコードポイントへの変換をエンコードと呼ぶ。

デコードされた **M-text** は、キーが `Mcharset` であるようなテキストプロパティを持つ。シンボル `Mcharset` は `"charset"` という名前を持つ。

2.12.2 マクロ定義詳解

2.12.2.1 `MCHAR_INVALID_CODE`

```
#define MCHAR_INVALID_CODE
```

無効なコードポイント.

マクロ `MCHAR_INVALID_CODE` は無効なコードポイントを示す。

2.12.3 関数詳解

2.12.3.1 mchar_define_charset()

```
MSymbol mchar_define_charset (
    const char * name,
    MPlist * plist )
```

2.12.3.2 mchar_resolve_charset()

```
MSymbol mchar_resolve_charset (
    MSymbol symbol )
```

文字セット名を解決する。

関数 [mchar_resolve_charset\(\)](#) は `symbol` が文字セットを示していればそれを返す。

そうでなければ、`symbol` を文字セット名として正規化し、それが文字セットを示していれば正規化したものを返す。そうでなければ、`::Mnil` を返す。

2.12.3.3 mchar_list_charset()

```
int mchar_list_charset (
    MSymbol ** symbols )
```

文字セットを表わすシンボルを列挙する。

関数 [mchar_list_charsets\(\)](#) は、文字セットを示すシンボルを並べた配列を作り、`symbols` でポイントされた場所にこの配列へのポインタを置き、配列の長さを返す。

2.12.3.4 mchar_decode()

```
int mchar_decode (
    MSymbol charset_name,
    unsigned code )
```

コードポイントをデコードする。

関数 [mchar_decode\(\)](#) は、シンボル `charset_name` で示される文字セット内の `code` というコードポイントをデコードして文字コードを得る。

戻り値:

デコードが成功すれば、[mchar_decode\(\)](#) はデコードされた文字コードを返す。そうでなければ -1 を返す。

参照:

[mchar_encode\(\)](#)

2.12.3.5 mchar_encode()

```
unsigned mchar_encode (
    MSymbol charset_name,
    int c )
```

文字コードをエンコードする.

関数 `mchar_encode()` は、文字コード `c` をエンコードしてシンボル `charset_name` で示される文字セット内におけるコードポイントを得る。

戻り値:

エンコードが成功すれば、`mchar_encode()` はエンコードされたコードポイントを返す。 そうでなければ `MCHAR_INVALID_CODE` を返す。

参照:

`mchar_decode()`

2.12.3.6 mchar_map_charset()

```
int mchar_map_charset (
    MSymbol charset_name,
    void(*) (int from, int to, void *arg) func,
    void * func_arg )
```

指定した文字セットのすべての文字に対して関数を呼ぶ.

関数 `mcharset_map_chars()` は `charset_name` という名前を持つ文字セット中のすべての文字に対して `func` を呼ぶ。 呼び出しは一文字毎ではなく、連続した文字のまとまり単位で行なわれる。

関数 `func` には `from`, `to`, `arg` の 3 引数が渡される。`from` と `to` は `charset` 中の文字コードの範囲を指定する。`arg` は `func_arg` と同じである。

戻り値:

処理に成功すれば `mcharset_map_chars()` は 0 を返す。 そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_CHARSET`

2.12.4 変数詳解

2.12.4.1 Mcharset_ascii

[MSymbol](#) Mcharset_ascii

ASCII 文字セットを表現するシンボル.

シンボル [Mcharset_ascii](#) は "ascii" という名前を持ち、ISO 646, USA Version X3.4-1968 (ISO-IR-6) 文字セットを表現する。

2.12.4.2 Mcharset_iso.8859.1

[MSymbol](#) Mcharset_iso.8859.1

ISO/IEC 8859-1:1998 文字セットを表現するシンボル.

シンボル [Mcharset_iso.8859.1](#) は "iso-8859-1" という名前を持ち、ISO/IEC 8859-1:1998 文字セットを表現する。

2.12.4.3 Mcharset_unicode

[MSymbol](#) Mcharset_unicode

Unicode 文字セットを表現するシンボル.

シンボル [Mcharset_unicode](#) は "unicode" という名前を持ち、Unicode 文字セットを表現する。

2.12.4.4 Mcharset_m17n

[MSymbol](#) Mcharset_m17n

全文字を含む文字セットを表現するシンボル.

シンボル [Mcharset_m17n](#) は "m17n" という名前を持ち、m17n ライブラリが扱う全ての文字を含む文字セットを表現する。

2.12.4.5 Mcharset_binary

[MSymbol](#) Mcharset_binary

正しくデコードできない文字の文字セットを表現するシンボル.

シンボル [Mcharset_binary](#) は "binary" という名前を持ち、偽の (fake) 文字セットを表現する。デコード関数は、M-text のテキストプロパティとして、無効なバイト (シークエンス) に遭遇した位置を付加する。

詳細は [コード変換](#) 参照のこと。

2.12.4.6 Mmethod

`MSymbol` `Mmethod`

2.12.4.7 Mdimension

`MSymbol` `Mdimension`

2.12.4.8 Mmin_range

`MSymbol` `Mmin_range`

2.12.4.9 Mmax_range

`MSymbol` `Mmax_range`

2.12.4.10 Mmin_code

`MSymbol` `Mmin_code`

2.12.4.11 Mmax_code

`MSymbol` `Mmax_code`

2.12.4.12 Mascii_compatible

`MSymbol` `Mascii_compatible`

2.12.4.13 Mfinal_byte

[MSymbol](#) Mfinal_byte

2.12.4.14 Mrevision

[MSymbol](#) Mrevision

2.12.4.15 Mmin_char

[MSymbol](#) Mmin_char

2.12.4.16 Mmapfile

[MSymbol](#) Mmapfile

2.12.4.17 Mparents

[MSymbol](#) Mparents

2.12.4.18 Msubset_offset

[MSymbol](#) Msubset_offset

2.12.4.19 Mdefine_coding

[MSymbol](#) Mdefine_coding

2.12.4.20 Maliaes

[MSymbol](#) Maliaes

2.12.4.21 Moffset

[MSymbol](#) Moffset

@brief オフセット型のメソッドを示すシンボル.

シンボル `#Moffset` は `<tt>"offset"</tt>` という名前を持ち、文字セットの `@b Mmethod` パラメータの値として用いられた場合には、コードポイントと文字セットの文字コードの間の変換が以下の式に従って行われることを意味する。

文字コード = コードポイント - MIN-CODE + MIN-CHAR

ここで、MIN-CODE は文字セットの `@b Mmin_code` パラメータの値であり、MIN-CHAR は `@b Mmin_char` パラメータの値である。

2.12.4.22 Mmap

[MSymbol](#) Mmap

マップ型のメソッドを示すシンボル.

シンボル `Mmap` は `"map"` という名前を持ち、文字セットの `Mmethod` パラメータの値として用いられた場合には、コードポイントと文字セットの文字コードの間の変換がマップを参照することによって行われることを意味する。マップは `Mmapfile` パラメータとして与えなければならない。

2.12.4.23 Munify

[MSymbol](#) Munify

ユニファイ型のメソッドを示すシンボル.

シンボル `Munify` は `"unify"` という名前を持ち、文字セットの `Mmethod` パラメータの値として用いられた場合には、コードポイントと文字セットの文字コードの間の変換が、マップの参照とオフセットの組み合わせによって行われることを意味する。マップは `Mmapfile` パラメータとして与えなければならない。この種の各文字セットには、全文字に対して連続するコードスペースがそれぞれ割り当てられる。

コードポイントがマップに含まれていれば、変換はマップ参照によって行われる。そうでなければ、以下の式に従う。

CHARACTER-CODE = CODE-POINT - MIN-CODE + LOWEST-CHAR-CODE

ここで、MIN-CODE は文字セットの `@b Mmin_code` パラメータの値であり、LOWEST-CHAR-CODE は割り当てられたコードスペースの最も小さい文字コードである。

2.12.4.24 Msubset

`MSymbol` `Msubset`

サブセット型のメソッドを示すシンボル。

シンボル `Msubset` は "subset" という名前を持ち、文字セットの `Mmethod` パラメータの値として用いられた場合には、この文字セットが別の文字セット（親文字セット）の部分集合であることを意味する。親文字セットは `Mparents` パラメータによって与えられなくてはならない。コードポイントと文字セットの文字コードの間の変換は、概念的には以下の式に従う。

$$\text{CHARACTER-CODE} = \text{PARENT-CODE}(\text{CODE-POINT}) + \text{SUBSET-OFFSET}$$

ここで `PARENT-CODE` は `CODE-POINT` の親文字セット中での文字コードを返す擬関数であり、`SUBSET-OFFSET` は `@b Msubset_offset` パラメータで与えられる値である。

2.12.4.25 Msuperset

`MSymbol` `Msuperset`

スーパーセット型のメソッドを示すシンボル。

シンボル `Msuperset` は "superset" という名前を持ち、文字セットの `Mmethod` パラメータの値として用いられた場合には、この文字セットが別の文字セット（親文字セット）の上位集合であることを意味する。親文字セットは `Mparents` パラメータによって与えられなくてはならない。

2.12.4.26 Mcharset

`MSymbol` `Mcharset`

2.13 コード変換

コード系オブジェクトとそれに関する API。

コード変換 連携図



データ構造

- struct `MConverter`
コード変換に用いられる構造体.
- struct `MCodingInfoISO2022`
`MCODING_TYPE_ISO_2022` タイプのコード系に必要な付加情報用構造体.
- struct `MCodingInfoUTF`
`MCODING_TYPE_UTF` タイプのコード系に必要な付加情報用の構造体.

列挙型

- enum `MConversionResult` {
`MCONVERSION_RESULT_SUCCESS` ,
`MCONVERSION_RESULT_INVALID_BYTE` ,
`MCONVERSION_RESULT_INVALID_CHAR` ,
`MCONVERSION_RESULT_INSUFFICIENT_SRC` ,
`MCONVERSION_RESULT_INSUFFICIENT_DST` ,
`MCONVERSION_RESULT_IO_ERROR` }
コード変換の結果を示すコード.
- enum `MCodingType` {
`MCODING_TYPE_CHARSET` ,
`MCODING_TYPE_UTF` ,
`MCODING_TYPE_ISO_2022` ,
`MCODING_TYPE_MISC` }
コード系のタイプ.
- enum `MCodingFlagISO2022` {
`MCODING_ISO_RESET_AT_EOL` = 0x1 ,
`MCODING_ISO_RESET_AT_CNTL` = 0x2 ,
`MCODING_ISO_EIGHT_BIT` = 0x4 ,
`MCODING_ISO_LONG_FORM` = 0x8 ,
`MCODING_ISO_DESIGNATION_G0` = 0x10 ,
`MCODING_ISO_DESIGNATION_G1` = 0x20 ,
`MCODING_ISO_DESIGNATION_CTEXT` = 0x40 ,
`MCODING_ISO_DESIGNATION_CTEXT_EXT` = 0x80 ,
`MCODING_ISO_LOCKING_SHIFT` = 0x100 ,
`MCODING_ISO_SINGLE_SHIFT` = 0x200 ,
`MCODING_ISO_SINGLE_SHIFT_7` = 0x400 ,
`MCODING_ISO_EUC_TW_SHIFT` = 0x800 ,
`MCODING_ISO_ISO6429` = 0x1000 ,
`MCODING_ISO_REVISION_NUMBER` = 0x2000 ,
`MCODING_ISO_FULL_SUPPORT` = 0x3000 ,
`MCODING_ISO_FLAG_MAX` }
`MCODING_TYPE_ISO_2022` タイプのコード系の詳細を表わすビットマスク.

関数

- `MSymbol mconv_define_coding` (const char *name, `MPList` *plist, int(*resetter)(`MConverter` *), int(*decoder)(const unsigned char *, int, `MText` *, `MConverter` *), int(*encoder)(`MText` *, int, int, unsigned char *, int, `MConverter` *), void *extra_info)
- `MSymbol mconv_resolve_coding` (`MSymbol` symbol)
コード系の名前を解決する.

- `int mconv_list_codings (MSymbol **symbols)`
コード系を表わすシンボルを列挙する。
- `MConverter * mconv_buffer_converter (MSymbol name, const unsigned char *buf, int n)`
バッファに結び付けられたコードコンバータを作る。
- `MConverter * mconv_stream_converter (MSymbol name, FILE *fp)`
ストリームに結び付けられたコードコンバータを作る。
- `int mconv_reset_converter (MConverter *converter)`
コードコンバータをリセットする。
- `void mconv_free_converter (MConverter *converter)`
コードコンバータを解放する。
- `MConverter * mconv_rebind_buffer (MConverter *converter, const unsigned char *buf, int n)`
コードコンバータにバッファ領域を結び付ける。
- `MConverter * mconv_rebind_stream (MConverter *converter, FILE *fp)`
コードコンバータにストリームを結び付ける。
- `MText * mconv_decode (MConverter *converter, MText *mt)`
バイト列を M-text にデコードする。
- `MText * mconv_decode_buffer (MSymbol name, const unsigned char *buf, int n)`
コード系に基づいてバッファ領域をデコードする。
- `MText * mconv_decode_stream (MSymbol name, FILE *fp)`
コード系に基づいてストリーム入力をデコードする。
- `int mconv_encode (MConverter *converter, MText *mt)`
M-text をバイト列にエンコードする。
- `int mconv_encode_range (MConverter *converter, MText *mt, int from, int to)`
M-text の一部をバイト列にエンコードする。
- `int mconv_encode_buffer (MSymbol name, MText *mt, unsigned char *buf, int n)`
M-text をエンコードしてバッファ領域に書き込む。
- `int mconv_encode_stream (MSymbol name, MText *mt, FILE *fp)`
M-text をエンコードしてストリームに書き込む。
- `int mconv_getc (MConverter *converter)`
コードコンバータ経由で一文字を読みこむ。
- `int mconv_ungetc (MConverter *converter, int c)`
コードコンバータに一文字戻す。
- `int mconv_putc (MConverter *converter, int c)`
コードコンバータを経由して一文字書き出す。
- `MText * mconv_gets (MConverter *converter, MText *mt)`
コードコンバータを使って一行読み込む。

変数: 定義済みコード系を指定するためのシンボル

- `MSymbol Mcoding_us_ascii`
US-ASCII コード系のシンボル。
- `MSymbol Mcoding_iso_8859_1`
ISO-8859-1 コード系のシンボル。
- `MSymbol Mcoding_utf_8`
UTF-8 コード系のシンボル。
- `MSymbol Mcoding_utf_8_full`
UTF-8-FULL コード系のシンボル。
- `MSymbol Mcoding_utf_16`

- UTF-16 コード系のシンボル.
 - [MSymbol Mcoding_utf_16be](#)
- UTF-16BE コード系のシンボル.
 - [MSymbol Mcoding_utf_16le](#)
- UTF-16LE コード系のシンボル.
 - [MSymbol Mcoding_utf_32](#)
- UTF-32 コード系のシンボル.
 - [MSymbol Mcoding_utf_32be](#)
- UTF-32BE コード系のシンボル.
 - [MSymbol Mcoding_utf_32le](#)
- UTF-32LE コード系のシンボル.
 - [MSymbol Mcoding_sjis](#)
- SJIS コード系のシンボル.

変数: `mconv_define_coding()` 用パラメータキー

- [MSymbol Mtype](#)
- [MSymbol Mcharsets](#)
- [MSymbol Mflags](#)
- [MSymbol Mdesignation](#)
- [MSymbol Minvocation](#)
- [MSymbol Mcode_unit](#)
- [MSymbol Mbom](#)
- [MSymbol Mlittle_endian](#)

変数: コード系のタイプを示すシンボル.

- [MSymbol Mutf](#)
- [MSymbol Miso_2022](#)

変数: パラメータ **Mflags** の値となり得るシンボル.

関数 `mconv_define_coding()` の引数として用いられるコード系のパラメータ **Mflags** の値となり得るシンボル。(詳細は `mconv_define_coding()` 参照)。

- [MSymbol Mreset_at_eol](#)
- [MSymbol Mreset_at_cnl](#)
- [MSymbol Meight_bit](#)
- [MSymbol Mlong_form](#)
- [MSymbol Mdesignation_g0](#)
- [MSymbol Mdesignation_g1](#)
- [MSymbol Mdesignation_ctxt](#)
- [MSymbol Mdesignation_ctxt_ext](#)
- [MSymbol Mlocking_shift](#)
- [MSymbol Msingle_shift](#)
- [MSymbol Msingle_shift_7](#)
- [MSymbol Meuc_tw_shift](#)
- [MSymbol Miso_6429](#)
- [MSymbol Mrevision_number](#)
- [MSymbol Mfull_support](#)

変数: その他

ほかの変数。

- `MSymbol Mmaybe`
"maybe"という名前を持つシンボル.
- `MSymbol Mcoding`
シンボル `Mcoding`.

2.13.1 詳解

コード系オブジェクトとそれに関する API.

`m17n` ライブラリは、符号化文字集合 (coded character set; CCS) の文字符合化方式 (character encoding scheme; CES) をコード系と呼ぶオブジェクトで表現する。アプリケーションプログラムは独自にコード系を追加することもできる。

コードポイントから文字コードへの変換を エンコード と呼び、文字コードからコードポイントへの変換を デコード と呼ぶ。

アプリケーションプログラムは、指定されたコード系でバイト列をデコードすることによって `M-text` を得ることができる。また逆に、指定されたコード系で `M-text` をエンコードしすることによってバイト列を得ることができる。

2.13.2 列挙型詳解

2.13.2.1 MConversionResult

enum `MConversionResult`

コード変換の結果を示すコード.

これらの値のうち一つが `MConverter->result` に設定される。

列挙値

<code>MCONVERSION_RESULT_SUCCESS</code>	コード変換は成功.
<code>MCONVERSION_RESULT_INVALID_BYTE</code>	デコード時、ソースに不正なバイトが含まれている.
<code>MCONVERSION_RESULT_INVALID_CHAR</code>	エンコード時、指定のコード系でエンコードできない文字がソースに含まれている.
<code>MCONVERSION_RESULT_INSUFFICIENT_SRC</code>	デコード時、不完全なバイト列でソースが終わっている.
<code>MCONVERSION_RESULT_INSUFFICIENT_DST</code>	エンコード時、結果を格納する領域が短かすぎる.
<code>MCONVERSION_RESULT_IO_ERROR</code>	コード変換中に I/O エラーが起こった.

2.13.2.2 MCodingType

enum [MCodingType](#)

コード系のタイプ.

列挙値

MCODING_TYPE_CHARSET	このタイプのコード系は文字セットを直接サポートする。各文字セットの次元とは、その文字セットで一文字を表現するために必要なバイト数であり、バイト列は文字のコードポイントを直接表す。m17n ライブラリはこのタイプ用のデフォルトのエンコード／デコードルーティンを提供する。
MCODING_TYPE_UTF	このタイプのコード系は、UTF 系 (UTF-8, UTF-16, UTF-32) のバイト列をサポートする。m17n ライブラリはこのタイプ用のデフォルトのエンコード／デコードルーティンを提供する。
MCODING_TYPE_ISO_2022	このタイプのコード系は、ISO-2022 系のバイト列をサポートする。 各コード系の構造の詳細は MCodingInfoISO2022 で指定される。m17n ライブラリはこのタイプ用のデフォルトのエンコード／デコードルーティンを提供する。
MCODING_TYPE_MISC	このタイプのコード系は、その他の構造のバイト列のためのものである。m17n ライブラリはこのタイプ用のエンコード／デコードルーティンを提供しないので、アプリケーションプログラム側で準備する必要がある。

2.13.2.3 MCodingFlagISO2022

enum [MCodingFlagISO2022](#)

MCODING_TYPE_ISO_2022 タイプのコード系の詳細を表わすビットマスク.

列挙値

MCODING_ISO_RESET_AT_EOL	エンコードの際、行末で呼び出し (invocation) と指示 (designation) の状態を初期値に戻す。
MCODING_ISO_RESET_AT_CNTL	エンコードの際、すべての制御文字の前で、呼び出し (invocation) と指示 (designation) の状態を初期値に戻す。
MCODING_ISO_EIGHT_BIT	図形文字集合の右側を使う。
MCODING_ISO_LONG_FORM	JISX0208-1978, GB2312, JISX0208-1983 の文字集合に対する指示シークエンスとして、非標準の 4 バイト形式を用いる。

列挙値

MCODING_ISO_DESIGNATION_G0	エンコードの際、特に指定されない限り、文字集合を G0 に指示する。
MCODING_ISO_DESIGNATION_G1	エンコードの際、特に指定されない限り、ASCII 以外の文字集合を G1 に指示する。
MCODING_ISO_DESIGNATION_CTEXT	エンコードの際、特に指定されない限り、94 文字集合を G0 に、96 文字集合を G1 に指示する。
MCODING_ISO_DESIGNATION_CTEXT_EXT	エンコードの際、ISO-2022 に合致しない文字集合を ESC % / ... でエンコードする。サポートされていない Unicode 文字は ESC % G ... ESC % @ でエンコードする。デコードの際、これらのエスケープ・シーケンスを解釈する。
MCODING_ISO_LOCKING_SHIFT	ロッキングシフトを使う。
MCODING_ISO_SINGLE_SHIFT	シングルシフト (SS2 (0x8E or ESC N), SS3 (0x8F or ESC O)) を使う。
MCODING_ISO_SINGLE_SHIFT_7	7 ビットシングルシフト 2 (SS2 (0x19)) を使う。
MCODING_ISO_EUC_TW_SHIFT	EUC-TW 風の特別なシフトを使う。
MCODING_ISO_ISO6429	ISO-6429 のエスケープシーケンスで方向を指示する。未実装。
MCODING_ISO_REVISION_NUMBER	エンコードの際、文字セットに revision number があればそれを表わすエスケープシーケンスを生成する。
MCODING_ISO_FULL_SUPPORT	ISO-2022 の全文字集合をサポートする。
MCODING_ISO_FLAG_MAX	

2.13.3 関数詳解

2.13.3.1 mconv_define_coding()

```
MSymbol mconv_define_coding (
    const char * name,
    MPlist * plist,
    int(*) (MConverter *) resetter,
    int(*) (const unsigned char *, int, MText *, MConverter *) decoder,
    int(*) (MText *, int, int, unsigned char *, int, MConverter *) encoder,
    void * extra_info )
```

2.13.3.2 mconv_resolve_coding()

```
MSymbol mconv_resolve_coding (
    MSymbol symbol )
```

コード系の名前を解決する。

関数 `mconv_resolve_coding()` は `symbol` がコード系を示していればそれを返す。そうでなければコード系の名前として `symbol` を正規化し、それがコード系を表していれば正規化した `symbol` を返す。そうでなければ `::Mnil` を返す。

2.13.3.3 mconv_list_codings()

```
int mconv_list_codings (
    MSymbol ** symbols )
```

コード系を表わすシンボルを列挙する。

関数 `mchar_list_codings()` は、コード系を示すシンボルを並べた配列を作り、`symbols` でポイントされた場所にこの配列へのポインタを置き、配列の長さを返す。

2.13.3.4 mconv_buffer_converter()

```
MConverter* mconv_buffer_converter (
    MSymbol name,
    const unsigned char * buf,
    int n )
```

バッファに結び付けられたコードコンバータを作る。

関数 `mconv_buffer_converter()` は、コード系 `name` 用のコードコンバータを作る。このコードコンバータは、`buf` で示される大きさ `n` バイトのバッファ領域に結び付けられる。これ以降のデコードおよびエンコードは、このバッファ領域に対して行なわれる。

`name` は `Mnil` であってもよい。この場合は現在のロケール (`LC_CTYPE`) に関連付けられたコード系が使われる。

戻り値:

もし処理が成功すれば `mconv_buffer_converter()` は作成したコードコンバータを返す。そうでなければ `NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_SYMBOL, MERROR_CODING`

参照:

[mconv_stream_converter\(\)](#)

2.13.3.5 mconv_stream_converter()

```
MConverter* mconv_stream_converter (
    MSymbol name,
    FILE * fp )
```

ストリームに結び付けられたコードコンバータを作る。

関数 `mconv_stream_converter()` は、コード系 `name` 用のコードコンバータを作る。このコードコンバータは、ストリーム `fp` に結び付けられる。これ以降のデコードおよびエンコードは、このストリームに対して行なわれる。

`name` は `Mnil` であってもよい。この場合は現在のロケール (`LC_CTYPE`) に関連付けられたコード系が使われる。

戻り値:

もし処理が成功すれば、`mconv_stream_converter()` は作成したコードコンバータを返す。そうでなければ `NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_SYMBOL, MERROR_CODING`

参照:

[mconv_buffer_converter\(\)](#)

2.13.3.6 mconv_reset_converter()

```
int mconv_reset_converter (
    MConverter * converter )
```

コードコンバータをリセットする。

関数 `mconv_reset_converter()` はコードコンバータ `converter` を初期状態に戻す。

戻り値:

もし `converter->coding` にリセット用の関数が定義されているならば、`mconv_reset_converter()` はその関数に `converter` を適用した結果を返し、そうでなければ `0` を返す。

2.13.3.7 mconv_free_converter()

```
void mconv_free_converter (
    MConverter * converter )
```

コードコンバータを解放する。

関数 `mconv_free_converter()` はコードコンバータ `converter` を解放する。

2.13.3.8 `mconv_rebind_buffer()`

```
MConverter* mconv_rebind_buffer (
    MConverter * converter,
    const unsigned char * buf,
    int n )
```

コードコンバータにバッファ領域を結び付ける。

関数 `mconv_rebind_buffer()` は、`buf` によって指された大きさ `n` バイトのバッファ領域をコードコンバータ `converter` に結び付ける。これ以降のデコードおよびエンコードは、この新たに結び付けられたバッファ領域に対して行なわれるようになる。

戻り値:

この関数は常に `converter` を返す。

参照:

[`mconv_rebind_stream\(\)`](#)

2.13.3.9 `mconv_rebind_stream()`

```
MConverter* mconv_rebind_stream (
    MConverter * converter,
    FILE * fp )
```

コードコンバータにストリームを結び付ける。

関数 `mconv_rebind_stream()` は、ストリーム `fp` をコードコンバータ `converter` に結び付ける。これ以降のデコードおよびエンコードは、この新たに結び付けられたストリームに対して行なわれるようになる。

戻り値:

この関数は常に `converter` を返す。

参照:

[`mconv_rebind_buffer\(\)`](#)

2.13.3.10 mconv_decode()

```
MText* mconv_decode (
    MConverter * converter,
    MText * mt )
```

バイト列を M-text にデコードする。

関数 `mconv_decode()` は、バイト列をデコードしてその結果を M-text `mt` の末尾に追加する。デコード元のバイト列は、`converter` に現在結び付けられているバッファ領域あるいはストリームから取られる。

戻り値:

もし処理が成功すれば、`mconv_decode()` は更新された `mt` を返す。そうでなければ `NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_IO`, `MERROR_CODING`

参照:

`mconv_rebind_buffer()`, `mconv_rebind_stream()`, `mconv_encode()`, `mconv_encode_range()`,
`mconv_decode_buffer()`, `mconv_decode_stream()`

2.13.3.11 mconv_decode_buffer()

```
MText* mconv_decode_buffer (
    MSymbol name,
    const unsigned char * buf,
    int n )
```

コード系に基づいてバッファ領域をデコードする。

関数 `mconv_decode_buffer()` は、`buf` によって指された `n` バイトのバッファ領域を、コード系 `name` に基づいてデコードする。デコードに必要なコードコンバータの作成と解放は自動的に行なわれる。

戻り値:

もし処理が成功すれば、`mconv_decode_buffer()` は得られた M-text を返す。そうでなければ `NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_IO`, `MERROR_CODING`

参照:

`mconv_decode()`, `mconv_decode_stream()`

2.13.3.12 mconv_decode_stream()

```
MText* mconv_decode_stream (
    MSymbol name,
    FILE * fp )
```

コード系に基づいてストリーム入力をデコードする。

関数 `mconv_decode_stream()` は、ストリーム `fp` から読み込まれるバイト列全体を、コード系 `name` に基づいてデコードする。デコードに必要なコードコンバータの作成と解放は自動的に行なわれる。

戻り値:

もし処理が成功すれば、`mconv_decode_stream()` は得られた M-text を返す。そうでなければ NULL を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_IO, MERROR_CODING

参照:

`mconv_decode()`, `mconv_decode_buffer()`

2.13.3.13 mconv_encode()

```
int mconv_encode (
    MConverter * converter,
    MText * mt )
```

M-text をバイト列にエンコードする。

関数 `mconv_encode()` は、M-text `mt` をエンコードして、コードコンバータ `converter` に現在結び付けられているバッファ領域あるいはストリームに得られたバイト列を書き込む。

戻り値:

もし処理が成功すれば、`mconv_encode()` は書き込まれたバイト数を返す。そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_IO, MERROR_CODING

参照:

`mconv_rebind_buffer()`, `mconv_rebind_stream()`, `mconv_decode()`, `mconv_encode_range()`

2.13.3.14 mconv_encode_range()

```
int mconv_encode_range (
    MConverter * converter,
    MText * mt,
    int from,
    int to )
```

M-text の一部をバイト列にエンコードする。

関数 [mconv_encode_range\(\)](#) は、M-text *mt* の *from* (*from* 自体も含む) から *to* (*to* 自体は含まない) までの範囲のテキストをエンコードして、コードコンバータ *converter* に現在結び付けられているバッファ領域あるいはストリームに得られたバイト列を書き込む。

戻り値:

もし処理が成功すれば、[mconv_encode_range\(\)](#) は書き込まれたバイト数を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

[MERROR_RANGE](#), [MERROR_IO](#), [MERROR_CODING](#)

参照:

[mconv_rebind_buffer\(\)](#), [mconv_rebind_stream\(\)](#), [mconv_decode\(\)](#), [mconv_encode\(\)](#)

2.13.3.15 mconv_encode_buffer()

```
int mconv_encode_buffer (
    MSymbol name,
    MText * mt,
    unsigned char * buf,
    int n )
```

M-text をエンコードしてバッファ領域に書き込む。

関数 [mconv_encode_buffer\(\)](#) は M-text *mt* をコード系 *name* に基づいてエンコードし、得られたバイト列を *buf* の指すバッファ領域に書き込む。 *n* は書き込む最大バイト数である。エンコードに必要なコードコンバータの作成と解放は自動的に行なわれる。

戻り値:

もし処理が成功すれば、[mconv_encode_buffer\(\)](#) は書き込まれたバイト数を返す。そうでなければ -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

エラー:

[MERROR_IO](#), [MERROR_CODING](#)

参照:

[mconv_encode\(\)](#), [mconv_encode_stream\(\)](#)

2.13.3.16 mconv_encode_stream()

```
int mconv_encode_stream (
    MSymbol name,
    MText * mt,
    FILE * fp )
```

M-text をエンコードしてストリームに書き込む。

関数 `mconv_encode_stream()` はM-text `mt` をコード系 `name` に基づいてエンコードし、得られたバイト列をストリーム `fp` に書き出す。エンコードに必要なコードコンバータの作成と解放は自動的に行なわれる。

戻り値:

もし処理が成功すれば、`mconv_encode_stream()` は書き込まれたバイト数を返す。そうでなければ -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_IO`, `MERROR_CODING`

参照:

`mconv_encode()`, `mconv_encode_buffer()`, `mconv_encode_file()`

2.13.3.17 mconv_getc()

```
int mconv_getc (
    MConverter * converter )
```

コードコンバータ経由で一文字を読みこむ。

関数 `mconv_getc()` は、コードコンバータ `converter` に現在結び付けられているバッファ領域あるいはストリームから文字を一つ読み込む。バイト列のデコードには `converter` のデコーダが用いられる。 `converter` の内部状態は必要に応じて更新される。

戻り値:

処理が成功すれば、`mconv_getc()` は読み込まれた文字を返す。入力源が EOF に達した場合は、外部変数 `merror_code` を変えずに EOF を返す。エラーが検出された場合は EOF を返し、`::merror_code` にエラーコードを設定する。

エラー:

`MERROR_CODING`

参照:

`mconv_ungetc()`, `mconv_putc()`, `mconv_gets()`

2.13.3.18 mconv_ungetc()

```
int mconv_ungetc (
    MConverter * converter,
    int c )
```

コードコンバータに一文字戻す。

関数 `mconv_ungetc()` は、コードコンバータ `converter` に文字 `c` を押し戻す。戻される文字数に制限はない。この後で `mconv_getc()` を呼び出した際には、最後に戻された文字が最初に読まれる。戻された文字は `converter` の内部に蓄えられるだけであり、実際に入力源に書き込まれるわけではない。 `converter` の内部状態は必要に応じて更新される。

戻り値:

処理が成功すれば、`mconv_ungetc()` は `c` を返す。そうでなければ EOF を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_CODING, MERROR_CHAR

参照:

[mconv_getc\(\)](#), [mconv_putc\(\)](#), [mconv_gets\(\)](#)

2.13.3.19 mconv_putc()

```
int mconv_putc (
    MConverter * converter,
    int c )
```

コードコンバータを経由して一文字書き出す。

関数 `mconv_putc()` は、コードコンバータ `converter` に現在結び付けられているバッファ領域あるいはストリームに文字 `c` を書き出す。文字のエンコードには `converter` のエンコーダが用いられる。実際に書き出されたバイト数は、`converter` のメンバー `nbytes` にセットされる。 `converter` の内部状態は必要に応じて更新される。

戻り値:

処理が成功すれば、`mconv_putc()` は `c` を返す。エラーが検出された場合は EOF を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

MERROR_CODING, MERROR_IO, MERROR_CHAR

参照:

[mconv_getc\(\)](#), [mconv_ungetc\(\)](#), [mconv_gets\(\)](#)

2.13.3.20 mconv_gets()

```
MText* mconv_gets (
    MConverter * converter,
    MText * mt )
```

コードコンバータを使って一行読み込む。

関数 `mconv_gets()` は、コードコンバータ `converter` に現在結び付けられているバッファ領域あるいはストリームから 1 行を読み込む。バイト列のデコードには `converter` のデコーダが用いられる。デコードされた文字列は M-text `mt` の末尾に追加される。元のバイト列の終端改行文字は追加されない。`converter` の内部状態は必要に応じて更新される。

戻り値:

処理が成功すれば、`mconv_gets()` は変更された `mt` を返す。もし 1 文字も読まずに EOF に遭遇した場合は、`mt` を変更せずにそのまま返す。エラーが検出された場合は NULL を返し、`merror_code` にエラーコードを設定する。

エラー:

MERROR_CODING

参照:

`mconv_getc()`, `mconv_ungetc()`, `mconv_putc()`

2.13.4 変数詳解

2.13.4.1 Mcoding_us_ascii

MSymbol Mcoding_us_ascii

US-ASCII コード系のシンボル。

シンボル `Mcoding_us_ascii` は "us-ascii" という名前を持ち、CES US-ASCII 用のコード系を示す。

2.13.4.2 Mcoding_iso_8859_1

MSymbol Mcoding_iso_8859_1

ISO-8859-1 コード系のシンボル。

シンボル `Mcoding_iso_8859_1` は "iso-8859-1" という名前を持ち、CES ISO-8859-1 用のコード系を示す。

2.13.4.3 Mcoding_utf_8

MSymbol Mcoding_utf_8

UTF-8 コード系のシンボル.

シンボル **Mcoding_utf_8** は "utf-8" という名前を持ち、CES UTF-8 用のコード系を示す。

2.13.4.4 Mcoding_utf_8_full

MSymbol Mcoding_utf_8_full

UTF-8-FULL コード系のシンボル.

シンボル **Mcoding_utf_8_full** は "utf-8-full" という名前を持ち、"UTF-8" の拡張であるコード系を示す。このコード系は UTF-8 と同じエンコーディングアルゴリズムを用いるが、対象は Unicode 文字には限定されない。また m17n ライブラリが扱う全ての文字をエンコードすることができる。

2.13.4.5 Mcoding_utf_16

MSymbol Mcoding_utf_16

UTF-16 コード系のシンボル.

シンボル **Mcoding_utf_16** は "utf-16" という名前を持ち、CES UTF-16 (RFC 2279) 用のコード系を示す。

2.13.4.6 Mcoding_utf_16be

MSymbol Mcoding_utf_16be

UTF-16BE コード系のシンボル.

シンボル **Mcoding_utf_16be** は "utf-16be" という名前を持ち、CES UTF-16BE (RFC 2279) 用のコード系を示す。

2.13.4.7 Mcoding_utf_16le

MSymbol Mcoding_utf_16le

UTF-16LE コード系のシンボル.

シンボル **Mcoding_utf_16le** は "utf-16le" という名前を持ち、CES UTF-16LE (RFC 2279) 用のコード系を示す。

2.13.4.8 Mcoding_utf_32

[MSymbol](#) `Mcoding_utf_32`

UTF-32 コード系のシンボル.

シンボル [Mcoding_utf_32](#) は "utf-32" という名前を持ち、CES UTF-32 (RFC 2279) 用のコード系を示す。

2.13.4.9 Mcoding_utf_32be

[MSymbol](#) `Mcoding_utf_32be`

UTF-32BE コード系のシンボル.

シンボル [Mcoding_utf_32be](#) は "utf-32be" という名前を持ち、CES UTF-32BE (RFC 2279) 用のコード系を示す。

2.13.4.10 Mcoding_utf_32le

[MSymbol](#) `Mcoding_utf_32le`

UTF-32LE コード系のシンボル.

シンボル [Mcoding_utf_32le](#) は "utf-32le" という名前を持ち、CES UTF-32LE (RFC 2279) 用のコード系を示す。

2.13.4.11 Mcoding_sjis

[MSymbol](#) `Mcoding_sjis`

SJIS コード系のシンボル.

シンボル [Mcoding_sjis](#) は "sjis" という名前を持ち、CES Shift-JIS 用のコード系を示す。

2.13.4.12 Mtype

[MSymbol](#) `Mtype`

[mconv_define_coding\(\)](#) 用パラメータキー (詳細は [mconv_define_coding\(\)](#)参照).

2.13.4.13 Mcharsets

[MSymbol](#) `Mcharsets`

2.13.4.14 Mflags

[MSymbol](#) Mflags

2.13.4.15 Mdesignation

[MSymbol](#) Mdesignation

2.13.4.16 Minvocation

[MSymbol](#) Minvocation

2.13.4.17 Mcode_unit

[MSymbol](#) Mcode_unit

2.13.4.18 Mbom

[MSymbol](#) Mbom

2.13.4.19 Mlittle_endian

[MSymbol](#) Mlittle_endian

2.13.4.20 Mutf

[MSymbol](#) Mutf

関数 [mconv_define_coding\(\)](#) の引数として用いられるコード系のパラメータ [Mtype](#) の値となり得るシンボル。(詳細は [mconv_define_coding\(\)](#)参照)。

2.13.4.21 Miso_2022

`MSymbol Miso_2022`

2.13.4.22 Mreset_at_eol

`MSymbol Mreset_at_eol`

2.13.4.23 Mreset_at_cntl

`MSymbol Mreset_at_cntl`

2.13.4.24 Meight_bit

`MSymbol Meight_bit`

2.13.4.25 Mlong_form

`MSymbol Mlong_form`

2.13.4.26 Mdesignation_g0

`MSymbol Mdesignation_g0`

2.13.4.27 Mdesignation_g1

`MSymbol Mdesignation_g1`

2.13.4.28 Mdesignation_ctxt

[MSymbol](#) Mdesignation_ctxt

2.13.4.29 Mdesignation_ctxt_ext

[MSymbol](#) Mdesignation_ctxt_ext

2.13.4.30 Mlocking_shift

[MSymbol](#) Mlocking_shift

2.13.4.31 Msingle_shift

[MSymbol](#) Msingle_shift

2.13.4.32 Msingle_shift_7

[MSymbol](#) Msingle_shift_7

2.13.4.33 Meuc_tw_shift

[MSymbol](#) Meuc_tw_shift

2.13.4.34 Miso_6429

[MSymbol](#) Miso_6429

2.13.4.35 Mrevision_number

`MSymbol` Mrevision_number

2.13.4.36 Mfull_support

`MSymbol` Mfull_support

2.13.4.37 Mmaybe

`MSymbol` Mmaybe

"maybe"という名前を持つシンボル.

変数 `Mmaybe` は "maybe" という名前を持つ。これは関数 `mconv_define_coding()` パラメータ `Mbom` の値として用いられる。(詳細は `mconv_define_coding()` 参照)。

2.13.4.38 Mcoding

`MSymbol` Mcoding

シンボル Mcoding.

デコードされた **M-text** はすべて、キーが定義済みシンボル Mcoding であるようなテキストプロパティを持つ。シンボル Mcoding は "coding" という名前を持つ。

2.14 ロケール

ロケールオブジェクトとそれに関する API.

ロケール 連携図



型定義

- typedef struct `MLocale` `MLocale`
`MLocale` 構造体.

関数

- [MPlist * mlanguage_jlist](#) (void)
3 文字言語コードをリストする.
- [MSymbol mlanguage_code](#) (MSymbol language, int len)
言語コードを得る.
- [MPlist * mlanguage_name_list](#) (MSymbol language, MSymbol target, MSymbol script, MSymbol territory)
- [MText * mlanguage_text](#) (MSymbol language)
与えられた言語自身で書かれた言語名を返す.
- [MPlist * mscript_list](#) (void)
スクリプト名をリストする.
- [MPlist * mscript_language_list](#) (MSymbol script)
与えられたスクリプトを用いる言語をリストする.
- [MLocale * mlocale_set](#) (int category, const char *name)
現在のロケールを設定する.
- [MSymbol mlocale_get_prop](#) (MLocale *locale, MSymbol key)
ロケールプロパティの値を得る.
- [int mtextftime](#) (MText *mt, const char *format, const struct tm *tm, MLocale *locale)
日付と時間をフォーマットする.
- [MText * mtext_getenv](#) (const char *name)
環境変数を得る.
- [int mtext_putenv](#) (MText *mt)
環境変数を変更／追加する.
- [int mtext_coll](#) (MText *mt1, MText *mt2)
現在のロケールを用いて 2 つの M-text を比較する.

変数

- [MSymbol Miso639_1](#)
- [MSymbol Miso639_2](#)
- [MSymbol Mterritory](#)
- [MSymbol Mmodifier](#)
- [MSymbol Mcodeset](#)

2.14.1 詳解

ロケールオブジェクトとそれに関する API.

m17n ライブラリはロケール関連情報を [MLocale](#) 型のオブジェクトで表現する。

2.14.2 型定義詳解

2.14.2.1 MLocale

```
typedef struct MLocale MLocale
```

MLocale 構造体.

MLocale 構造体は、ロケールの名前、言語、地域、モディファイア、コードセット、および対応するコード系に関する情報を保持するために用いられる。

この構造体の内容は実装に依存する。内部構造はアプリケーションプログラムからは見えない。

参照:

[mlocale_get_prop\(\)](#)

2.14.3 関数詳解

2.14.3.1 mlanguage_list()

```
MPList* mlanguage_list (  
    void )
```

3 文字言語コードをリストする.

関数 [mlanguage_list\(\)](#) は、整形式 (well-formed) plist を返す。各キーは [Msymbol](#) であり、個々の値は ISO639-2 に定められた 3 文字言語コードを名前とするシンボルである。

戻り値:

この関数が返す plist は、呼び出し側が [m17n_object_unref\(\)](#) を使って解放する必要がある。

参照:

[mscript_list\(\)](#).

2.14.3.2 mlanguage_code()

```
MSymbol mlanguage_code (  
    MSymbol language,  
    int len )
```

言語コードを得る.

関数 [mlanguage_code\(\)](#) は、`language` に対応した ISO-639 言語コードが名前であるようなシンボルを返す。`language` はシンボルであり、その名前は、ISO639-2 3 文字言語コード、ISO639-1 2 文字言語コード、英語名、のいずれかである。

`len` は返される言語コードの種類を決定する。`len` が 3 の場合は ISO639-2 3 文字言語コードが返される。2 の場合は、もし定義されていれば ISO639-1 2 文字言語コードが、そうでなければ [Mnil](#) が返される。0 の場合は、もし定義されていれば 2 文字コードが、そうでなければ 3 文字コードが返される。

戻り値:

もし情報が得られれば、この関数は [Mnil](#) 以外のシンボルを返す。そうでなければ [Mnil](#) を返す。

参照:

[mlanguage_name\(\)](#), [mlanguage_text\(\)](#).

2.14.3.3 mlanguage_name_list()

```
MPList* mlanguage_name_list (
    MSymbol language,
    MSymbol target,
    MSymbol script,
    MSymbol territory )
```

2.14.3.4 mlanguage_text()

```
MText* mlanguage_text (
    MSymbol language )
```

与えられた言語自身で書かれた言語名を返す。

関数 `mlanguage_text()` は、言語 `language` で書かれた `language` の名前を M-text の形式で返す。その言語の代表的な文字がわかっている場合は、返される M-text の各文字に、キーが `Mtext` で値がその代表的な文字を含む M-text であるようなテキストプロパティが付加される。

戻り値:

求める情報が得られた場合、この関数が返す M-text を変更したり解放したりしてはいけない。情報が得られなかった場合は NULL が返される。

参照:

`mlanguage_code()`, `mlanguage_name()`.

2.14.3.5 mscript_list()

```
MPList* mscript_list (
    void )
```

スクリプト名をリストする。

関数 `mscript_list()` は、整形式 (well-formed) plist を返す。各キーは `Msymbol` であり、個々の値はスクリプト名を名前とするシンボルである。

戻り値:

この関数が返す plist は、呼び出し側が `m17n_object_unref()` を使って解放する必要がある。

参照:

`mscript_language_list()`, `mlanguage_list()`.

2.14.3.6 mscript_language_list()

```
MPList* mscript_language_list (
    MSymbol script )
```

与えられたスクリプトを用いる言語をリストする。

関数 [mscript_language_list\(\)](#) は、`script` を用いる言語をリストする。`script` はシンボルで、その名前は Unicode Character Database に示されているスクリプト名をすべて小文字にしたものである。

戻り値:

この関数は、整形式 (well-formed) plist を返す。各キーは [MSymbol](#) であり、個々の値は ISO639-1 に定められた 2 文字言語コード (定義されていない場合は ISO639-2 に定められた 3 文字言語コード) を名前とするシンボルである。返される plist は変更したり解放したりしてはならない。`script` が未知の場合は NULL が返される。

参照:

[mscript_list\(\)](#), [mlanguage_list\(\)](#).

2.14.3.7 mlocale_set()

```
MLocale* mlocale_set (
    int category,
    const char * name )
```

現在のロケールを設定する。

関数 [mlocale_set\(\)](#) は現在のロケールの一部を設定したり問い合わせたりする。ここで一部とは `category` で指定され、`setlocale()` の有効な第一引数となるものでなくてはならない。

`locale` が NULL でなければ、指定した部分のロケールが `locale` に設定される。`locale` がシステムにサポートされていない場合は、設定は行われず、現在のロケールは変わらない。

`locale` が NULL ならば、現在のロケールの指定した部分を問い合わせる。

戻り値:

呼び出しに成功すれば、[mlocale_set\(\)](#) はロケールに対応する opaque ロケールオブジェクトを返す。ロケールの名前は関数 [mlocale_get_prop\(\)](#) によって得ることができる。そうでなければ NULL を返す。

エラー:

`MERROR_LOCALE`

2.14.3.8 mlocale_get_prop()

```
MSymbol mlocale_get_prop (
    MLocale * locale,
    MSymbol key )
```

ロケールプロパティの値を得る。

関数 `mlocale_get_prop()` は、ロケール `locale` の `key` プロパティの値を返す。`key` は `Mname`, `Mlanguage`, `Mterritory`, `Mcodeset`, `Mmodifier`, `Mcoding` のいずれかである。

2.14.3.9 mtext_ftime()

```
int mtext_ftime (
    MText * mt,
    const char * format,
    const struct tm * tm,
    MLocale * locale )
```

日付と時間をフォーマットする。

関数 `mtext_ftime()` は時刻データ (broken-down time) `tm` を `format` で指定された形式に清書し、結果をM-text `mt` に付加する。フォーマットは NULL でなければロケール `locale` に、または現在のロケール (LC_TIME) に従う。

引数 `tm` と `format` の意味は `strftime()` の場合と同じ。

参照:

`strftime()`。

2.14.3.10 mtext_getenv()

```
MText* mtext_getenv (
    const char * name )
```

環境変数を得る。

関数 `mtext_getenv()` は `name` で指される文字列と合致する文字列を環境変数のリストの中から探す。

見つかった場合には、その値を現在のロケール (LC_CTYPE) に従って M-text にデコードし、そのM-text を返す。

見つからなければ、NULL を返す。

2.14.3.11 mtext_putenv()

```
int mtext_putenv (
    MText * mt )
```

環境変数を変更／追加する。

関数 `mtext_putenv()` は M-text `mt` に従って、環境変数の値を変更したり追加したりする。この関数は、現在のロケール (LC_CTYPE) に従って `mt` をエンコードし、それを引数として関数 `putenv` を呼ぶ。

戻り値:

この関数は、成功した場合には 0 を、エラーが起これば -1 を返す。

2.14.3.12 mtext_coll()

```
int mtext_coll (
    MText * mt1,
    MText * mt2 )
```

現在のロケールを用いて 2 つの M-text を比較する。

関数 `mtext_coll()` は 2 つの M-text `mt1` と `mt2` を比較する。戻り値は負の整数値, 0, 正の整数値のいずれかであり、それぞれ `mt1` が `mt2` より小さい、同じ、大きい場合に相当する。比較は現在のロケール (LC_COLLATE) に基づいて行われる。

この関数は M-text のテキストプロパティとして自動的にキャッシュされる情報を利用するので、2 度目以降の同じ比較は 1 度目より速く実行される。

2.14.4 変数詳解

2.14.4.1 Miso639_1

`MSymbol Miso639_1`

2.14.4.2 Miso639_2

`MSymbol Miso639_2`

2.14.4.3 Mterritory

`MSymbol` Mterritory

"territory" という名前を持つシンボル.

2.14.4.4 Mmodifier

`MSymbol` Mmodifier

"modifier" という名前を持つシンボル.

2.14.4.5 Mcodeset

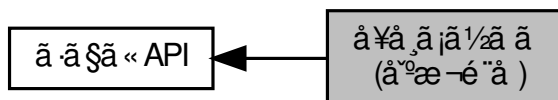
`MSymbol` Mcodeset

"codeset" という名前を持つシンボル.

2.15 入力メソッド (基本部分)

入力メソッド用API.

入力メソッド (基本部分) 連携図



データ構造

- struct `MInputDriver`
入力ドライバ用構造体.
- struct `MInputMethod`
入力メソッドの構造体.
- struct `MInputContext`
入力コンテキスト用構造体.

型定義

- typedef void(* `MInputCallbackFunc`) (`MInputContext` *ic, `MSymbol` command)
入力メソッドコールバック関数の型宣言.

列挙型

- `enum MinputCandidatesChanged {`
`MINPUT_CANDIDATES_LIST_CHANGED = 1,`
`MINPUT_CANDIDATES_INDEX_CHANGED = 2,`
`MINPUT_CANDIDATES_SHOW_CHANGED = 4,`
`MINPUT_CANDIDATES_CHANGED_MAX }`

入力メソッドの入力候補がどう変更されたかを示すビットマスク。

変数

- `MSymbol Minput_method`
“input-method” を名前として持つシンボル。
- `MinputDriver minput_default_driver`
内部入力メソッド用デフォルトドライバ。
- `MinputDriver * minput_driver`
内部入力メソッド用ドライバ。
- `MSymbol Minput_driver`

変数： コールバックコマンド用定義済みシンボル。

入力メソッドドライバのコールバック関数において `COMMAND` 引数として用いられる定義済みシンボル (`MinputDriver::callbackJist` 参照)。

ほとんどは追加の引数を必要としないし値を返さないが、以下は例外である。

`Minput_get_surrounding_text`: このコマンドに割り当てられたコールバック関数が呼ばれた際には、`MinputContext::plist` の第一要素はキーとして `::Minteger` をとり、その値はサラウンディングテキストのうちの部分を取って来るかを指定する。値が正であれば、現在のカーソル位置に続く値の個数分の文字を取る。負であれば、カーソル位置に先行する値の絶対値分の文字を取る。現在サラウンドテキストがサポートされているかどうかを知りたいだけであれば、この値はゼロでも良い。

サラウンディングテキストがサポートされていれば、コールバック関数はこの要素のキーを `Mtext` に、値を取り込んだ `M-text` に設定しなくてはならない。もしテキストの長さが充分でなければ、この `M-text` の長さは要求されている文字数より短くて良い。最悪の場合 0 でもよいし、アプリケーション側で必要で効率的だと思えば長くて良い。

サラウンディングテキストがサポートされていなければ、コールバック関数は `MinputContext::plist` の第一要素を変更してはならない。

`Minput_delete_surrounding_text`: このコマンドに割り当てられたコールバック関数が呼ばれた際には、`::MinputContext::plist` の第一要素は、キーとして `::Minteger` をとり、値は削除すべきサラウンディングテキストを `Minput_get_surrounding_text` と同様のやり方で指定する。コールバック関数は指定されたテキストを削除しなければならない。また `MinputContext::plist` を変えてはならない。

- `MSymbol Minput_preedit_start`
- `MSymbol Minput_preedit_done`
- `MSymbol Minput_preedit_draw`
- `MSymbol Minput_status_start`
- `MSymbol Minput_status_done`
- `MSymbol Minput_status_draw`

- `MSymbol Minput_candidates_start`
- `MSymbol Minput_candidates_done`
- `MSymbol Minput_candidates_draw`
- `MSymbol Minput_set_spot`
- `MSymbol Minput_toggle`
- `MSymbol Minput_reset`
- `MSymbol Minput_get_surrounding_text`
- `MSymbol Minput_delete_surrounding_text`

変数: 特別な入力イベント用定義済みシンボル.

`minput_filter()` の `KEY` 引数として用いられる定義済みシンボル。

- `MSymbol Minput_focus_out`
- `MSymbol Minput_focus_in`
- `MSymbol Minput_focus_move`

変数: 入力メソッド情報用定義済みシンボル.

- `MSymbol Minherited`
- `MSymbol Mcustomized`
- `MSymbol Mconfigured`

関数

- `MInputMethod * minput_open_im (MSymbol language, MSymbol name, void *arg)`
入力メソッドをオープンする.
- `void minput_close_im (MInputMethod *im)`
入力メソッドをクローズする.
- `MInputContext * minput_create_ic (MInputMethod *im, void *arg)`
入力コンテキストを生成する.
- `void minput_destroy_ic (MInputContext *ic)`
入力コンテキストを破壊する.
- `int minput_filter (MInputContext *ic, MSymbol key, void *arg)`
入力キーをフィルタする.
- `int minput_lookup (MInputContext *ic, MSymbol key, void *arg, MText *mt)`
入力コンテキスト中のテキストを探す.
- `void minput_set_spot (MInputContext *ic, int x, int y, int ascent, int descent, int fontsize, MText *mt, int pos)`
入力コンテキストのスポットを設定する.
- `void minput_toggle (MInputContext *ic)`
入力メソッドを切替える.
- `void minput_reset_ic (MInputContext *ic)`
入力コンテキストをリセットする.
- `MPlist * minput_get_title_icon (MSymbol language, MSymbol name)`
入力メソッドのタイトルとアイコン用ファイル名を得る.
- `MText * minput_get_description (MSymbol language, MSymbol name)`

入力メソッドの説明テキストを得る.

- `MPList * minput_get_command (MSymbol language, MSymbol name, MSymbol command)`
- `int minput_config_command (MSymbol language, MSymbol name, MSymbol command, MPList *keyseqlist)`
- `MPList * minput_get_variable (MSymbol language, MSymbol name, MSymbol variable)`
- `int minput_config_variable (MSymbol language, MSymbol name, MSymbol variable, MPList *value)`

入力メソッドの変数の値を設定する.

- `char * minput_config_file ()`
ユーザ毎のカスタマイズファイルの名前を得る.
- `int minput_save_config (void)`
設定をユーザ毎のカスタマイズファイルに保存する.
- `MPList * minput_list (MSymbol language)`

Obsolete な関数

- `MPList * minput_get_variables (MSymbol language, MSymbol name)`
- `int minput_set_variable (MSymbol language, MSymbol name, MSymbol variable, void *value)`
入力メソッド変数の初期値を設定する.
- `MPList * minput_get_commands (MSymbol language, MSymbol name)`
入力メソッドのコマンドに関する情報を得る.
- `int minput_assign_command_keys (MSymbol language, MSymbol name, MSymbol command, MPList *keyseq)`
入力メソッドコマンドにキーシーケンスを割り当てる.
- `MPList * minput_parse_lm_names (MText *mt)`
- `int minput_callback (MInputContext *ic, MSymbol command)`

2.15.1 詳解

入力メソッド用API.

入力メソッドは多様な文字を入力するためのオブジェクトである。入力メソッドはシンボル `LANGUAGE` と `NAME` の組によって識別され、この組合せによって入力メソッドドライバが決定する。入力メソッドドライバとは、ある入力メソッドを扱うための関数の集まりである。入力メソッドには内部メソッドと外部メソッドの二種類がある。

• 内部入力メソッド

内部入力メソッドとは `LANGUAGE` が `Mnil` 以外のものであり、その本体は `m17n` データベースに `<Minput_method, LANGUAGE, NAME>` というタグを付けて定義されている。この種の入力メソッドに対して、`m17n` ライブラリではCUI用とGUI用それぞれの入力メソッドドライバをあらかじめ定義している。これらのドライバは `m17n` ライブラリ自体の入力処理エンジンを利用する。`m17n` データベースには、特定の言語専用でない入力メソッドを定義することもでき、そのような入力メソッドの `LANGUAGE` は `Mt` である。

内部入力メソッドは、ユーザの入力イベントに対応したシンボルである入力キーを受け取る。`m17n` ライブラリは入力イベントがアプリケーションプログラムでどう表現されているかを知ることができないので、入力イベントから入力キーへの変換はアプリケーションプログラマの責任で行わなくてはならない。詳細については関数 `minput_event_to_key()` の説明を参照。

- 外部入力メソッド外部入力メソッドとは LANGUAGE が Mnil のものであり、その本体は外部のリソースとして定義される。(たとえば X Window System の XIM など。) この種の入力メソッドでは、シンボル NAME は `Minput.driver` をキーとするプロパティを持ち、その値は入力メソッドドライバへのポインタである。このことにより、適切なドライバを準備することによって、いかなる種類の入力メソッドも m17n ライブラリの枠組の中で扱う事ができる。

利便性の観点から、m17n X ライブラリは XIM の OverTheSpot の入力スタイルを実現する入力メソッドドライバを提供し、またシンボル Mxim の `Minput.driver` プロパティの値としてそのドライバへのポインタを保持している。詳細については m17n GUI API のドキュメントを参照のこと。

処理の流れ

入力メソッド処理の典型的な処理は以下のようになる。

- 入力メソッドのオープン
- その入力メソッドの入力コンテキストの生成
- 入力イベントのフィルタ
- 入力コンテキストでの生成テキストの検索

2.15.2 型定義詳解

2.15.2.1 MInputCallbackFunc

```
typedef void(* MInputCallbackFunc) (MInputContext *ic, MSymbol command)
```

入力メソッドコールバック関数の型宣言。

入力メソッドから呼ばれるコールバック関数の型である。ic は入力コンテキストへのポインタ、command は関数が呼ばれるコールバックの名前である。

2.15.3 列挙型詳解

2.15.3.1 MInputCandidatesChanged

```
enum MInputCandidatesChanged
```

入力メソッドの入力候補がどう変更されたかを示すビットマスク。

列挙値

MINPUT_CANDIDATES_LIST_CHANGED	
MINPUT_CANDIDATES_INDEX_CHANGED	
MINPUT_CANDIDATES_SHOW_CHANGED	
MINPUT_CANDIDATES_CHANGED_MAX	

2.15.4 関数詳解

2.15.4.1 minput_open_im()

```
MInputMethod* minput_open_im (
    MSymbol language,
    MSymbol name,
    void * arg )
```

入力メソッドをオープンする。

関数 `minput_open_im()` は言語 `language` と名前 `name` に合致する入力メソッドをオープンし、新たに割り当てられた入力メソッドオブジェクトへのポインタを返す。

この関数は、まず入力メソッド用のドライバを以下のようにして決定する。

`language` が `Mnil` でなければ、変数 `minput_driver` で指されているドライバを用いる。

`language` が `Mnil` であり、`name` が `Minput_driver` プロパティを持つ場合には、そのプロパティの値で指されている入力ドライバを用いて入力メソッドをオープンする。`name` にそのようなプロパティが無かった場合は `NULL` を返す。

次いで、ドライバのメンバ `MInputDriver::open_im()` が呼ばれる。

`arg` は構造体 `MInputMethod` のメンバ `arg` に設定され、ドライバから参照できる。

2.15.4.2 minput_close_im()

```
void minput_close_im (
    MInputMethod * im )
```

入力メソッドをクローズする。

関数 `minput_close_im()` は、入力メソッド `im` をクローズする。この入力メソッド `im` は `minput_open_im()` によって作られたものでなければならない。

2.15.4.3 minput_create_ic()

```
MInputContext* minput_create_ic (
    MInputMethod * im,
    void * arg )
```

入力コンテキストを生成する。

関数 `minput_create_ic()` は入力メソッド `im` に対応する入力コンテキストオブジェクトを生成し、`Minput_preedit_start`, `Minput_status_start`, `Minput_status_draw` に対応するコールバック関数をこの順に呼ぶ。

戻り値:

入力コンテキストが生成された場合、`minput_create_ic()` はその入力コンテキストへのポインタを返す。失敗した場合は `NULL` を返す。

2.15.4.4 minput_destroy_ic()

```
void minput_destroy_ic (
    MInputContext * ic )
```

入力コンテキストを破壊する。

関数 `minput_destroy_ic()` は、入力コンテキスト `ic` を破壊する。この入力コンテキストは `minput_create_ic()` によって作られたものでなければならない。この関数は `Minput_preedit_done`, `Minput_status_done`, `Minput_candidates_done` に対応するコールバック関数をこの順に呼ぶ。

2.15.4.5 minput_filter()

```
int minput_filter (
    MInputContext * ic,
    MSymbol key,
    void * arg )
```

入力キーをフィルタする。

関数 `minput_filter()` は入力キー `key` を入力コンテキスト `ic` に応じてフィルタし、`preedit` テキスト、ステータス、現時点での候補が変化した時点で、それぞれ `Minput_preedit_draw`, `Minput_status_draw`, `Minput_candidates_draw` に対応するコールバック関数を呼ぶ。

戻り値:

`key` がフィルタされれば、この関数は 1 を返す。この場合呼び出し側はこのキーを捨てるべきである。そうでなければ 0 を返し、呼び出し側は、たとえば同じキーで関数 `minput_lookup()` を呼ぶなどして、このキーを処理する。

2.15.4.6 minput_lookup()

```
int minput_lookup (
    MInputContext * ic,
    MSymbol key,
    void * arg,
    MText * mt )
```

入力コンテキスト中のテキストを探す。

関数 `minput_lookup()` は入力コンテキスト `ic` 中のテキストを探す。 `key` は関数 `minput_filter()` への直前の呼び出しに用いられたものと同じでなくてはならない。

テキストが入力メソッドによって生成されていれば、テキストは M-text `mt` に連結される。

この関数は、`::MInputDriver::lookup` を呼ぶ。

戻り値:

`key` が入力メソッドによって適切に処理できれば、この関数は 0 を返す。そうでなければ -1 を返す。この場合でも `mt` に何らかのテキストが生成されていることがある。

2.15.4.7 minput_set_spot()

```
void minput_set_spot (
    MInputContext * ic,
    int x,
    int y,
    int ascent,
    int descent,
    int fontsize,
    MText * mt,
    int pos )
```

入力コンテキストのスポットを設定する。

関数 `minput_set_spot()` は、入力コンテキスト `ic` のスポットを、座標 (x, y) の位置に、高さ `ascent`、`descent` で設定する。これらの値の意味は入力メソッドドライバに依存する。

たとえば CUI 環境で動作するドライバは `x` と `y` をそれぞれ列と行の番号として用い、`ascent` と `descent` を無視するかもしれない。またウィンドウシステム用のドライバは `x` と `y` をクライアントウィンドウの原点からのオフセットをピクセル単位で表したものと扱い、`ascent` と `descent` を (x, y) の列のアセントとディセントをピクセル単位で表したものと扱うかもしれない。

`fontsize` には `preedit` テキストのフォントサイズを 1/10 ポイント単位で指定する。

`mt` と `pos` はそのスポットの M-text と文字位置である。`mt` は NULL でもよく、その場合には入力メソッドはスポット周辺のテキストに関する情報を得ることができない。

2.15.4.8 minput_toggle()

```
void minput_toggle (
    MInputContext * ic )
```

入力メソッドを切替える。

関数 `minput_toggle()` は入力コンテキスト `ic` に対応付けられた入力メソッドをトグルする。

2.15.4.9 minput_reset_ic()

```
void minput_reset_ic (
    MInputContext * ic )
```

入力コンテキストをリセットする。

関数 `minput_reset_ic()` は `Minput_reset` に対応するコールバック関数を呼ぶことによって入力コンテキスト `ic` をリセットする。リセットとは、実際には入力メソッドを初期状態に移すことである。現在入力中のテキストはコミットされることなく削除されるので、アプリケーションプログラムは、必要ならば予め `minput_filter()` を引数 `key Mnil` で呼んで強制的にプリエディットテキストをコミットさせること。

2.15.4.10 minput_get_title_icon()

```
MPlist* minput_get_title_icon (
    MSymbol language,
    MSymbol name )
```

入力メソッドのタイトルとアイコン用ファイル名を得る.

関数 `minput_get_title_icon()` は、`language` と `name` で指定される 入力メソッドのタイトルと（あれば）アイコン用ファイルを含む `plist` を返す。

`plist` の第一要素は、`::Mtext` をキーに持ち、値は入力メソッドを識別する タイトルを表す `M-text` である。第二要素があれば、キーは `Mtext` であり、値は識別用アイコン画像ファイルの絶対パスを表す `M-text` である。

戻り値:

指定の入力メソッドが存在し、タイトルが定義されていれば `plist` を返す。そうでなければ `NULL` を返す。呼出側は 関数 `m17n_object_unref()` を用いて `plist` を解放しなくてはならない。

2.15.4.11 minput_get_description()

```
MText* minput_get_description (
    MSymbol language,
    MSymbol name )
```

入力メソッドの説明テキストを得る.

関数 `minput_get_description()` は、`language` と `name` によって指定された入力メソッドを説明する `M-text` を返す。

戻り値:

指定された入力メソッドが説明するテキストを持っていれば、`MText` へのポインタを返す。呼び出し側は、それを `m17n_object_unref()` を用いて解放しなくてはならない。入力メソッドに説明テキストが無ければ `NULL` を返す。

2.15.4.12 minput_get_command()

```
MPList* minput_get_command (
    MSymbol language,
    MSymbol name,
    MSymbol command )
```

@brief 入力メソッドのコマンドに関する情報を得る。

関数 `minput_get_command()` は、@b language と @b name で指定される入力メソッドのコマンド @b command に関する情報を返す。入力メソッドのコマンドとは、疑似キーイベントであり、1 つ以上の実際の入力キーシーケンスが割り当てられる。

コマンドには、グローバルとローカルの 2 種類がある。グローバルなコマンドはグローバルに定義され、ローカルなコマンドはその説明とキー割り当てを継承することができる。各入力メソッドはローカルなキー割当てを持つローカルなコマンドを定義する。また同名のグローバルなコマンドの定義を継承するローカルなコマンドを宣言することもできる。

@b language が #Mt で @b name が #Mnil の場合は、この関数はグローバルコマンドに関する情報を返す。そうでなければローカルコマンドに関するものを返す。

@b command が #Mnil の場合は、すべてのコマンドに関する情報を返す。

戻り値は以下の形式の @e well-formed plist (@ref m17nPlist) である。

```
((NAME DESCRIPTION STATUS [KEYSEQ ...]) ...)
```

NAME はコマンド名を示すシンボルである。

DESCRIPTION はコマンドを説明する M-text であるか、説明が無い場合には **Mnil** である。

STATUS はキー割り当てがどのように定められるかをあらわすシンボルであり、その値は **Mnil** (デフォルトの割り当て)、**Mcustomized** (ユーザ毎のカスタマイズファイルによってカスタマイズされた割り当て)、**Mconfigured** (`minput_config_command()` を呼ぶことによって設定される割り当て) のいずれかである。ローカルコマンドの場合には、**Minherited** (対応するグローバルコマンドからの継承による割り当て) でもよい。

KEYSEQ は 1 つ以上のシンボルからなる plist であり、各シンボルはコマンドに割り当てられているキーシーケンスを表す。KEYSEQ が無い場合は、そのコマンドは現状で使用不能である。(すなわちコマンドの動作を起動できるキーシーケンスが無い。)

command が **Mnil** でなければ、返される plist の最初の要素は、command に関する情報を含む。

戻り値:

求められた情報が見つければ、空でない plist へのポインタを返す。リストはライブラリが管理しているので、呼出側が変更したり解放したりすることはできない。

そうでなければ、すなわち指定の入力メソッドやコマンドが存在しなければ **NULL** を返す。

例：

```
MText *
get_im_command_description (MSymbol language, MSymbol name, MSymbol command)
{
    /* Return a description of the command COMMAND of the input method
       specified by LANGUAGE and NAME. */
    MPlist *cmd = minput_get_command (language, name, command);
    MPlist *plist;
    if (! cmd)
        return NULL;
    plist = mplist_value (cmd); /* (NAME DESCRIPTION STATUS KEY-SEQ ...) */
    plist = mplist_next (plist); /* (DESCRIPTION STATUS KEY-SEQ ...) */
    return (mplist_key (plist) == Mtext
        ? (MText *) mplist_value (plist)
        : NULL);
}
```

2.15.4.13 minput_config_command()

```
int minput_config_command (
    MSymbol language,
    MSymbol name,
    MSymbol command,
    MPlist * keyseqlist )
```

@brief 入力メソッドのコマンドのキーシーケンスを設定する。

関数 minput_config_command() はキーシーケンスのリスト @b keyseqlist を、@b language と @b name によって指定される入力メソッドのコマンド @b command に割り当てる。

@b keyseqlist が空リストでなければ、キーシーケンスのリストであり、各キーシーケンスはシンボルの plist である。

@b keyseqlist が空の plist ならば、そのコマンドの設定やカスタマイズはすべてキャンセルされ、デフォルトのキーシーケンスが有効になる。

@b keyseqlist が NULL であれば、そのコマンドの設定はキャンセルされ、元のキーシーケンス（ユーザ毎のカスタマイズファイルに保存されているもの、あるいはデフォルトのもの）が有効になる。

後のふたつの場合には、@b command は #Mnil をとることができ、指定の入力メソッドの全てのコマンド設定のキャンセルを意味する。

@b name が #Mnil ならば、この関数は個々の入力メソッドではなくグローバルなコマンドのキー割り当てを設定する。

これらの設定は、現行のセッション中で入力メソッドがオープン（または再オープン）された時点で有効になる。将来のセッション中でも有効にするためには、関数 minput_save_config() を用いてユーザ毎のカスタマイズファイルに保存しなくてはならない。

@par 戻り値:

この関数は、処理が成功すれば 0 を、失敗すれば -1 を返す。失敗とは以下の場合である。

```
<ul>
<li>@b keyseqlist が有効な形式でない。
<li>@b command が指定の入力メソッドで利用できない。
<li>@b language と @b name で指定される入力メソッドが存在しない。
</ul>
```

@par 参照:

minput_get_commands(), minput_save_config() .

例：

```
/* Add "C-x u" to the "start" command of Unicode input method. */
{
  MSymbol start_command = msymbol ("start");
  MSymbol unicode = msymbol ("unicode");
  Mplist *cmd, *plist, *key_seq_list, *key_seq;
  /* At first get the current key-sequence assignment. */
  cmd = minput_get_command (Mt, unicode, start_command);
  if (! cmd)
    {
      /* The input method does not have the command "start". Here
         should come some error handling code. */
    }
  /* Now CMD == ((start DESCRIPTION STATUS KEY-SEQUENCE ...) ...).
     Extract the part (KEY-SEQUENCE ...). */
  plist = mplist_next (mplist_next (mplist_next (mplist_value (cmd))));
  /* Copy it because we should not modify it directly. */
  key_seq_list = mplist_copy (plist);

  key_seq = mplist ();
  mplist_add (key_seq, MSymbol, msymbol ("C-x"));
  mplist_add (key_seq, MSymbol, msymbol ("u"));
  mplist_add (key_seq_list, Mplist, key_seq);
  ml7n_object_unref (key_seq);
  minput_config_command (Mt, unicode, start_command, key_seq_list);
  ml7n_object_unref (key_seq_list);
}
```

2.15.4.14 minput_get_variable()

```
Mplist* minput_get_variable (
  MSymbol language,
  MSymbol name,
  MSymbol variable )
```

@brief 入力メソッドの変数に関する情報を得る。

関数 `minput_get_variable()` は、@b language と @b name で指定される入力メソッドの変数 @b variable に関する情報を返す。入力メソッドの変数とは、入力メソッドの振舞を制御するものである。

変数には、グローバルとローカルの2種類がある。グローバルな変数はグローバルに定義され、ローカルな変数はその説明と値を継承することができる。各入力メソッドはローカルな値を持つローカルな変数を定義する。また同名のグローバルな変数の定義を継承するローカルな変数を宣言することもできる。

@b language が #Mt で @b name が #Mnil の場合は、この関数はグローバル変数に関する情報を返す。そうでなければローカル変数に関するものを返す。

@b variable が #Mnil の場合は、すべてのコマンドに関する情報を返す。

戻り値は以下の形式の @e well-formed plist (@ref ml7nPlist) である。

```
((NAME DESCRIPTION STATUS VALUE [VALID-VALUE ...]) ...)
```

@c NAME は変数の名前を示すシンボルである。

@c DESCRIPTION は変数を説明する M-text であるか、説明が無い場合には #Mnil である。

@c STATUS は値がどのように定められるかをあらわすシンボルであり、@c STATUS の値は #Mnil (デフォルトの値), @b Mcustomized (ユーザ毎のカスタマイズファイルによってカスタマイズされた値), @b Mconfigured

(`minput_config_variable()` を呼ぶことによって設定される値) のいずれかである。ローカル変数の場合には、`@b Minherited` (対応するグローバル変数から継承した値) でもよい。

`@c VALUE` は変数の初期値である。この要素のキーが `#Mt` であれば初期値を持たない。そうでなければ、キーは `#Minteger`, `#Msymbol`, `#Mtext` のいずれかであり、値はそれぞれ対応する型のものである。

`@c VALID-VALUE` はもしあれば、変数の取り得る値を指定する。これは `@c VALUE` と同じ型 (すなわち同じキーを持つ) であるが、例外として `@c VALUE` が `integer` の場合は `@c VALID-VALUE` は可能な値の範囲を示す二つの整数からなる `plist` となることができる。

`@c VALID-VALUE` がなければ、変数は `@c VALUE` と同じ型である限りいかなる値もとることができる。

`@b variable` が `#Mnil` でなければ、返される `plist` の最初の要素は `@b variable` に関する情報を含む。

`@par` 戻り値:

求められた情報が見つければ、空でない `plist` へのポインタを返す。リストはライブラリが管理しているので、呼出側が変更したり解放したりすることはできない。

そうでなければ、すなわち指定の入力メソッドや変数が存在しなければ `@c NULL` を返す。

2.15.4.15 minput_config_variable()

```
int minput_config_variable (
    MSymbol language,
    MSymbol name,
    MSymbol variable,
    MPlist * value )
```

入力メソッドの変数の値を設定する。

関数 `minput_config_variable()` は値 `value` を、`language` と `name` によって指定される入力メソッドの変数 `variable` に割り当てる。

`value` が空リストでなければ、1 要素の `plist` であり、そのキーは `Minteger`, `Msymbol`, `Mtext` のいずれか、値は対応する型のものである。この値が変数 `variable` に割り当てられる。

`value` が空リストであれば、変数の設定とカスタマイズがキャンセルされ、デフォルト値が変数 `variable` に割り当てられる。

`value` が `NULL` であれば、変数の設定はキャンセルされ、元の値 (ユーザ毎のカスタマイズファイル中の値、またはデフォルトの値) が割り当てられる。

後のふたつの場合には、`variable` は `Mnil` をとることができ、指定された入力メソッドの全ての変数設定のキャンセルを意味する。

`name` が `Mnil` ならば、この関数は個々の入力メソッドではなくグローバルな変数の値を設定する。

これらの設定は、現行のセッション中で入力メソッドがオープン (または再オープン) された時点で有効になる。将来のセッション中でも有効にするためには、関数 `minput_save_config()` を用いてユーザ毎のカスタマイズファイルに保存しなくてはならない。

戻り値:

この関数は、処理が成功すれば 0 を、失敗すれば -1 を返す。失敗とは以下の場合である。

- `value` が有効な形式でない。型が定義に合わない、または値が範囲外である。
- `variable` が指定の入力メソッドで利用できない。
- `language` と `name` で指定される入力メソッドが存在しない。

参照:

[minput_get_commands\(\)](#), [minput_save_config\(\)](#).

2.15.4.16 minput_config_file()

```
char* minput_config_file (  
    void )
```

ユーザ毎のカスタマイズファイルの名前を得る。

関数 [minput_config_file\(\)](#) は、関数 [minput_save_config\(\)](#) が設定を保存するユーザ毎のカスタマイズファイルへの絶対パス名を返す。通常は、ユーザのホームディレクトリの下のディレクトリ ".ml7n.d" にある "config.mic" となる。返された名前のファイルが存在するか、読み書きできるかは保証されない。関数 [minput_save_config\(\)](#) が失敗して -1 を返した場合には、アプリケーションプログラムはファイルの存在を確認し、（できれば）書き込み可能にし再度 [minput_save_config\(\)](#) を試すことができる。

戻り値:

この関数は文字列を返す。文字列はライブラリが管理しているので、呼出側が修正したり解放したりすることはできない。

参照:

[minput_save_config\(\)](#)

2.15.4.17 minput_save_config()

```
int minput_save_config (
    void )
```

設定をユーザ毎のカスタマイズファイルに保存する。

関数 `minput_save_config()` は現行のセッションでこれまでに行った設定をユーザ毎のカスタマイズファイルに保存する。

戻り値:

成功すれば 1 を返す。ユーザ毎のカスタマイズファイルがロックされていれば 0 を返す。この場合、呼出側はしばらく待って再試行できる。設定ファイルが書き込み不可の場合、-1 を返す。この場合、`minput_config_file ()` を呼んでファイル名をチェックし、できれば書き込み可能にし、再試行できる。

参照:

[minput_config_file\(\)](#)

2.15.4.18 minput_list()

```
MPlist* minput_list (
    MSymbol language )
```

例:

```
#include <stdio.h>
#include <string.h>
#include <m17n.h>
int
main (int argc, char **argv)
{
    MPlist *imlist, *pl;
    M17N_INIT();
    imlist = minput_list ((argc > 1) ? msymbol (argv[1]) : Mnil);
    for (pl = imlist; mplist_key (pl) != Mnil; pl = mplist_next (pl))
    {
        MPlist *p = mplist_value (pl);
        MSymbol lang, name, sane;
        lang = mplist_value (p);
        p = mplist_next (p);
        name = mplist_value (p);
        p = mplist_next (p);
        sane = mplist_value (p);
        printf ("%s %s %s\n", msymbol_name (lang), msymbol_name (name),
            sane == Mt ? "ok" : "no");
    }
    m17n_object_unref (imlist);
    M17N_FINI();
    exit (0);
}
```

2.15.4.19 minput_get_variables()

```
MPlist* minput_get_variables (
    MSymbol language,
    MSymbol name )
```

@brief 入力メソッドの変数リストを得る。

関数 minput_get_variables() は、@b language と @b name によって指定された入力メソッドの振る舞いを制御する変数のプロパティリスト (#MPlist) を返す。このリストは @e well-formed であり (@ref m17nPlist) 以下の形式である。

```
(VARNAME (DOC-MTEXT DEFAULT-VALUE [ VALUE ... ] )
 VARNAME (DOC-MTEXT DEFAULT-VALUE [ VALUE ... ] )
 ...)
```

@c VARNAME は変数の名前を示すシンボルである。

@c DOC-MTEXT は変数を説明する M-text である。

@c DEFAULT-VALUE は変数のデフォルト値であり、シンボル、整数もしくは M-text である。

@c VALUE は、もし指定されていれば変数の取り得る値を示す。もし @c DEFAULT-VALUE が整数なら、@c VALUE は (@c FROM @c TO) という形のリストでも良い。この場合 @c FROM と @c TO は可能な値の範囲を示す。

例として、ある入力メソッドが次のような変数を持つ場合を考えよう。

```
@li name:intvar, 説明:"value is an integer",
  初期値:0, 値の範囲:0..3,10,20

@li name:symvar, 説明:"value is a symbol",
  初期値:nil, 値の範囲:a, b, c, nil

@li name:txtvar, 説明:"value is an M-text",
  初期値:empty text, 値の範囲なし (どんな M-text でも可)
```

この場合、返されるリストは以下のようになる。

```
(intvar ("value is an integer" 0 (0 3) 10 20)
 symvar ("value is a symbol" nil a b c nil)
 txtvar ("value is an M-text" ""))
```

@par 戻り値:

入力メソッドが何らかの変数を使用していれば #MPlist へのポインタを返す。返されるプロパティリストはライブラリによって管理されており、呼び出し側で変更したり解放したりしてはならない。
入力メソッドが変数を一切使用してなければ、@c NULL を返す。

2.15.4.20 minput_set_variable()

```
int minput_set_variable (
    MSymbol language,
    MSymbol name,
    MSymbol variable,
    void * value )
```

入力メソッド変数の初期値を設定する。

関数 `minput_set_variable()` は、`language` と `name` によって指定された入力メソッドの入力メソッド変数 `variable` の初期値を、`value` に設定する。

デフォルトの初期値は 0 である。

この設定は、新しくオープンされた入力メソッドから有効となる。

戻り値:

処理が成功すれば 0 を返す。そうでなければ -1 を返し、`merror_code` を `MERROR_IM` に設定する。

2.15.4.21 minput_get_commands()

```
MPlist* minput_get_commands (
    MSymbol language,
    MSymbol name )
```

入力メソッドのコマンドに関する情報を得る。

関数 `minput_get_commands()` は、`language` と `name` によって指定された入力メソッドの入力メソッドコマンドに関する情報を返す。入力メソッドコマンドとは、疑似キーイベントであり、それぞれに 1 つ以上の実際の入力キーシーケンスが割り当てられているものを指す。

コマンドにはグローバルとローカルの 2 種類がある。グローバルコマンドは複数の入力メソッドにおいて、同じ目的で、グローバルなキー割り当てで用いられる。ローカルコマンドは特定の入力メソッドでのみ、ローカルなキー割り当てで使用する。

個々の入力メソッドはグローバルコマンドのキー割り当てを変更することもできる。グローバルコマンド用のグローバルキー割り当ては、使用する入力メソッドにおいてそのコマンド用のローカルなキー割り当てが存在しない場合にのみ有効である。

`name` が `Mnil` であれば、グローバルコマンドに関する情報を返す。この場合、`language` は無視される。

`name` が `Mnil` でなければ、`language` と `name` によって指定される入力メソッドに付けるローカルなキー割り当てを持つコマンドに関する情報を返す。

戻り値:

入力メソッドコマンドが見つからなければ、この関数は `NULL` を返す。

そうでなければプロパティリストへのポインタを返す。リストの各要素のキーは個々のコマンドを示すシンボルであり、値は下記の `COMMAND-INFO` の形式のプロパティリストである。

`COMMAND-INFO` の第一要素のキーは `Mtext` または `MSymbol` である。キーが `Mtext` なら、値はそのコマンドを説明する M-text である。キーが `MSymbol` なら値は `Mnil` であり、このコマンドは説明テキストを持たないことになる。

それ以外の要素が無ければ、このコマンドに対してキーシーケンスが割り当てられていないことを意味する。そうでなければ、残りの各要素はキーとして `::Mplist` を、値としてプロパティリストを持つ。このプロパティリストのキーは `MSymbol` であり、値は現在そのコマンドに割り当てられている入力キーを表すシンボルである。

返されるプロパティリストはライブラリによって管理されており、呼び出し側で変更したり解放したりしてはならない。

2.15.4.22 minput_assign_command_keys()

```
int minput_assign_command_keys (
    MSymbol language,
    MSymbol name,
    MSymbol command,
    MPlist * keyseq )
```

入力メソッドコマンドにキーシーケンスを割り当てる。

関数 `minput_assign_command_keys()` は、`language` と `name` によって 指定された入力メソッド用の入力メソッドコマンド `command` に対して、入力キーシーケンス `keyseq` を割り当てる。 `name` が `Mnil` ならば、`language` に関係なく、入力キーシーケンスはグローバルに割り当てられる。そうでなければ、割り当てはローカルである。

`keyseq` の各要素はキーとして `msymbol` を、値として入力キーを表すシンボルを持たなくてはならない。

`keyseq` は `NULL` でもよい。この場合、グローバルもしくはローカルなすべての割り当てが消去される。

この割り当ては、割り当て以降新しくオープンされた入力メソッドから有効になる。

戻り値:

処理が成功すれば 0 を返す。そうでなければ -1 を返し、 `merror_code` を `MERROR_IM` に設定する。

2.15.4.23 minput_parse_im_names()

```
MPlist* minput_parse_im_names (
    MText * mt )
```

2.15.4.24 minput_callback()

```
int minput_callback (
    MInputContext * ic,
    MSymbol command )
```

2.15.5 変数詳解

2.15.5.1 Minput_method

`MSymbol` `Minput_method`

"input-method" を名前として持つシンボル。

2.15.5.2 Minput_preedit_start

`MSymbol` `Minput_preedit_start`

2.15.5.3 Minput_preedit_done

`MSymbol` `Minput_preedit_done`

2.15.5.4 Minput_preedit_draw

`MSymbol` `Minput_preedit_draw`

2.15.5.5 Minput_status_start

`MSymbol` `Minput_status_start`

2.15.5.6 Minput_status_done

`MSymbol` `Minput_status_done`

2.15.5.7 Minput_status_draw

`MSymbol` `Minput_status_draw`

2.15.5.8 Minput_candidates_start

`MSymbol` `Minput_candidates_start`

2.15.5.9 Minput_candidates_done

`MSymbol` `Minput_candidates_done`

2.15.5.10 Minput_candidates_draw

`MSymbol` `Minput_candidates_draw`

2.15.5.11 Minput_set_spot

`MSymbol` `Minput_set_spot`

2.15.5.12 Minput_toggle

`MSymbol` `Minput_toggle`

2.15.5.13 Minput_reset

`MSymbol` `Minput_reset`

2.15.5.14 Minput_get_surrounding_text

`MSymbol` `Minput_get_surrounding_text`

2.15.5.15 Minput_delete_surrounding_text

`MSymbol` `Minput_delete_surrounding_text`

2.15.5.16 Minput_focus_out

`MSymbol` Minput_focus_out

2.15.5.17 Minput_focus_in

`MSymbol` Minput_focus_in

2.15.5.18 Minput_focus_move

`MSymbol` Minput_focus_move

2.15.5.19 Minherited

`MSymbol` Minherited

入力メソッドのコマンドや変数の状態を表し、`minput_get_command()` と `minput_get_variable()` の戻り値として用いられる定義済みシンボル。

2.15.5.20 Mcustomized

`MSymbol` Mcustomized

2.15.5.21 Mconfigured

`MSymbol` Mconfigured

2.15.5.22 minput_default_driver

`MInputDriver minput_default_driver`

内部入力メソッド用デフォルトドライバ.

変数 `minput_default_driver` は内部入力メソッド用のデフォルトのドライバを表す。

メンバ `MInputDriver::open_jm()` は `m17n` データベース中からタグ `< Minput_method, language, name>` に合致する入力メソッドを探し、それをロードする。

メンバ `MInputDriver::callback_jlist()` は `NULL` であり、したがって、プログラマ側で責任を持って適切なコールバック関数の `plist` に設定しなくてはならない。さもないと、`preedit` テキストなどのフィードバック情報がユーザに表示されない。

マクロ `M17N_INIT()` は変数 `minput_driver` をこのドライバへのポインタに設定し、全ての内部入力メソッドがこのドライバを使うようにする。

したがって、`minput_driver` がデフォルト値のままであれば、`minput_` で始まる関数のドライバに依存する引数 `arg` はすべて無視される。

2.15.5.23 minput_driver

`MInputDriver* minput_driver`

内部入力メソッド用ドライバ.

変数 `minput_driver` は内部入力メソッドによって使用されている入力メソッドドライバへのポインタである。マクロ `M17N_INIT()` はこのポインタを `::minput_default_driver` (`<m17n.h>` が `include` されている時) に初期化する。

2.15.5.24 Minput_driver

`MSymbol Minput_driver`

The variable `Minput_driver` is a symbol for a foreign input method. See [foreign input method](#) for the detail.

2.16 FLT API

`libm17n-flt.so` が提供する API

データ構造

- struct [MFLTGlyph](#)
グリフに関する情報の型.
- struct [MFLTGlyphAdjustment](#)
グリフ位置調整情報のための型.
- struct [MFLTGlyphString](#)
グリフ列の情報のための型.
- struct [MFLTOfSpec](#)
GSUB および GPOS OpenType テーブルの仕様のための型.
- struct [MFLTFont](#)
FLT ドライバが使うフォントの型.

型定義

- typedef struct _MFLT [MFLT](#)
FLT (Font Layout Table) の型.

関数

- [MFLT * mflt_get](#) ([MSymbol](#) name)
指定された名前を持つ FLT オブジェクトを返す.
- [MFLT * mflt_find](#) (int c, [MFLTFont](#) *font)
指定された文字とフォントに合った FLT を探す.
- const char * [mflt_name](#) ([MFLT](#) *flt)
FLT の名前を返す.
- [MCharTable * mflt_coverage](#) ([MFLT](#) *flt)
FLT の範囲を返す.
- int [mflt_run](#) ([MFLTGlyphString](#) *gstring, int from, int to, [MFLTFont](#) *font, [MFLT](#) *flt)
FLT を使って文字をレイアウトする.
- [MFLT * mdebug_dump_flt](#) ([MFLT](#) *flt, int indent)
- void [mflt_dump_gstring](#) ([MFLTGlyphString](#) *gstring)

変数

- int [mflt_enable_new_feature](#)
- int(* [mflt_iterate_otf_feature](#))(struct _MFLTFont *font, [MFLTOfSpec](#) *spec, int from, int to, unsigned char *table)
- [MSymbol](#)(* [mflt_font_id](#))(struct _MFLTFont *font)
- int(* [mflt_try_otf](#))(struct _MFLTFont *font, [MFLTOfSpec](#) *spec, [MFLTGlyphString](#) *gstring, int from, int to)

2.16.1 詳解

libm17n-flt.so が提供する API

ウィンドウシステムのための FLT サポート.

このセクションでは、FLT (Font Layout Table) を用いた文字レイアウト機能に関する m17n FLT API を定義する。FLT の形式は [フォントレイアウトテーブル](#) に記述されている。

2.16.2 型定義詳解

2.16.2.1 MFLT

```
typedef struct _MFLT MFLT
```

FLT (Font Layout Table) の型.

型 **MFLT** は FLT オブジェクトのための型である。この内部構造は、アプリケーションプログラムからは隠蔽されている。

2.16.3 関数詳解

2.16.3.1 mflt_get()

```
MFLT * mflt_get (
    MSymbol name )
```

指定された名前を持つ FLT オブジェクトを返す。

関数 **mflt_get()** は、**name** という名前を持つ FLT オブジェクトを返す。

戻り値:

もし成功すれば、**mflt_get()** は見つかった FLT オブジェクトへのポインタを返す。失敗した場合は **NULL** を返す。

2.16.3.2 mflt_find()

```
MFLT * mflt_find (
    int c,
    MFLTFont * font )
```

指定された文字とフォントに合った FLT を探す。

関数 **mflt_find()** は、文字 **c** をフォント **font** でレイアウトするために最も適切な FLT を返す。

戻り値:

もし成功すれば、**mflt_find()** は見つかった FLT オブジェクトへのポインタを返す。失敗した場合は **NULL** を返す。

2.16.3.3 mflt_name()

```
const char * mflt_name (
    MFLT * flt )
```

FLT の名前を返す。

関数 `mflt_name()` は flt の名前を返す。

2.16.3.4 mflt_coverage()

```
MCharTable * mflt_coverage (
    MFLT * flt )
```

FLT の範囲を返す。

関数 `mflt_coverage()` は、flt がサポートする文字に対して 0 でない値を含む文字テーブルを返す。

2.16.3.5 mflt_run()

```
int mflt_run (
    MFLTGlyphString * gstring,
    int from,
    int to,
    MFLTFont * font,
    MFLT * flt )
```

FLT を使って文字をレイアウトする。

関数 `mflt_run()` は、gstring 中の from から to 直前までの文字を font を用いてレイアウトする。もし flt がゼロでなければ、その値をすべての文字に対して用いる。そうでなければ適切な FLT を自動的に選択する。

戻り値

>=0	実行成功を示す。返される値は、gstring->glyphs 中で以前 to によって示されていたグリフへのインデクスである。
-2	結果を格納するには gstring->glyphs が短すぎることを示す。呼び出し側は、より長い gstring->glyphs を用いて再度この関数を呼ぶことができる。
-1	その他のエラーが起きたことを示す。

2.16.3.6 mdebug_dump_flt()

```
MFLT* mdebug_dump_flt (
    MFLT * flt,
    int indent )
```

2.16.3.7 mflt_dump_gstring()

```
void mflt_dump_gstring (
    MFLTGlyphString * gstring )
```

2.16.4 変数詳解

2.16.4.1 mflt_enable_new_feature

```
int mflt_enable_new_feature
```

2.16.4.2 mflt_iterate_otf_feature

```
int(* mflt_iterate_otf_feature) (struct _MFLTFont *font, MFLTOfSpec *spec, int from, int to,
unsigned char *table) (
    struct _MFLTFont * font,
    MFLTOfSpec * spec,
    int from,
    int to,
    unsigned char * table )
```

2.16.4.3 mflt_font_id

```
MSymbol(* mflt_font_id) (struct _MFLTFont *font) (
    struct _MFLTFont * font )
```

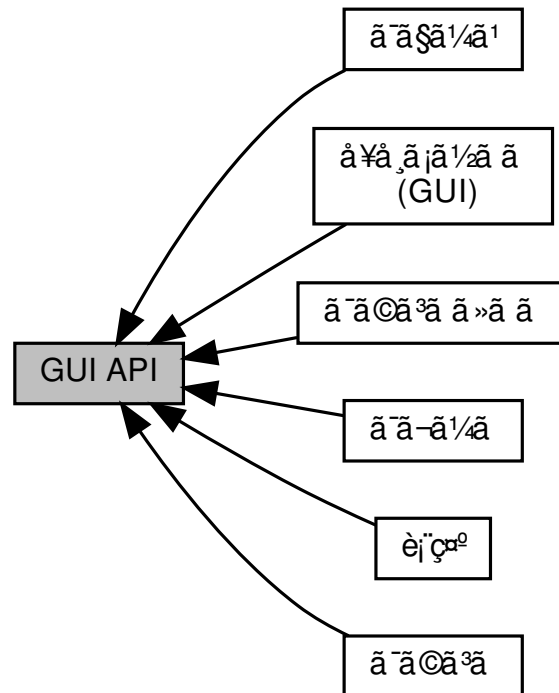
2.16.4.4 mflt_try_otf

```
int(* mflt_try_otf) (struct _MFLTFont *font, MFLTOfSpec *spec, MFLTGlyphString *gstring, int
from, int to) (
    struct _MFLTFont * font,
    MFLTOfSpec * spec,
    MFLTGlyphString * gstring,
    int from,
    int to )
```

2.17 GUI API

libm17n-gui.so が提供する API

GUI API 連携図



モジュール

- フレーム
フレームとはグラフィックデバイスに対応するオブジェクトである。
- フォント
- フォントセット
フォントセットは文字からフォントへの対応付けを行うオブジェクトである。
- フェース
フェースとは、M-text の見栄えを制御するオブジェクトである。
- 表示
M-text をウィンドウに描画する。
- 入力メソッド (GUI)
ウィンドウシステム上の入力メソッドのサポート。

2.17.1 詳解

libm17n-gui.so が提供する API

ウィンドウシステム上の GUI サポート。

このセクションはウィンドウシステムのもとでの M-text の表示と入力にかかわる m17n GUI API を定義する。

ここでのすべての定義はウィンドウシステムとは独立である。しかし、実際のライブラリファイルは個別のウィンドウシステムに依存する場合がある。たとえばライブラリファイル m17n-X.so は、m17n GUI API の X ウィンドウ用の実装例である。

現実には、GUI API は主にツールキットライブラリ向けであるか、または XOM を実装するために用いられており、アプリケーションプログラムからの直接の利用を念頭においたものではない。

2.18 フレーム

フレームとはグラフィックデバイスに対応するオブジェクトである。

フレーム 連携図



関数

- `MFrame * mframe (MPlist *plist)`
新しいフレームを作る。
- `void * mframe_get_prop (MFrame *frame, MSymbol key)`

変数

- `MFrame * mframe_default`
デフォルトのフレーム。

変数： フレームパラメータ用キー

フレームを生成する際のパラメータに用いるシンボル。詳しくは関数 `mframe()` の説明参照。

Mdevice、Mdisplay、Mscreen、Mdrawable、Mdepth、Mcolormap はフレームプロパティのキーでもある。

- `MSymbol Mdevice`
- `MSymbol Mdisplay`
- `MSymbol Mscreen`
- `MSymbol Mdrawable`
- `MSymbol Mdepth`
- `MSymbol Mcolormap`
- `MSymbol Mwidget`
- `MSymbol Mgd`

変数： フレームプロパティのキー

関数 `mframe.get_prop()` の引数に用いられるシンボル。

- `MSymbol Mfont`
- `MSymbol Mfont.width`
- `MSymbol Mfont.ascent`
- `MSymbol Mfont.descent`

2.18.1 詳解

フレームとはグラフィックデバイスに対応するオブジェクトである。

フレームとは `MFrame` 型のオブジェクトであり、個々の表示／入力デバイスの情報を格納するために用いられる。ほとんどすべての `m17n GUI` 関数は、引数としてフレームへのポインタを要求する。

2.18.2 関数詳解

2.18.2.1 `mframe()`

```
MFrame* mframe (
    MPlist * plist )
```

新しいフレームを作る。

関数 `mframe()` は `plist` 中のパラメータを持つ新しいフレームを作る。 `plist` は `NULL` でも良い。

`plist` に現われるキーのうちどれが認識されるかはウィンドウシステムに依存する。

以下のキーは常に認識される。

- `Mdevice`. 値は `Mx`, `Mgd`, `Mnil` のいずれかでなくてはならない。
値が `Mx` ならば、新しいフレームは X ウィンドウシステム用である。このフレームと共に指定された引数 `MDrawWindow` は、`Window` 型でなくてはならない。フレームは読み書きともに可能であり、すべての `GUI` 関数が使用できる。
値が `Mgd` ならば、新しいフレームは `GD` ライブラリのイメージオブジェクト用である。このフレームと共に指定された引数 `MDrawWindow` は、`gdImagePtr` 型でなくてはならない。フレームは書き出し専用であり、`minput_` で始まる名前の関数は使用できない。
値が `Mnil` ならば、新しいフレームは、`null` デバイス用である。このフレームは読み書きできないので、引数 `MDrawWindow` を必要とする `mdraw_` で始まる名前の関数や、`minput_` で始まる名前の関数は使用できない。
- `Mface`. 値は `MFace` へのポインタでなくてはならない。
この値はフレームのデフォルトのフェースとして用いられる。

これらのキーに加え、Mdevice のキーが **Mx** である場合に限り以下のキーも認識される。以下のキーはルートウィンドウと、フレームで利用できる **drawable** の深さを指定する。

- **Mdrawable**. 値は **Drawable** 型でなくてはならない。
キー **Mdisplay** を持つパラメータも指定されている必要がある。生成されたフレームは、指定されたディスプレイ上の指定された **drawable** と同じルートウィンドウと深さを持つ **drawable** に用いられる。
このパラメータがある場合には、**Mscreen** をキーとするパラメータは無視される。
- **Mwidget**. 値は **Widget** 型でなくてはならない。
生成されたフレームは、指定したウィジェットと同じルートウィンドウと深さを持つ **drawable** に用いられる。
キー **Mface** を持つパラメータがなければ、デフォルトのフェースはこのウィジェットのリソースから作られる。
このパラメータがある場合には、**Mdisplay**, **Mscreen**, **Mdrawable**, **Mdepth** をキーとするパラメータは無視される。
- **Mdepth**. 値は **unsigned** 型でなくてはならない。
生成されたフレームは、指定した深さの **drawable** に用いられる。
- **Mscreen**. 値は (**Screen** *) 型でなくてはならない。
生成したフレームは、指定したスクリーンと同じルートウィンドウを持ち、スクリーンのデフォルトの深さと同じ深さを持つ **drawable** に用いられる。
このパラメータがある場合には、**Mdisplay** をキーとするパラメータは無視される。
- **Mdisplay**. 値は (**Display** *) 型でなくてはならない。
生成されたフレームは、指定したディスプレイのデフォルトスクリーンと同じルートウィンドウと同じ深さを持つ **drawables** に用いられる。
- **Mcolormap**. 値は (**Colormap**) 型でなくてはならない。
生成されたフレームは、指定したカラーマップを使用する。
- **Mfont**. 値は、**::Mx**, **Mfreetype**, **Mxft** のいずれか。
生成されたフレームは指定したフォントバックエンドを使用する。値が **Mx** であれば **X** のコアフォント、**::Mfreetype** であれば **FreeType** でサポートされているローカルフォント、**::Mxft** であれば **Xft** ライブラリ経由で用いるローカルフォントを使用する。複数のフォントバックエンドを使用したい場合には、このパラメータを複数回、異なる値で指定することができる。指定したバックエンドがサポートされていないデバイスでは、このパラメータは無視される。
このパラメータが無い場合には、デバイスでサポートされているすべてのフォントバックエンドを利用する。

戻り値:

成功すれば **mframe()** は新しいフレームへのポインタを返す。そうでなければ **NULL** を返す。

2.18.2.2 mframe_get_prop()

```
void* mframe_get_prop (
    MFrame * frame,
    MSymbol key )
```

@brief フレームのプロパティの値を返す.

関数 `mframe_get_prop()` はフレーム `@b frame` のキー `@b key` を持つプロパティの値を返す。有効なキーとその値は以下の通り。

キー	値の型	値の意味
---	-----	-----
Mface	MFace *	デフォルトのフェース
Mfont	MFont *	デフォルトのフォント
Mfont_width	int	デフォルトのフォントの幅
Mfont_ascent	int	デフォルトのフォントの ascent
Mfont_descent	int	デフォルトのフォントの descent

m17n-X ライブラリでは、以下のキーも使用できる。

キー	値の型	値の意味
---	-----	-----
Mdisplay	Display *	フレームと関連付けられたディスプレイ
Mscreen	int	フレームと関連付けられたスクリーンのスクリーンナンバ
Mcolormap	Colormap	フレームのカラーマップ
Mdepth	unsigned	フレームの深さ

2.18.3 変数詳解

2.18.3.1 Mdevice

`MSymbol` Mdevice

2.18.3.2 Mdisplay

`MSymbol` Mdisplay

2.18.3.3 Mscreen

[MSymbol](#) Mscreen

2.18.3.4 Mdrawable

[MSymbol](#) Mdrawable

2.18.3.5 Mdepth

[MSymbol](#) Mdepth

2.18.3.6 Mcolormap

[MSymbol](#) Mcolormap

2.18.3.7 Mwidget

[MSymbol](#) Mwidget

2.18.3.8 Mgd

[MSymbol](#) Mgd

2.18.3.9 Mfont

[MSymbol](#) Mfont

2.18.3.10 Mfont_width

`MSymbol` Mfont_width

2.18.3.11 Mfont_ascent

`MSymbol` Mfont_ascent

2.18.3.12 Mfont_descent

`MSymbol` Mfont_descent

2.18.3.13 mframe_default

`MFrame*` mframe_default

デフォルトのフレーム.

外部変数 `mframe_default` は、デフォルトのフレームへのポインタを持つ。デフォルトのフレームは、最初に `mframe()` が呼び出されたときに作られる。

2.19 フォント

フォント 連携図



関数

- `MFont * mfont ()`
新しいフォントを作る.
- `MFont * mfont_parse_name (const char *name, MSymbol format)`
フォント名からフォントを作る.
- `char * mfont_unparse_name (MFont *font, MSymbol format)`
フォントからフォント名を作る.
- `MFont * mfont_copy (MFont *font)`
フォントのコピーを作る.
- `void * mfont_get_prop (MFont *font, MSymbol key)`
フォントのプロパティの値を得る.
- `int mfont_put_prop (MFont *font, MSymbol key, void *val)`
フォントのプロパティに値を設定する.
- `MSymbol * mfont_selection_priority ()`
フォント選択の優先度を返す.
- `int mfont_set_selection_priority (MSymbol *keys)`
フォント選択優先度を設定する.
- `MFont * mfont_find (MFrame *frame, MFont *spec, int *score, int max_size)`
フォントを探す.
- `int mfont_set_encoding (MFont *font, MSymbol encoding_name, MSymbol repertory_name)`
フォントのエンコーディングを設定する.
- `char * mfont_name (MFont *font)`
フォント名からフォントを作る.
- `MFont * mfont_from_name (const char *name)`
フォントからフォント名を作る.
- `int mfont_resize_ratio (MFont *font)`
フォントのリサイズ情報を得る
- `MPlist * mfont_list (MFrame *frame, MFont *font, MSymbol language, int maxnum)`
フォントのリストを得る
- `MPlist * mfont_list_family_names (MFrame *frame)`
- `int mfont_check (MFrame *frame, MFontset *fontset, MSymbol script, MSymbol language, MFont *font)`
- `int mfont_match_p (MFont *font, MFont *spec)`
- `MFont * mfont_open (MFrame *frame, MFont *font)`
- `MFont * mfont_encapsulate (MFrame *frame, MSymbol data_type, void *data)`
- `int mfont_close (MFont *font)`

変数

- `MPlist * mfont_freetype_path`
フォントファイルとフォントファイルを含むディレクトリのリスト.

変数: フォントプロパティを指定する定義済みシンボル

- [MSymbol Mfoundry](#)
開発元を指定するフォントプロパティのキー.
- [MSymbol Mfamily](#)
ファミリを指定するフォントプロパティのキー.
- [MSymbol Mweight](#)
太さを指定するフォントプロパティのキー.
- [MSymbol Mstyle](#)
スタイルを指定するフォントプロパティのキー.
- [MSymbol Mstretch](#)
幅を指定するフォントプロパティのキー.
- [MSymbol Madstyle](#)
adstyle を指定するフォントプロパティのキー.
- [MSymbol Mspacing](#)
spacing を指定するフォントプロパティのキー.
- [MSymbol Mregistry](#)
レジストリを指定するフォントプロパティのキー.
- [MSymbol Msize](#)
サイズを指定するフォントプロパティのキー.
- [MSymbol Mottf](#)
- [MSymbol Mfontfile](#)
フォントファイルを指定するフォントプロパティのキー.
- [MSymbol Mresolution](#)
解像度を指定するフォントプロパティのキー.
- [MSymbol Mmax_advance](#)
- [MSymbol Mfontconfig](#)
"fontconfig" という名前を持つシンボル.
- [MSymbol Mx](#)
"x" という名前を持つシンボル.
- [MSymbol Mfreetype](#)
"freetype" という名前を持つシンボル.
- [MSymbol Mxft](#)
"xft" という名前を持つシンボル.

2.19.1 詳解

`@addtogroup m17nFont`
`@brief` フォントオブジェクト.

`m17n GUI API` はフォントを `@c MFont` 型のオブジェクトとして表現する。フォントは `@e` フォントプロパティを持つことができる。他のタイプのプロパティ同様、フォントプロパティはキーと値からなり、キーは以下のシンボルのいずれかである。

`@c Mfoundry`, `@c Mfamily`, `@c Mweight`, `@c Mstyle`, `@c Mstretch`,
`@c Madstyle`, `@c Mregistry`, `@c Msize`, `@c Mresolution`, `@c Mspacing`

フォントプロパティのキーが `@c Msize` あるいは `@c Mresolution` の場合、値は整数値であり、キーがそれ以外の場合、値はシンボルである。

「フォント `F` のフォントプロパティのうちキーが `@c Mxxx`

であるもの」のことを簡単に「F の xxx プロパティ」と呼ぶことがある。

foundry プロパティの値は、adobe, misc 等のフォントの開発元情報を示すシンボルである。

family プロパティの値は、times, helvetica 等のフォントファミリーを示すシンボルである。

weight プロパティの値は、normal, bold 等の太さに関する情報を示すシンボルである。

style プロパティの値は、normal, italic 等のスタイルに関する情報を示すシンボルである。

stretch プロパティの値は、normal, semicondensed 等の文字幅に関する情報を示すシンボルである。

adstyle プロパティの値は、serif, sans-serif 等の抽象的なフォントファミリーに関する情報を示すシンボルである。

registry プロパティの値は、iso10646, iso8895-1 等のレジストリ情報を示すシンボルである。

size プロパティの値は、フォントのデザインサイズを表わす整数値であり、単位は 1/10 ポイントである。

resolution プロパティの値は、想定されているデバイスの解像度を表わす整数値であり、単位は dots per inch (dpi) である。

type プロパティの値は、フォントドライバを指示し、現在 Mx もしくは Mfreetype である。

m17n ライブラリはフォントオブジェクトを2つの目的で用いている。アプリケーションプログラムからフォントの指定を受け取る目的と、アプリケーションプログラムに利用可能なフォントを提示する目的である。アプリケーションプログラムに対して提示を行う際には、フォントプロパティはすべて具体的な値を持つ。

m17n ライブラリは Window システムフォント、FreeType フォント、OpenType フォントの3種類をサポートしている。

 Window システムフォント

m17n x ライブラリは、x サーバと x フォントサーバが取り扱う全てのフォントをサポートする。XLFD の各フィールドとフォントプロパティの対応は以下の通り。この表にないフィールドは無視される。

XLFD フィールド	プロパティ
-----	-----
FOUNDRY	foundry
FAMILY_NAME	family
WEIGHT_NAME	weight
SLANT	style
SETWIDTH_NAME	stretch
ADD_STYLE_NAME	adstyle
PIXEL_SIZE	size
RESOLUTION_Y	resolution
CHARSET_REGISTRY-CHARSET_ENCODING	registry

 FreeType fonts

m17n ライブラリは、FreeType ライブラリを使うように設定された場合には、FreeType が扱うすべてのフォントをサポートする。変数 #mfont_freetype_path は m17n ライブラリの設定と環境変数 @c M17NDIR に応じて初期化される。詳細は変数の説明を参照のこと。

もし m17n ライブラリが fontconfig ライブラリを使うように設定された場合には、#mfont_freetype_path に加えて、fontconfig で使用可能なフォントもすべてサポートされる。

FreeType フォントのファミリー名は `family` プロパティに対応する。
 FreeType フォントのスタイル名は、下の表のように `weight`, `style`,
`stretch` プロパティに対応する。

スタイル名	weight	style	stretch
-----	-----	-----	-----
Regular	medium	r	normal
Italic	medium	i	normal
Bold	bold	r	normal
Bold Italic	bold	i	normal
Narrow	medium	r	condensed
Narrow Italic	medium	i	condensed
Narrow Bold	bold	r	condensed
Narrow Bold Italic	bold	i	condensed
Black	black	r	normal
Black Italic	black	i	normal
Oblique	medium	o	normal
BoldOblique	bold	o	normal

上の表に現われないスタイル名は "Regular" として扱われる。

platform ID と encoding ID の組み合わせが registry
 プロパティに対応する。たとえばあるフォントが (1 1) という ID の組合せを持てば、
 registry プロパティは 1-1 となる。頻繁にあらわれる組合せには以下のような定義済み
 registry プロパティ が与えられている。

platform ID	encoding ID	registry プロパティ
-----	-----	-----
0	3	unicode-bmp
0	4	unicode-full
1	0	apple-roman
3	1	unicode-bmp
3	1	unicode-full

したがって、二つの組合せ (1 0) 、(3 1) を持つフォントは、それぞれ
 registry プロパティが 1-0, apple-roman, 3-1, unicode-bmp
 である4つのフォントオブジェクトに対応する。

 OpenType フォント

m17n ライブラリは、FreeType ライブラリと OTF
 ライブラリを使用するように設定すれば、すべての OpenType
 フォントをサポートする。実際に利用できるフォントのリストは FreeType
 フォントの場合と同様に作られる。OpenType フォントを FLT (Font Layout Table)
 経由で使用するようフォントセットに指定されており、FLT に OTF
 関連のコマンド (たとえば `otf:deva`) があれば、OTF ライブラリがフォントの OpenType
 レイアウトテーブルに従って文字列をグリフコード列に変換し、FreeType
 ライブラリが各グリフのビットマップイメージを提供する。

2.19.2 関数詳解

2.19.2.1 mfont()

```
MFont* mfont ( )
```

新しいフォントを作る。

関数 **mfont()** はプロパティを一切持たない新しいフォントをオブジェクトを作る。

戻り値:

この関数は作ったフォントオブジェクトへのポインタを返す。

2.19.2.2 mfont_parse_name()

```
MFont* mfont_parse_name (
    const char * name,
    MSymbol format )
```

フォント名からフォントを作る。

関数 **mfont_parse_name()** は、フォント名 **name** から取り出されたプロパティを持つ、新しいフォントオブジェクトを作る。

format は **name** のフォーマットを指定する。**format** が **Mx** であれば、**name** は XLFD (X Logical Font Description) に従って解析される。**format** が **Mfontconfig** であれば **name** は **Fontconfig** のフォントテキスト表現に従って解析される。**format** が **Mnil** であれば、まず XLFD に従って解析され、それに失敗したら **Fontconfig** に従って解析される。

戻り値:

処理が成功すれば **mfont_parse_name()** は新しく作られたフォントへのポインタを返す。そうでなければ **NULL** を返す。

2.19.2.3 mfont_unparse_name()

```
char* mfont_unparse_name (
    MFont * font,
    MSymbol format )
```

フォントからフォント名を作る。

関数 **mfont_unparse_name()** はフォント名の文字列をフォント **font** を元に **format** に従って作る。

format は **Mx** または **Mfontconfig** である。**Mx** ならばフォント名は XLFD (X Logical Font Description) に従う。**Mfontconfig** ならばフォント名は **Fontconfig** のフォントテキスト表現に従う。

戻り値:

この関数は新たにアロケートしたフォント名の文字列を返す。文字列は、ユーザが **free()** によって明示的に解放しない限り解放されない。

2.19.2.4 mfont.copy()

```
MFont* mfont_copy (
    MFont * font )
```

フォントのコピーを作る。

関数 `Mfont.copy()` はフォント `font` のコピーを作り、それを返す。

2.19.2.5 mfont.get_prop()

```
void* mfont_get_prop (
    MFont * font,
    MSymbol key )
```

フォントのプロパティの値を得る。

関数 `mfont.get_prop()` はフォント `font` のプロパティのうち、キーが `key` であるものの値を返す。`key` は以下のシンボルのいずれかでなければならない。

`Mfoundry`, `Mfamily`, `Mweight`, `Mstyle`, `Mstretch`, `Madstyle`, `Mregistry`, `Msize`,
`Mresolution`, `Mspacing`.

戻り値:

`key` が `Mfoundry`, `Mfamily`, `Mweight`, `Mstyle`, `Mstretch`, `Madstyle`, `Mregistry`, `Mspacing` のいずれかであれば、相当する値をシンボルとして返す。フォントがそのプロパティを持たない場合には `Mnil` を返す。`key` が `Msize` あるいは `Mresolution` の場合には、相当する値を整数値として返す。フォントがそのプロパティを持たない場合には `0` を返す。`key` がそれ以外のものであれば、`NULL` を返し、外部変数 `merror_code` にエラーコードを設定する。

2.19.2.6 mfont.put_prop()

```
int mfont_put_prop (
    MFont * font,
    MSymbol key,
    void * val )
```

フォントのプロパティに値を設定する。

関数 `mfont.put_prop()` は、フォント `font` のキーが `key` であるプロパティの値を `val` に設定する。`key` は以下のシンボルのいずれかである。

`Mfoundry`, `Mfamily`, `Mweight`, `Mstyle`, `Mstretch`, `Madstyle`, `Mregistry`, `Msize`,
`Mresolution`.

`key` が `Msize` か `Mresolution` であれば `val` は整数値でなくてはならない。それ以外の場合、`val` はプロパティ値の名前のシンボルでなくてはならない。ただしもしその名前が `"nil"` の場合は、名前が `"Nil"` のシンボルでなくてはならない。

2.19.2.7 mfont_selection_priority()

```
MSymbol* mfont_selection_priority ( )
```

フォント選択の優先度を返す。

関数 [mfont_selection_priority\(\)](#) は 6 つのシンボルからなる配列を作って返す。配列の要素は、以下のフォントプロパティのキーを優先度順に並べたものである。

Mfamily, Mweight, Mstyle, Mstretch, Madstyle, Msize.

m17n ライブラリはこの配列に従って、最も合致するフォントを選択する。目的のフォントと、それぞれ違うプロパティの値が合致しないフォントがあった場合、優先度の低いプロパティの値が合致しないフォント（優先度の高いプロパティの値が合致しているフォント）が選択される。

2.19.2.8 mfont_set_selection_priority()

```
int mfont_set_selection_priority (
    MSymbol * keys )
```

フォント選択優先度を設定する。

関数 [mfont_set_selection_priority\(\)](#) は、6 つのシンボルの配列 `keys` にしたがってフォント選択優先度を設定する。配列は以下の各要素を適切な順番で並べたものである。

Mfamily, Mweight, Mstyle, Mstretch, Madstyle, Msize.

詳細は関数 [mfont_selection_priority\(\)](#) の説明を参照のこと。

2.19.2.9 mfont_find()

```
MFont* mfont_find (
    MFrame * frame,
    MFont * spec,
    int * score,
    int max_size )
```

フォントを探す。

関数 [mfont_find\(\)](#) は、フレーム `frame` 上でフォント定義 `spec` にもっとも合致する利用可能なフォントへのポインタを返す。

`score` は NULL であるか、見つかったフォントが `spec` にどれほど合っているかを示すスコアを保存する場所へのポインタである。スコアが小さいほど良く合っていることを意味する。

2.19.2.10 mfont_set_encoding()

```
int mfont_set_encoding (
    MFont * font,
    MSymbol encoding_name,
    MSymbol repertory_name )
```

フォントのエンコーディングを設定する。

関数 `mfont_set_encoding()` はフォント `font` のエンコーディング情報を設定する。

`encoding_name` はフォントと同じエンコーディングを持つ文字セットを示すシンボルである。

`repertory_name` は `Mnil` であるか、フォントと同じエンコーディングを持つ文字セットを示すシンボルである。 `Mnil` であれば、個々の文字がそのフォントでサポートされているかどうかは、各々のフォントドライバに問い合わせる。

戻り値:

処理が成功すればこの関数は `0` を返す。そうでなければ `-1` を返し、外部変数 `merror_code` にエラーコードを設定する。

2.19.2.11 mfont_name()

```
char* mfont_name (
    MFont * font )
```

フォント名からフォントを作る。

この関数は廃止予定である。 `mfont_unparse_name()` を使用のこと。

2.19.2.12 mfont_from_name()

```
MFont* mfont_from_name (
    const char * name )
```

フォントからフォント名を作る。

これは関数は廃止予定である。 `mfont_parse_name()` を使用のこと。

2.19.2.13 mfont_resize_ratio()

```
int mfont_resize_ratio (
    MFont * font )
```

フォントのリサイズ情報を得る

関数 `mfont_resize_ratio` は `m17n` データベース `<font,resize>` を検索し、フォント `FONT` のリサイズの比率（パーセンテージ）を返す。たとえば返す値が `150` であれば、`m17n` ライブラリは指定されたサイズの `1.5` 倍のフォントを使用することを意味する。

2.19.2.14 mfont_list()

```
MPlist* mfont_list (
    MFrame * frame,
    MFont * font,
    MSymbol language,
    int maxnum )
```

フォントのリストを得る

関数 `mfont_list()` はフレーム `frame` で利用可能なフォントのリストを返す。`font` が `NULL` でなければ、`font` と合致する利用可能なフォントのリストを返す。`language` が `Mnil` でなければ、`language` をサポートする利用可能なフォントのリストを返す。`maxnum` は、0 より大きい場合には、返すフォントの数の上限である。

ただし、引数 `language` は旧版との整合性のためだけにあり、その使用は勧められない。フォントの `Mlanguage` プロパティを使うべきである。もし `font` がすでにこのプロパティを持っていたら、引数 `language` は無

戻り値:

この関数はキーがフォントファミリー名であり値が `MFont` オブジェクトへのポインタであるような `plist` を返す。`plist` は `m17n_object_unref()` で解放する必要がある。フォントが見つからなければ `NULL` を返す。

2.19.2.15 mfont_list_family_names()

```
MPlist* mfont_list_family_names (
    MFrame * frame )
```

2.19.2.16 mfont_check()

```
int mfont_check (
    MFrame * frame,
    MFontset * fontset,
    MSymbol script,
    MSymbol language,
    MFont * font )
```

2.19.2.17 mfont_match_p()

```
int mfont_match_p (
    MFont * font,
    MFont * spec )
```

2.19.2.18 mfont_open()

```
MFont* mfont_open (
    MFrame * frame,
    MFont * font )
```

2.19.2.19 mfont_encapsulate()

```
MFont* mfont_encapsulate (
    MFrame * frame,
    MSymbol data_type,
    void * data )
```

2.19.2.20 mfont_close()

```
int mfont_close (
    MFont * font )
```

2.19.3 変数詳解

2.19.3.1 Mfoundry

`MSymbol` Mfoundry

開発元を指定するフォントプロパティのキー。

変数 `Mfoundry` は "foundry" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントの開発元名を名前として持つシンボルである。

2.19.3.2 Mfamily

`MSymbol` Mfamily

ファミリを指定するフォントプロパティのキー。

変数 `Mfamily` は "family" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントのファミリ名を名前として持つシンボルである。

2.19.3.3 Mweight

`MSymbol Mweight`

太さを指定するフォントプロパティのキー。

変数 `Mweight` は "weight" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントの太さ名 ("medium", "bold" 等) を名前として持つシンボルである。

2.19.3.4 Mstyle

`MSymbol Mstyle`

スタイルを指定するフォントプロパティのキー。

変数 `Mstyle` は "style" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントのスタイル名 ("r", "i", "o" 等) を名前として持つシンボルである。

2.19.3.5 Mstretch

`MSymbol Mstretch`

幅を指定するフォントプロパティのキー。

変数 `Mstretch` は "stretch" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントの文字幅名 ("normal", "condensed" 等) を名前として持つシンボルである。

2.19.3.6 Madstyle

`MSymbol Madstyle`

`adstyle` を指定するフォントプロパティのキー。

変数 `Madstyle` は "adstyle" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントの `adstyle` 名 ("serif", "", "sans" 等) を名前として持つシンボルである。

2.19.3.7 Mspacing

`MSymbol Mspacing`

`spacing` を指定するフォントプロパティのキー。

変数 `Mspacing` は "spacing" という名前を持つシンボルであり、フォントプロパティのキーとして用いられる。値は、フォントの `spacing` 特性を示す名前 ("p", "m" 等) を持つシンボルである。

2.19.3.8 Mregistry

`MSymbol Mregistry`

レジストリを指定するフォントプロパティのキー。

変数 `Mregistry` は "registry" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントのレジストリ名 ("iso8859-1", "jisx0208.1983-0" 等) を名前として持つシンボルである。

2.19.3.9 Msize

`MSymbol Msize`

サイズを指定するフォントプロパティのキー。

変数 `Msize` は "size" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、100 dpi のディスプレイ上でのフォントのデザインサイズを 1/10 ポイント単位で示す整数値である。

2.19.3.10 Motf

`MSymbol Motf`

2.19.3.11 Mfontfile

`MSymbol Mfontfile`

フォントファイルを指定するフォントプロパティのキー。

変数 `Mfontfile` は "fontfile" という名前を持つシンボルであり、フォントプロパティのキーとして用いられる。値は、フォントファイル名を名前として持つとするシンボルである。

2.19.3.12 Mresolution

`MSymbol Mresolution`

解像度を指定するフォントプロパティのキー。

変数 `Mresolution` は "resolution" という名前を持つシンボルであり、フォントプロパティとフェースプロパティのキーとして用いられる。値は、フォントの解像度を dots per inch (dpi) 単位で示す整数値である。

2.19.3.13 Mmax_advance

MSymbol Mmax_advance

2.19.3.14 Mfontconfig

MSymbol Mfontconfig

"fontconfig" という名前を持つシンボル.

変数 **Mfontconfig** は関数 `mfont_parse_name()` と `mfont_unparse_name()` の引数として用いられる。

2.19.3.15 Mx

MSymbol Mx

"x" という名前を持つシンボル.

変数 **Mx** は構造 **MDrawGlyph** のメンバ `<type>` の値として用いられ、メンバ `<fontp>` の型が実際には (`XFontStruct *`) であることを表す.

2.19.3.16 Mfreetype

MSymbol Mfreetype

"freetype" という名前を持つシンボル.

変数 **Mfreetype** は構造 **MDrawGlyph** のメンバ `<type>` の値として用いられ、メンバ `<fontp>` の型が実際には `FT_Face` であることを表す。

2.19.3.17 Mxft

MSymbol Mxft

"xft" という名前を持つシンボル.

変数 **Mxft** は構造 **MDrawGlyph** のメンバ `<type>` の値として用いられ、メンバ `<fontp>` の型が実際には (`XftFont *`) であることを表す。

2.20.2 関数詳解

2.20.2.1 mfontset()

```
MFontset * mfontset (
    char * name )
```

フォントセットを返す.

関数 `mfontset()` は名前 `name` を持つフォントセットオブジェクトへのポインタを返す。 `name` が `NULL` ならば、デフォルトフォントセットへのポインタを返す。

`name` という名前を持つフォントセットがなければ、新しいものが作られる。その際、 `m17n` データベースに `<fontset, name>` というデータがあれば、フォントセットはそのデータに沿って初期化される。なければ、空のままにされる。

マクロ `M17N_INIT()` はデフォルトのフォントセットを作る。アプリケーションプログラムは `mframe()` を初めて呼ぶまでの間はデフォルトフォントセットを変更することができる。

戻り値:

この関数は見つかった、あるいは作ったフォントセットへのポインタを返す。

2.20.2.2 mfontset_name()

```
MSymbol mfontset_name (
    MFontset * fontset )
```

フォントセットの名前を返す.

関数 `mfontset_name()` はフォントセット `fontset` の名前を返す。

2.20.2.3 mfontset_copy()

```
MFontset * mfontset_copy (
    MFontset * fontset,
    char * name )
```

フォントセットのコピーを作る.

関数 `mfontset_copy()` はフォントセット `fontset` のコピーを作って、名前 `name` を与え、そのコピーへのポインタを返す。 `name` は既存のフォントセットの名前であってはならない。そのような場合にはコピーを作らずに `NULL` を返す。

2.20.2.4 mfontset_modify_entry()

```
int mfontset_modify_entry (
    MFontset * fontset,
    MSymbol script,
    MSymbol language,
    MSymbol charset,
    MFont * spec,
    MSymbol layouter_name,
    int how )
```

フォントセットの内容を変更する。

関数 `mfontset_modify_entry()` は、`language` と `script` の組み合わせ、または `charset` に対して `font` のコピーを使うように、フォントセット `fontset` を設定する。

フォントセット中の各フォントは、特定のスクリプトと言語のペア、特定の文字セット、シンボル `Mnil` のいずれかと関連付けられている。同じものと関連付けられたフォントはグループを構成する。

`script` は `Mnil` であるか、スクリプトを特定するシンボルである。シンボルである場合には、`language` は言語を特定するシンボルか `Mnil` であり、`font` は the `script / language` ペアに関連付けられる。

`charset` は `Mnil` であるか、文字セットオブジェクトを表すシンボルである。シンボルである場合には `font` はその文字セットと関連付けられる。

`script` と `charset` の双方が `Mnil` でない場合には `font` のコピーが2つ作られ、それぞれ `script / language` ペアと文字セットに関連付けられる。

`script` と `charset` の双方が `Mnil` ならば、`font` は `Mnil` と関連付けられる。この種のフォントは `fallback font` と呼ばれる。

引数 `how` は `font` の優先度を指定する。`how` が正ならば、`font` は同じものと関連付けられたグループ中で最高の優先度を持つ。`how` が負ならば、最低の優先度を持つ。`how` が 0 ならば、`font` は関連付けられたものに対する唯一の利用可能なフォントとなり、他のフォントはグループから取り除かれる。

`layouter_name` は `Mnil` であるか、[フォントレイアウトテーブル](#)（フォントレイアウトテーブル）を示すシンボルである。シンボルであれば、`font` を用いて `M-text` を表示する際には、そのフォントレイアウトテーブルを使って文字列からグリフコード列を生成する。

戻り値:

処理が成功したとき、`mfontset_modify_entry()` は 0 を返す。失敗したときは -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_SYMBOL`

2.20.2.5 mfontset.Lookup()

```
MPlist * mfontset.lookup (
    MFontset * fontset,
    MSymbol script,
    MSymbol language,
    MSymbol charset )
```

フォントセットを検索する。

関数 `mfontset.Lookup()` は `fontset` を検索し、`fontset` の内容のうち指定したスクリプト、言語、文字セットに対応する部分を表す `plist` を返す。

`script` が `Mt` ならば、返す `plist` のキーはフォントが指定されているスクリプト名のシンボルであり、値は `NULL` である。

`script` がスクリプト名のシンボルであれば、返す `plist` は `language` によって定まる。

- `language` が `Mt` ならば、`plist` のキーはフォントが指定されている言語名のシンボルであり、値は `NULL` である。キーは `Mt` であることもあり、その場合そのスクリプトにフォールバックフォントがあることを意味する。
- `language` が言語名のシンボルならば、`plist` は指定のスクリプトと言語に対する `FONT-GROUP` である。`FONT-GROUP` とは、キーが `FLT` (`FontLayoutTable`) 名のシンボルであり、値が `MFont` へのポインタであるような `plist` である。ただしフォントに `FLT` が対応付けられていない時には、キーは `Mt` になる。
- `language` が `Mnil` ならば、`plist` はそのスクリプト用のフォールバック `FONT-GROUP` である。

`script` が `Mnil` ならば、返す `plist` は以下のように定まる。

- `charset` が `Mt` ならば、`plist` のキーはフォントが指定されている文字セット名のシンボルであり、値は `NULL` である。
- `charset` が文字セット名のシンボルならば、`plist` はその文字セット用の `FONT-GROUP` である。
- `charset` が `Mnil` ならば、`plist` はフォールバック `FONT-GROUP` である。

戻り値:

この関数はフォントセットの内容を表す `plist` を返す。`plist` は `m17n_object_unref()` で解放されるべきである。

2.21 フェース

フェースとは、M-text の見栄えを制御するオブジェクトである。

フェース 連携図



データ構造

- struct [MFaceHLineProp](#)
フェースの水平線指定用型宣言.
- struct [MFaceBoxProp](#)
フェースの囲み枠指定用型宣言.

型定義

- typedef void(* [MFaceHookFunc](#)) ([MFace](#) *face, void *arg, void *info)
フェースのフック関数の型宣言.

関数

- [MFace](#) * [mface](#) ()
新しいフェースをつくる.
- [MFace](#) * [mface_copy](#) ([MFace](#) *face)
フェースのコピーを作る.
- int [mface_equal](#) ([MFace](#) *face1, [MFace](#) *face2)
- [MFace](#) * [mface_merge](#) ([MFace](#) *dst, [MFace](#) *src)
フェースを統合する.
- [MFace](#) * [mface_from_font](#) ([MFont](#) *font)
フォントからフェースを作る.
- void * [mface_get_prop](#) ([MFace](#) *face, [MSymbol](#) key)
フェースのプロパティの値を得る.
- [MFaceHookFunc](#) [mface_get_hook](#) ([MFace](#) *face)
フェースのフック関数を得る.
- int [mface_put_prop](#) ([MFace](#) *face, [MSymbol](#) key, void *val)
フェースプロパティの値を設定する.
- int [mface_put_hook](#) ([MFace](#) *face, [MFaceHookFunc](#) func)
フェースのフック関数を設定する.
- void [mface_update](#) ([MFrame](#) *frame, [MFace](#) *face)
フェースを更新する.

変数: フェースプロパティのキー

- [MSymbol](#) [Mforeground](#)
前景色を指定するフェースプロパティのキー.
- [MSymbol](#) [Mbackground](#)
背景色を指定するためのフェースプロパティのキー.
- [MSymbol](#) [Mvideomode](#)
ビデオモードを指定するためのフェースプロパティのキー.
- [MSymbol](#) [Mratio](#)
フォントのサイズの比率を指定するためのフェースプロパティのキー.
- [MSymbol](#) [Mhline](#)
水平線を指定するためのフェースプロパティのキー.
- [MSymbol](#) [Mbox](#)

囲み枠を指定するためのフェースプロパティのキー。

- [MSymbol Mfontset](#)

フォントセットを指定するためのフェースプロパティのキー。

- [MSymbol Mhook_func](#)

フックを指定するためのフェースプロパティのキー。

- [MSymbol Mhook_arg](#)

フックの引数を指定するためのフェースプロパティのキー。

変数： フェースの **#Mvideomode** プロパティの可能な値

変数 [Mvideomode](#) の説明を参照のこと。

- [MSymbol Mnormal](#)
- [MSymbol Mreverse](#)

変数: 定義済みフェース

- [MFace * mface_normal_video](#)
標準ビデオフェース。
- [MFace * mface_reverse_video](#)
リバーズビデオフェース。
- [MFace * mface_underline](#)
- [MFace * mface_medium](#)
ミディアムフェース。
- [MFace * mface_bold](#)
ボールドフェース。
- [MFace * mface_italic](#)
イタリックフェース。
- [MFace * mface_bold_italic](#)
ボールドイタリックフェース。
- [MFace * mface_xx_small](#)
最小のフェース。
- [MFace * mface_x_small](#)
より小さいフェース。
- [MFace * mface_small](#)
小さいフェース。
- [MFace * mface_normalsize](#)
標準の大きさのフェース。
- [MFace * mface_large](#)
大きいフェース。
- [MFace * mface_x_large](#)
もっと大きいフェース。
- [MFace * mface_xx_large](#)
最大のフェース。
- [MFace * mface_black](#)
黒フェース。

- `MFace * mface_white`
白フェース.
- `MFace * mface_red`
赤フェース.
- `MFace * mface_green`
緑フェース.
- `MFace * mface_blue`
青フェース.
- `MFace * mface_cyan`
シアンフェース.
- `MFace * mface_yellow`
黄フェース.
- `MFace * mface_magenta`
マゼンタフェース.

変数: フェースを取り扱うためのその他のシンボル

- `MSymbol Mface`
フェースを指定するテキストプロパティのキー.

2.21.1 詳解

フェースとは、`M-text` の見栄えを制御するオブジェクトである.

フェースは `MFace` 型のオブジェクトであり、`M-text` の表示方法を制御する。フェースは固定個の フェースプロパティを持つ。他のプロパティ同様フェースプロパティはキーと値からなり、キーは以下のシンボルのいずれかである。

`Mforeground`, `Mbackground`, `Mvideomode`, `Mhline`, `Mbox`, `Mfoundry`, `Mfamily`, `Mweight`, `Mstyle`, `Mstretch`, `Madstyle`, `Msize`, `Mfontset`, `Mratio`, `Mhook.func`, `Mhook.arg`

「フェース `F` のフェースプロパティのうちキーが `Mxxx` であるもの」のことを簡単に「`F` の `xxx` プロパティ」と呼ぶことがある。

`M-text` の表示関数は、まず最初にその `M-text` からキーがシンボル `Mface` であるようなテキストプロパティを探し、次にその値に従って `M-text` を表示する。この値はフェースオブジェクトへのポインタでなければならない。

`M-text` が、`::Mface` をキーとするテキストプロパティを複数持っており、かつそれらの値が衝突しないならば、フェース情報は組み合わされて用いられる。

あるテキスト属性がどのフェースによっても指定されていない場合は、デフォルトフェースの値が用いられる。

2.21.2 型定義詳解

2.21.2.1 MFaceHookFunc

```
typedef void(* MFaceHookFunc) (MFace *face, void *arg, void *info)
```

フェースのフック関数の型宣言.

MFaceHookFunc はフェースのフック関数の型である。

2.21.3 関数詳解

2.21.3.1 mface()

```
MFace* mface ( )
```

新しいフェースをつくる.

関数 **mface()** はプロパティを一切持たない新しいフェースオブジェクトを作る。

戻り値:

この関数は作ったフェースへのポインタを返す。

2.21.3.2 mface_copy()

```
MFace* mface_copy (
    MFace * face )
```

フェースのコピーを作る.

関数 **mface_copy()** はフェース **face** のコピーを作り、そのコピーへのポインタを返す。

2.21.3.3 mface_equal()

```
int mface_equal (
    MFace * face1,
    MFace * face2 )
```

2.21.3.4 mface_merge()

```
MFace* mface_merge (
    MFace * dst,
    MFace * src )
```

フェースを統合する。

関数 [mface_merge\(\)](#) は、フェース `src` のプロパティをフェース `dst` に統合する。

戻り値:

この関数は `dst` を返す。

2.21.3.5 mface_from_font()

```
MFace* mface_from_font (
    MFont * font )
```

フォントからフェースを作る。

関数 [mface_from_font\(\)](#) はフォント `font` のプロパティをプロパティとして持つ新しいフェースを作り、それを返す。

2.21.3.6 mface_get_prop()

```
void* mface_get_prop (
    MFace * face,
    MSymbol key )
```

フェースのプロパティの値を得る。

関数 [mface_get_prop\(\)](#) は、フェース `face` が持つフェースプロパティの内、キーが `key` であるものの値を返す。`key` は下記のいずれかでなければならない。

```
#Mforeground, #Mbackground, #Mvideomode, #Mhline, #Mbox,
#Mfoundry, #Mfamily, #Mweight, #Mstyle, #Mstretch, #Madstyle,
#Msize, #Mfontset, #Mratio, #Mhook_arg
```

戻り値:

戻り値の型は `key` に依存する。上記のキーの説明を参照すること。エラーが検出された場合は `NULL` を返し、外部変数 [merror.code](#) にエラーコードを設定する。

参照:

[mface_put_prop\(\)](#), [mface_put_hook\(\)](#)

エラー:

`MERROR_FACE`

2.21.3.7 mface_get_hook()

```
MFaceHookFunc mface_get_hook (
    MFace * face )
```

フェースのフック関数を得る。

関数 [mface_get_hook\(\)](#) はフェース **face** のフック関数を返す。

2.21.3.8 mface_put_prop()

```
int mface_put_prop (
    MFace * face,
    MSymbol key,
    void * val )
```

フェースプロパティの値を設定する。

関数 [mface_put_prop\(\)](#) は、フェース **face** 内でキーが **key** であるプロパティの値を **val** に設定する。**key** は以下のいずれかでなくてはならない。

```
#Mforeground, #Mbackground, #Mvideomode, #Mhline, #Mbox,
#Mfoundry, #Mfamily, #Mweight, #Mstyle, #Mstretch, #Madstyle,
#Msize, #Mfontset, #Mratio, #Mhook_func, #Mhook_arg.
```

これらのうちの、フォント関連のプロパティ ([Mfamily](#) から [Msize](#) まで) は、フェースのフォントセット中のフォントに関するデフォルト値となり、個々のフォントが値を指定しなかった場合に用いられる。

戻り値の型は **key** に依存する。上記のキーの説明を参照すること。

戻り値:

処理が成功した場合、[mface_put_prop\(\)](#) は 0 を返す。失敗した場合は -1 を返し、外部変数 [merror_code](#) にエラーコードを設定する。

参照:

[mface_get_prop\(\)](#)

エラー:

MERROR_FACE

2.21.3.9 mface_put_hook()

```
int mface_put_hook (
    MFace * face,
    MFaceHookFunc func )
```

フェースのフック関数を設定する。

関数 [mface_set_hook\(\)](#) は、フェース **face** のフック関数を **func** に設定する。

2.21.3.10 mface_update()

```
void mface_update (
    MFrame * frame,
    MFace * face )
```

フェースを更新する。

関数 `mface_update()` はフレーム `frame` のフェース `face` を `face` のフック関数を（あれば）呼んで更新する。

2.21.4 変数詳解

2.21.4.1 Mforeground

`MSymbol` Mforeground

前景色を指定するフェースプロパティのキー。

変数 `Mforeground` はフェースプロパティのキーとして用いられる。プロパティの値は、色名を名前として持つシンボルか `Mnil` である。

`Mnil` の場合、前景色は指定されない。そうでなければ `M-text` の前景は指定された色で表示される。

2.21.4.2 Mbackground

`MSymbol` Mbackground

背景色を指定するためのフェースプロパティのキー。

変数 `Mbackground` はフェースプロパティのキーとして用いられる。プロパティの値は、色名を名前として持つシンボルか `Mnil` である。

`Mnil` の場合、背景色は指定されない。そうでなければ `M-text` の背景は指定された色で表示される。

2.21.4.3 Mvideomode

`MSymbol` Mvideomode

ビデオモードを指定するためのフェースプロパティのキー。

変数 `Mvideomode` はフェースプロパティのキーとして用いられる。プロパティの値は、`Mnormal`, `Mreverse`, `Mnil` のいずれかでなくてはならない。

`Mnormal` の場合は、`M-text` を標準のビデオモード（前景を前景色で、背景を背景色で）で表示する。

`Mreverse` の場合はリバースビデオモードで（前景を背景色で、背景を前景色で）表示する。

`Mnil` の場合はビデオモードは指定されない。

2.21.4.4 Mratio

MSymbol Mratio

フォントのサイズの比率を指定するためのフェースプロパティのキー。

変数 **Mratio** はフェースプロパティのキーとして用いられる。値 **RATIO** は整数値でなくてはならない。

値が **0** ならば、フォントサイズは指定されない。そうでなければ、**M-text** は $(\text{FONTSIZE} * \text{RATIO} / 100)$ というサイズのフォントで表示される。**FONTSIZE** はフェースプロパティ **::Msize** で指定されたサイズである。

2.21.4.5 Mhline

MSymbol Mhline

水平線を指定するためのフェースプロパティのキー。

変数 **Mhline** はフェースプロパティのキーとして用いられる。値は **MFaceHLineProp** 型オブジェクトへのポインタか **NULL** でなくてはならない。

値が **NULL** ならば、このプロパティは指定されない。そうでなければ値が指すオブジェクトに指定されたように水平線を付加して **M-text** を表示する。

2.21.4.6 Mbox

MSymbol Mbox

囲み枠を指定するためのフェースプロパティのキー。

変数 **Mbox** はフェースプロパティのキーとして用いられる。値は **MFaceBoxProp** 型オブジェクトへのポインタか **NULL** でなくてはならない。

値が **NULL** ならば、このフェースは囲み枠を指定していない。そうでなければ値が指すオブジェクトに指定されたように囲み枠を付加して **M-text** を表示する。

2.21.4.7 Mfontset

MSymbol Mfontset

フォントセットを指定するためのフェースプロパティのキー。

変数 **Mfontset** はフェースプロパティのキーとして用いられる。値は **Mfontset** 型オブジェクトへのポインタか **NULL** でなくてはならない。

値が **NULL** ならば、フォントセットは指定されていない。そうでなければ値が指すオブジェクトに指定されたフォントセットから選んだフォントで **M-text** を表示する。

2.21.4.8 Mhook_func

`MSymbol Mhook_func`

フックを指定するためのフェースプロパティのキー。

変数 `Mhook_func` はフェースプロパティのキーとして用いられる。値は `MFaceHookFunc` 型の関数か `NULL` でなくてはならない。

値が `NULL` ならば、フックは指定されていない。そうでなければフェースを実現する前に指定された関数が呼ばれる。

2.21.4.9 Mhook_arg

`MSymbol Mhook_arg`

フックの引数を指定するためのフェースプロパティのキー。

変数 `Mhook_arg` はフェースプロパティのキーとして用いられる。値は何でもよく、フェースプロパティ `Mhook_func` で指定される関数に渡される。

2.21.4.10 Mnormal

`MSymbol Mnormal`

2.21.4.11 Mreverse

`MSymbol Mreverse`

2.21.4.12 mface_normal_video

`MFace* mface_normal_video`

標準ビデオフェース。

変数 `mface_normal_video` は `Mvideomode` プロパティの値が `Mnormal` であるフェースを指すポインタである。他のプロパティは指定されない。このフェースで表示される `M-text` は標準の色 (すなわち前景は前景色、背景は背景色) で描かれる。

2.21.4.13 mface_reverse_video

```
MFace* mface_reverse_video
```

リバースビデオフェース.

変数 `mface_reverse_video` は `Mvideomode` プロパティの値が `Mreverse` であるフェースを指すポインタである。他のプロパティは指定されない。このフェースで表示される `M-text` は前景色と背景色が入れ替わって(すなわち前景は背景色、背景は前景色) 描かれる。

2.21.4.14 mface_underline

```
MFace* mface_underline
```

@brief 下線フェース.

変数 `#mface_underline` は `#Mhline` プロパティの値が `#MFaceHLineProp` 型オブジェクトへのポインタであるフェースを指すポインタである。オブジェクトのメンバは以下の通り。

メンバ	値
-----	-----
type	MFACE_HLINE_UNDER
width	1
color	Mnil

他のプロパティは指定されない。このフェースを持つ `M-text` は下線付きで表示される。

2.21.4.15 mface_medium

```
MFace* mface_medium
```

ミディアムフェース.

変数 `mface_medium` は `Mweight` プロパティの値が "medium" という名前をもつシンボルであるようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ `M-text` は、ミディアムウェイトのフォントで表示される。

2.21.4.16 mface_bold

```
MFace* mface_bold
```

ボールドフェース.

変数 `mface_bold` は `Mweight` プロパティの値が "bold" という名前をもつシンボルであるようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ `M-text` は、ボールドフォントで表示される。

2.21.4.17 mface_italic

`MFace* mface_italic`

イタリックフェース.

変数 `mface_italic` は `Mstyle` プロパティの値が "italic" という名前をもつシンボルであるようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ **M-text** は、イタリック体で表示される。

2.21.4.18 mface_bold_italic

`MFace* mface_bold_italic`

ボールドイタリックフェース.

変数 `mface_bold_italic` は、`::Mweight` プロパティの値が "bold" という名前をもつシンボルであり、かつ `Mstyle` プロパティの値が "italic" という名前をもつシンボルであるようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ **M-text** は、ボールドイタリック体で表示される。

2.21.4.19 mface_xx_small

`MFace* mface_xx_small`

最小のフェース.

変数 `mface_xx_small` は、`::Mratio` プロパティの値が 50 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ **M-text** は標準フォントの 50% の大きさのフォントを用いて表示される。

2.21.4.20 mface_x_small

`MFace* mface_x_small`

より小さいフェース.

変数 `mface_x_small` は、`::Mratio` プロパティの値が 66 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ **M-text** は標準フォントの 66% の大きさのフォントを用いて表示される。

2.21.4.21 mface_small

`MFace* mface_small`

小さいフェース.

変数 `mface_small` は、`::Mratio` プロパティの値が 75 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ **M-text** は標準フォントの 75% の大きさのフォントを用いて表示される。

2.21.4.22 mface_normalsize

`MFace* mface_normalsize`

標準の大きさのフェース.

変数 `mface_normalsize` は、`::Mratio` プロパティの値が 100 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は標準フォントと同じ大きさのフォントを用いて表示される。

2.21.4.23 mface_large

`MFace* mface_large`

大きいフェース.

変数 `mface_large` は、`::Mratio` プロパティの値が 120 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は標準フォントの 120% の大きさのフォントを用いて表示される。

2.21.4.24 mface_x_large

`MFace* mface_x_large`

もっと大きいフェース.

変数 `mface_x_large` は、`::Mratio` プロパティの値が 150 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は標準フォントの 150% の大きさのフォントを用いて表示される。

2.21.4.25 mface_xx_large

`MFace* mface_xx_large`

最大のフェース.

変数 `mface_xx_large` は、`::Mratio` プロパティの値が 200 であるフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は標準フォントの 200% の大きさのフォントを用いて表示される。

2.21.4.26 mface_black

`MFace* mface_black`

黒フェース.

変数 `mface_black` は、`::Mforeground` プロパティの値として "black" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色として黒を用いて表示される。

2.21.4.27 mface_white

```
MFace* mface_white
```

白フェース.

変数 `mface.white` は、`::Mforeground` プロパティの値として "white" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色として白を用いて表示される。

2.21.4.28 mface_red

```
MFace* mface_red
```

赤フェース.

変数 `mface.red` は、`::Mforeground` プロパティの値として "red" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色として赤を用いて表示される。

2.21.4.29 mface_green

```
MFace* mface_green
```

緑フェース.

変数 `mface.green` は、`::Mforeground` プロパティの値として "green" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色として緑を用いて表示される。

2.21.4.30 mface_blue

```
MFace* mface_blue
```

青フェース.

変数 `mface.blue` は、`::Mforeground` プロパティの値として "blue" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色として青を用いて表示される。

2.21.4.31 mface_cyan

```
MFace* mface_cyan
```

シアンフェース.

変数 `mface.cyan` は、`::Mforeground` プロパティの値として "cyan" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ M-text は前景色としてシアンを用いて表示される。

2.21.4.32 mface_yellow

```
MFace* mface_yellow
```

黄フェース.

変数 `mface_yellow` は、`::Mforeground` プロパティの値として "yellow" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ `M-text` は前景色として黄色を用いて表示される。

2.21.4.33 mface_magenta

```
MFace* mface_magenta
```

マゼンタフェース.

変数 `mface_magenta` は、`::Mforeground` プロパティの値として "magenta" という名前のシンボルを持つようなフェースを指すポインタである。他のプロパティは指定されない。このフェースを持つ `M-text` は前景色としてマゼンタを用いて表示される。

2.21.4.34 Mface

```
MSymbol Mface
```

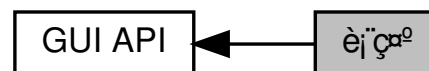
フェースを指定するテキストプロパティのキー.

変数 `Mface` は "face" という名前を持つシンボルである。このシンボルをキーとするテキストプロパティは、`MFace` 型のオブジェクトへのポインタを持たなければならない。これは管理キーである。

2.22 表示

`M-text` をウィンドウに描画する.

表示 連携図



データ構造

- struct `MDrawControl`
テキスト表示制御の型宣言.
- struct `MDrawMetric`
グリフとテキストの寸法の型宣言.
- struct `MDrawGlyphInfo`
グリフに関する情報の型宣言.
- struct `MDrawGlyph`
グリフの寸法とフォントに関する情報の型宣言.

型定義

- `typedef void * MDrawWindow`
ウィンドウシステムに依存する、ウィンドウの型宣言.
- `typedef void * MDrawRegion`
ウィンドウシステムに依存する、領域の型宣言.

関数

- `int mdraw_text (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to)`
ウィンドウに M-text を描画する.
- `int mdraw_image_text (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to)`
ディスプレイに M-text を画像として描く.
- `int mdraw_text_with_control (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to, MDrawControl *control)`
ディスプレイに M-text を詳細な制御つきで描く.
- `int mdraw_text_extents (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawMetric *overall_link_return, MDrawMetric *overall_logical_return, MDrawMetric *overall_line_return)`
テキストの幅 (ピクセル単位) を計算する.
- `int mdraw_text_per_char_extents (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawMetric *ink_array_return, MDrawMetric *logical_array_return, int array_size, int *num_chars_return, MDrawMetric *overall_link_return, MDrawMetric *overall_logical_return)`
M-text の各文字の表示範囲を計算する.
- `int mdraw_coordinates_position (MFrame *frame, MText *mt, int from, int to, int x_offset, int y_offset, MDrawControl *control)`
指定した座標に最も近い文字の文字位置を得る.
- `int mdraw_glyph_info (MFrame *frame, MText *mt, int from, int pos, MDrawControl *control, MDrawGlyphInfo *info)`
グリフに関する情報を計算する.
- `int mdraw_glyph_list (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawGlyph *glyphs, int array_size, int *num_glyphs_return)`
グリフ列に関する情報を計算する.
- `void mdraw_text_items (MFrame *frame, MDrawWindow win, int x, int y, MDrawTextItem *items, int nitems)`
textitem を表示する.
- `int mdraw_default_line_break (MText *mt, int pos, int from, int to, int line, int y)`
改行位置を計算する.
- `void mdraw_per_char_extents (MFrame *frame, MText *mt, MDrawMetric *array_return, MDrawMetric *overall_return)`
M-text の文字毎の表示範囲情報を得る.
- `void mdraw_clear_cache (MText *mt)`
キャッシュ情報を消す.

変数

- `int mdraw_line_break_option`

2.22.1 詳解

M-text をウィンドウに描画する。

m17n-gui API には、M-text を表示するための関数が用意されている。

表示に用いられるフォントは、フォントセットと face のプロパティに基づいて自動的に決定される。また、フォントのサイズや色や下線などの見栄えも face によって決まる。

M-text の描画フォーマットは多様な方法で制御できるので、強力な二次元レイアウト機能が実現できる。

2.22.2 型定義詳解

2.22.2.1 MDrawWindow

```
typedef void* MDrawWindow
```

ウィンドウシステムに依存する、ウィンドウの型宣言。

MDrawWindow はウィンドウ、すなわち幾つかの点でスクリーンのミニチュアとして働く矩形領域用の型である。

実際に何を指すかはウィンドウシステムに依存する。m17n X ライブラリを利用するプログラムは Drawable 型をこの型に変換しなくてはならない。

2.22.2.2 MDrawRegion

```
typedef void* MDrawRegion
```

ウィンドウシステムに依存する、領域の型宣言。

MDrawRegion は領域、すなわちスクリーン上の任意のピクセルの集合（典型的には矩形領域）用の型である。

実際に何を指すかはウィンドウシステムに依存する。m17n X ライブラリを利用するプログラムは Region 型をこの型に変換しなくてはならない。

2.22.3 関数詳解

2.22.3.1 mdraw_text()

```
int mdraw_text (
    MFrame * frame,
    MDrawWindow win,
    int x,
    int y,
    MText * mt,
    int from,
    int to )
```

ウィンドウに M-text を描画する。

関数 **mdraw_text()** は、フレーム **frame** のウィンドウ **win** の座標 (x, y) に、M-text **mt** の **from** から **to** までのテキストを描画する。

テキストの見栄え（フォント、スタイル、色など）は、キーが **Mface** であるテキストプロパティの値によって決まる。M-text の一部あるいは全部にそのようなテキストプロパティが付いていない場合には、**frame** のデフォルトフェースを代わりに用いる。

M-text の各文字を表示するフォントは、フェースの **fontset** プロパティの値のうちから、以下のアルゴリズムで選ばれる。

1. その文字のテキストプロパティのうち、キーが **Mcharset** であるものの値を調べる。この値は文字セットを表わすシンボルか **Mnil** のどちらかである。::Mnil ならば、次のステップに進む。そうでなければ、**fontset** のマッピングテーブルにその文字セット用のフォントがあるかどうかを調べる。無ければ、次のステップに進む。

その文字セット用のフォントがみつければ、それらのうち現在の文字用のグリフを持ち、フェースの各プロパティに最もよく合致するものを使う。

そのようなフォントが無ければ次のステップに進む。

2. その文字の文字プロパティ **"script"**（スクリプト）を調べる。そのプロパティが継承されているならばそれ以前の文字の文字プロパティ **"script"** を調べる。前の文字がなかったり、その文字プロパティを持っていなかった場合には、次のステップに進む。

その文字のテキストプロパティのうち、キーが **@c Mlanguage** であるものの値を調べる。

この値は言語を表わすシンボルか **Mnil** のいずれかである。

その言語とスクリプトの組み合わせが **fontset** のマッピングテーブルにあるかどうかを調べる。見つからなければ次のステップに進む。

見つかったばあいには、それらのフォントのうち現在の文字用のグリフを持ち、フェースの各プロパティに最もよく合致しているものを使う。そのようなフォントが無ければ次のステップに進む。

3. その文字のグリフを持つフォントを、フォントセットの **fall-back** テーブルから探す。フォントが見つければそれを使う。

以上のアルゴリズムでフォントが見つからなければ、この関数はその文字として空の四角形を表示する。

この関数が描画するのはグリフの前景だけである。背景色を指定するには、関数 **mdraw_image_text()** か関数 **mdraw_text_with_control()** を使うこと。

この関数は、X ウィンドウにおける関数 **XDrawString()**, **XmbDrawString()**, **XwcDrawString()** に相当する。

戻り値:

処理が成功した場合、`mdraw_text()` は 0 を返す。エラーが検出された場合は -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`

参照:

[mdraw_image_text\(\)](#)

2.22.3.2 mdraw_image_text()

```
int mdraw_image_text (
    MFrame * frame,
    MDrawWindow win,
    int x,
    int y,
    MText * mt,
    int from,
    int to )
```

ディスプレイにM-text を画像として描く。

関数 [mdraw_image_text\(\)](#) は、フレーム `frame` のウィンドウ `win` の座標 `(x, y)` に、M-text `mt` の `from` から `to` までのテキストを画像として描く。

テキストの描画方法は [mdraw_text\(\)](#) とほぼ同じであるが、この関数ではフェースで指定された色で背景も描く点が異なっている。

この関数は、X ウィンドウにおける `XDrawImageString()`, `XmbDrawImageString()`, `XwcDrawImageString()` に相当する。

戻り値:

処理が成功した場合、`mdraw_image_text()` は 0 を返す。エラーが検出された場合は -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`

参照:

[mdraw_text\(\)](#)

2.22.3.3 mdraw_text_with_control()

```
int mdraw_text_with_control (
    MFrame * frame,
    MDrawWindow win,
    int x,
    int y,
    MText * mt,
    int from,
    int to,
    MDrawControl * control )
```

ディスプレイにM-textを詳細な制御つきで描く。

関数 `mdraw_text_with_control()` は、フレーム `frame` のウィンドウ `win` の座標 `(x, y)` に、M-text `mt` の `from` から `to` までのテキストを描く。

テキストの描画方法は `mdraw_text()` とほぼ同じであるが、この関数は描画制御用のオブジェクト `control` の指示にも従う点が異なっている。

たとえば `control` の `<two_dimensional>` がゼロでなければ、この関数は M-text を 2 次元的に描く。すなわち M-text 中の改行で行を改め、続く文字は次の行に描く。詳細は構造体 @ `MDrawControl` の説明を参照すること。

2.22.3.4 mdraw_text_extents()

```
int mdraw_text_extents (
    MFrame * frame,
    MText * mt,
    int from,
    int to,
    MDrawControl * control,
    MDrawMetric * overall_link_return,
    MDrawMetric * overall_logical_return,
    MDrawMetric * overall_line_return )
```

テキストの幅（ピクセル単位）を計算する。

関数 `mdraw_text_extents()` は、関数 `mdraw_text_with_control()` が描画制御オブジェクト `control` を用いて M-text `mt` の `from` から `to` までをフレーム `frame` に表示する際に必要となる幅を返す。

`overallLink_return` が NULL でなければ、この関数は M-text の文字のインクのバウンディングボックスも計算し、`overallLink_return` が指す構造体のメンバにその結果を設定する。M-text に囲み枠 (surrounding box) を指定するフェースがあれば、それもバウンディングボックスに含む。

`overallLogical_return` が NULL でなければ、この関数は M-text と他の graphical feature（囲み枠など）との間の最小のスペースを示すバウンディングボックスも計算し、`overallLogical_return` が指す構造体のメンバにその結果を設定する。

`overallJline_return` が NULL でなければ、この関数は他の M-text との間の最小のスペースを示すバウンディングボックスも計算し、`overallJline_return` が指す構造体のメンバにその結果を設定する。オブジェクト `control` のメンバ `min_line_ascent`, `min_line_descent`, `max_line_ascent`, `max_line_descent` がすべて 0 の時には、この値は `overallLink_return` と `overallLogical_return` の和となる。

戻り値:

この関数は表示に必要なテキストの幅をピクセル単位で返す。`control->two_dimensional` が 0 でなく、テキストが複数の行に渡って描かれる場合には、最大の幅を返す。エラーが生じた場合は -1 を返し、外部変数 `merror_code` にエラーコードを設定する。

エラー:

`MERROR_RANGE`

2.22.3.5 mdraw_text_per_char_extents()

```
int mdraw_text_per_char_extents (
    MFrame * frame,
    MText * mt,
    int from,
    int to,
    MDrawControl * control,
    MDrawMetric * ink_array_return,
    MDrawMetric * logical_array_return,
    int array_size,
    int * num_chars_return,
    MDrawMetric * overall_ink_return,
    MDrawMetric * overall_logical_return )
```

M-text の各文字の表示範囲を計算する。

関数 `mdraw_text_per_char_extents()` は、関数 `mdraw_text_with_control()` が描画制御オブジェクト `control` を用いて M-text `mt` の `from` から `to` までをフレーム `frame` に表示する際の各文字のサイズを計算する。

`array_size` によって `ink_array_return` と `logical_array_return` のサイズを指定する。`ink_array_return` と `logical_array_return` の各要素は、それぞれ文字の描画インクと論理サイズ（M-text の表示原点からの相対位値）によって順に埋められる。設定された `ink_array_return` と `logical_array_return` の要素の数は、`num_chars_return` に戻される。

`array_size` がすべての寸法を戻せないほど小さい場合には、関数は -1 を返し、必要な大きさを `num_chars_return` に返す。そうでなければ 0 を返す。

ポインタ `overall_ink_return` と `overall_logical_return` が NULL でなければ、この関数はテキスト全体のサイズも計算し、結果を `overall_ink_return` と `overall_logical_return` で指される構造のメンバに保存する。

`control->two_dimensional` が 0 でなければ、この関数は最初の行の文字のサイズだけを計算する。

2.22.3.6 mdraw_coordinates_position()

```
int mdraw_coordinates_position (
    MFrame * frame,
    MText * mt,
    int from,
    int to,
    int x_offset,
    int y_offset,
    MDrawControl * control )
```

指定した座標に最も近い文字の文字位置を得る。

関数 `mdraw_coordinates_position()` は、関数 `mdraw_text_with_control()` が描画制御オブジェクト `control` を用いて、M-text `mt` の `from` から `to` までを座標 (0, 0) を起点として描画する際に、座標 (x, y) に描画される文字の文字位置を返す。ここで文字位置とは、当該 M-text 中においてその文字が最初から何番目かを示す整数である。ただし最初の文字の文字位置は 0 とする。

`frame` はデフォルトのフェースの情報を得るために用いられる。

戻り値:

座標 (x, y) がある文字のグリフで覆われる場合、関数 `mdraw_coordinates_position()` はその文字の文字位置を返す。

もし y が描画領域の最小Y 座標よりも小さいならば from を返す。

もし y が描画領域の最大Y 座標よりも大きいならば to を返す。

もし y が描画領域に乗っていてかつ x が描画領域の最小X 座標よりも 小さい場合は、直線 $y = y$ 上に描画される最初の文字の文字位置を返す。

もし y が描画領域に乗っていてかつ x が描画領域の最大X 座標よりも 大きい場合は、直線 $y = y$ 上に描画される最後の文字の文字位置を返す。

2.22.3.7 mdraw_glyph_info()

```
int mdraw_glyph_info (
    MFrame * frame,
    MText * mt,
    int from,
    int pos,
    MDrawControl * control,
    MDrawGlyphInfo * info )
```

グリフに関する情報を計算する。

関数 `mdraw_glyph_info()` は、関数 `mdraw_text_with_control()` が描画制御オブジェクト control を用いて M-text mt の from から to までをフレーム frame に描画した場合、M-text の文字位置 pos の文字を覆うグリフに関する情報を計算する。

情報は info のメンバに保持される。

参照:

[MDrawGlyphInfo](#)

2.22.3.8 mdraw_glyph_list()

```
int mdraw_glyph_list (
    MFrame * frame,
    MText * mt,
    int from,
    int to,
    MDrawControl * control,
    MDrawGlyph * glyphs,
    int array_size,
    int * num_glyphs_return )
```

グリフ列に関する情報を計算する。

関数 `mdraw_glyph_list()` は、関数 `mdraw_text_with_control()` が描画制御オブジェクト control を用いて M-text mt の from から to までをフレーム frame に描画した場合の、各グリフの情報を glyphs が指す配列に格納する。array_size はその配列のサイズである。

もし array_size がすべてのグリフについての情報を格納するのに十分であれば、num_glyphs_return が指す場所に実際に埋めた要素の数を設定し 0 を返す。

そうでなければ、num_glyphs_return が指す場所に必要な配列のサイズを設定し、-1 を返す。

参照:

[MDrawGlyph](#)

2.22.3.9 mdraw_text_items()

```
void mdraw_text_items (
    MFrame * frame,
    MDrawWindow win,
    int x,
    int y,
    MDrawTextItem * items,
    int nitems )
```

textitem を表示する.

関数 [mdraw_text_items\(\)](#) は、一個以上のテキストアイテムを、フレーム `frame` のウィンドウ `win` の座標 `(x, y)` に表示する。items は表示すべきテキストアイテムの配列であり、nitems はその個数である。

参照:

[MTextItem](#), [mdraw_text\(\)](#).

2.22.3.10 mdraw_default_line_break()

```
int mdraw_default_line_break (
    MText * mt,
    int pos,
    int from,
    int to,
    int line,
    int y )
```

改行位置を計算する.

関数 [mdraw_default_line_break\(\)](#) は、行が最大幅中に収まらない場合の改行位置を、行番号 `line` と座標 `y` に基づいて計算する。`pos` は最大幅に収まる最後の文字の次の文字の位置である。`from` はその行の最初の文字の位置、`to` は最大幅が指定されていなければその行に表示される最後の文字の位置である。`line` と `y` は改行文字によって行が改まった際には 0 にリセットされ、最大幅によって行が改まった場合には 1 ずつ増やされる。

戻り値:

この関数は改行する文字位置を返す。

2.22.3.11 mdraw_per_char_extents()

```
void mdraw_per_char_extents (
    MFrame * frame,
    MText * mt,
    MDrawMetric * array_return,
    MDrawMetric * overall_return )
```

M-text の文字毎の表示範囲情報を得る。

関数 [mdraw_per_char_extents\(\)](#) は、M-text *mt* 中の各文字の表示範囲を計算する。この計算に用いるフォントは、*mt* のテキストプロパティで指定されたフェースと、フレーム *frame* のデフォルトフェースによって決まる。*array_return* の各要素は、*mt* 中の各文字の表示範囲情報によって順に埋められる。表示範囲情報とは、表示原点からの相対位置と各文字の占める長方形である。*array_return* の要素数は、M-text 中の文字数以上でなければならない。

ポインタ *overall_return* が NULL でない場合は、テキスト全体の表示範囲情報も計算し、その結果を *overall_return* の指す構造体に格納する。

2.22.3.12 mdraw_clear_cache()

```
void mdraw_clear_cache (
    MText * mt )
```

キャッシュ情報を消す。

関数 [mdraw_clear_cache\(\)](#) は描画関数によって M-text *mt* に付加されたキャッシュ情報をすべて消去する。MDrawControl の 'format' あるいは 'line_break' メンバ関数の振舞いが変わった場合にはキャッシュを消去しなくてはならない。

参照:

[MDrawControl](#)

2.22.4 変数詳解

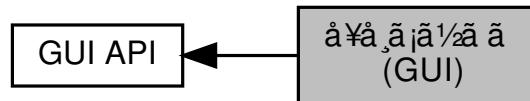
2.22.4.1 mdraw_line_break_option

```
int mdraw_line_break_option
```

2.23 入力メソッド (GUI)

ウィンドウシステム上の入力メソッドのサポート.

入力メソッド (GUI) 連携図



データ構造

- `struct MInputGUIArgIC`
関数 `minput_create_ic()` の引数の型宣言.
- `struct MInputXIMArgIM`
関数 `minput_open_im()` の引数 `arg` によって指される構造体.
- `struct MInputXIMArgIC`
関数 `minput_create_ic()` の引数 `arg` によって指される構造体.

関数

- `MSymbol minput_event_to_key (MFrame *frame, void *event)`
イベントを入力キーに変換する.

変数

- `MInputDriver minput_gui_driver`
ウィンドウシステムの内部入力メソッド用入力ドライバ.
- `MSymbol Mxim`
"xim"を名前として持つシンボル.

2.23.1 詳解

ウィンドウシステム上の入力メソッドのサポート.

入力ドライバ `minput_gui_driver` は、ウィンドウシステム上で用いられる内部入力メソッド用のドライバである。このドライバは入力スポットに `preedit` テキストと `status` テキストを表示する。詳細については `minput_gui_driver` の説明を参照のこと。

m17n-X ライブラリは、`Mxim` という名前を持つ外部入力メソッドを提供している。これは XIM (X Input Method) をバックグラウンドの入力エンジンとして利用する。シンボル `Mxim` は `Minput_driver` というプロパティを持っており、その値は入力ドライバ `minput_xim_driver` へのポインタである。詳細については `minput_xim_driver` の説明を参照のこと。

2.23.2 関数詳解

2.23.2.1 minput_event_to_key()

```
MSymbol minput_event_to_key (
    MFrame * frame,
    void * event )
```

イベントを入力キーに変換する。

関数 `minput_event_to_key()` は、`frame` のイベント `event` に対応する入力キーを返す。ここでの「対応」はウィンドウシステムに依存する。

m17n-X ライブラリの場合には、`event` は構造体 `XKeyEvent` へのポインタであり、次のように処理される。

まず、関数 `XKeysymToString` によって、`event` の `keysym` 名を取得し、次いで以下の変更を加える。

名前が "a" .. "z" のいずれかであって `event` に Shift モディファイアがあれば、名前はそれぞれ "A" .. "Z" に変換され、Shift モディファイアは取り除かれる。

名前が 1 バイト長で `event` に Control モディファイアがあれば、名前と 0x1F とをビット単位で `and` 演算し、Control モディファイアは取り除かれる。

それでも `event` にまだモディファイアがあれば、名前の前にそれぞれ "S-" (Shift), "C-" (Control), "M-" (Meta), "A-" (Alt), "G-" (AltGr), "s-" (Super), "H-" (Hyper) がこの順番で付く。

たとえば、`keysym` 名が "a" でイベントが Shift, Meta, and Hyper モディファイアを持てば、得られる名前は "M-H-A" である。

最後にその名前を持つシンボルを返す。

2.23.3 変数詳解

2.23.3.1 minput_gui_driver

```
MInputDriver minput_gui_driver
```

ウィンドウシステムの内部入力メソッド用入力ドライバ。

入力ドライバ `minput_gui_driver` は、ウィンドウシステム上で用いられる入力メソッド用ドライバである。

このドライバは、関数 `minput_set_spot()` によって設定された入力スポットに `preedit` テキスト用のサブウィンドウと `status` テキスト用のサブウィンドウを作り、それぞれを表示する。

マクロ `M17N_INIT()` は変数 `minput_driver` をこのドライバへのポインタに設定し、全ての内部入力メソッドがこのドライバを使うようにする。

したがって、`minput_driver` がデフォルト値のままであれば、`minput_` で始まる名前を持つ関数の引数のうちドライバ依存のものは以下ようになる。

関数 `minput_open_jm()` の引数 `arg` は無視される。

関数 `minput_create_ic()` の引数 `arg` は構造体 `MInputGUIArgIC` へのポインタでなくてはならない。詳細については `MInputGUIArgIC` の説明を参照のこと。

関数 `minput_filter()` の引数 `arg` が `Mnil` の場合、`arg` は `XEvent` 型のオブジェクトへのポインタでなくてはならない。この場合 `key` は `arg` から生成される。

関数 `minput_lookup()` の引数 `arg` は関数 `minput_filter()` の引数 `arg` と同じでなくてはならない。

2.23.3.2 Mxim

[MSymbol](#) Mxim

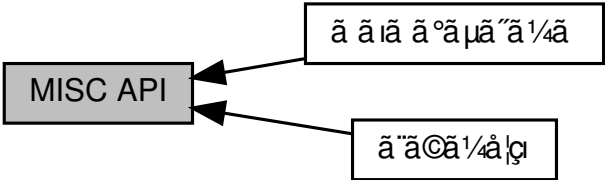
"xim"を名前として持つシンボル.

変数 Mxim は"xim"を名前として持つシンボルである。"xim" は入力メソッドドライバ [minput_xim_driver](#) の名前である。

2.24 MISC API

その他の API

MISC API 連携図



モジュール

- [エラー処理](#)
m17n ライブラリのエラー処理.
- [デバッグサポート](#)
m17n ライブラリユーザのためのプログラムデバッグサポート.

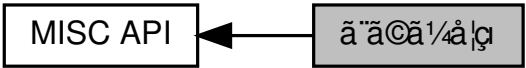
2.24.1 詳解

その他の API

2.25 エラー処理

m17n ライブラリのエラー処理.

エラー処理 連携図



列挙型

- enum `MErrorCode` {
 `MERROR_NONE` ,
 `MERROR_OBJECT` ,
 `MERROR_SYMBOL` ,
 `MERROR_MTEXT` ,
 `MERROR_TEXTPROP` ,
 `MERROR_CHAR` ,
 `MERROR_CHARTABLE` ,
 `MERROR_CHARSET` ,
 `MERROR_CODING` ,
 `MERROR_RANGE` ,
 `MERROR_LANGUAGE` ,
 `MERROR_LOCALE` ,
 `MERROR_PLIST` ,
 `MERROR_MISC` ,
 `MERROR_WIN` ,
 `MERROR_X` ,
 `MERROR_FRAME` ,
 `MERROR_FACE` ,
 `MERROR_DRAW` ,
 `MERROR_FLT` ,
 `MERROR_FONT` ,
 `MERROR_FONTSET` ,
 `MERROR_FONT_OTF` ,
 `MERROR_FONT_X` ,
 `MERROR_FONT_FT` ,
 `MERROR_JM` ,
 `MERROR_DB` ,
 `MERROR_IO` ,
 `MERROR_DEBUG` ,
 `MERROR_MEMORY` ,
 `MERROR_GD` ,
 `MERROR_MAX` }
 m17n ライブラリエラーコードの列挙.

変数

- int `merror_code`
 m17n ライブラリのエラーコードを保持する外部変数.
- void(* `m17n_memory_full_handler`)(enum `MErrorCode` err)
 メモリ割当てエラーハンドラ.

2.25.1 詳解

m17n ライブラリのエラー処理.

m17n ライブラリの関数では、2つの種類のエラーが起こり得る。

一つは引数のエラーである。ライブラリの関数が妥当でない引数とともに呼ばれた場合、その関数はエラーを意味する値を返し、同時に外部変数 `merror_code` にゼロでない整数をセットする。

もう一つの種類はメモリ割当てエラーである。システムが必要な量のメモリを割当てることができない場合、ライブラリ関数は外部変数 `m17n_memory_full_handler` が指す関数を呼ぶ。デフォルトでは、関数 `default_error_handle()` を指しており、この関数は単に `exit ()` を呼ぶ。

2.25.2 列挙型詳解

2.25.2.1 MErrorCode

enum MErrorCode

m17n ライブラリエラーコードの列挙.

m17n ライブラリエラーコードの列挙

ライブラリの関数が妥当でない引数とともに呼ばれた場合には、変数 `merror_code` をこれらの値のどれかにセットする。すべてのエラーコードは正の整数である。

メモリ割当てエラーの際には、外部変数 `m17n_memory_full_handler` の指す関数が、これらの値のうちのどれかを引数として呼ばれる。

列挙値

MERROR_NONE	
MERROR_OBJECT	
MERROR_SYMBOL	
MERROR_MTEXT	
MERROR_TEXTPROP	
MERROR_CHAR	
MERROR_CHARTABLE	
MERROR_CHARSET	
MERROR_CODING	
MERROR_RANGE	
MERROR_LANGUAGE	
MERROR_LOCALE	
MERROR_PLIST	
MERROR_MISC	
MERROR_WIN	
MERROR_X	
MERROR_FRAME	
MERROR_FACE	
MERROR_DRAW	
MERROR_FLT	
MERROR_FONT	
MERROR_FONTSET	
MERROR_FONT_OTF	
MERROR_FONT_X	
MERROR_FONT_FT	
MERROR_IM	
MERROR_DB	
MERROR_IO	
MERROR_DEBUG	
MERROR_MEMORY	
MERROR_GD	
MERROR_MAX	

2.25.3 変数詳解

2.25.3.1 merror_code

```
int merror_code
```

m17n ライブラリのエラーコードを保持する外部変数.

外部変数 `merror_code` は、m17n ライブラリのエラーコードを保持する。ライブラリ関数が妥当でない引数とともに呼ばれた際には、この変数を enum `MErrorCode` の一つにセットする。

この変数の初期値は 0 である。

2.25.3.2 m17n_memory_full_handler

```
void(* m17n_memory_full_handler) (enum MErrorCode err) (
    enum MErrorCode err )
```

メモリ割当てエラーハンドラ.

変数 `m17n_memory_full_handler` は、ライブラリ関数がメモリ割当てに失敗した際に呼ぶべき関数へのポインタである。`err` は enum `MErrorCode` のうちのいずれかであり、どのライブラリ関数でエラーが起きたかを示す。

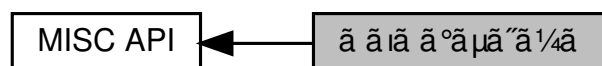
初期設定では、この変数は単に `exit()` を `err` を引数として呼ぶ関数を指している。

これとは異なるエラー処理を必要とするアプリケーションは、この変数を適当な関数に設定することで、目的を達成できる。

2.26 デバッグサポート

m17n ライブラリユーザのためのプログラムデバッグサポート.

デバッグサポート 連携図



関数

- `MFace * mdebug_dump_face (MFace *face, int indent)`
フェースをダンプする。
- `MInputMethod * mdebug_dump_im (MInputMethod *im, int indent)`
入力メソッドをダンプする。
- `int mdebug_hook ()`
エラーの際に呼ばれるフック関数。
- `MText * mdebug_dump_mtext (MText *mt, int indent, int fullp)`
M-text をダンプする。
- `MSymbol mdebug_dump_symbol (MSymbol symbol, int indent)`
シンボルをダンプする。
- `MSymbol mdebug_dump_all_symbols (int indent)`
すべてのシンボル名をダンプする。

2.26.1 詳解

m17n ライブラリユーザのためのプログラムデバッグサポート。

m17n ライブラリは、そのユーザが自分のプログラムをデバッグするために、以下の機能をサポートしている。

- さまざまな情報の標準エラー出力へのプリントを制御する環境変数。
 - `MDEBUG_INIT - 1` ならば、`M17N_INIT()` が呼ばれた時点で、ライブラリの初期化に関する情報をプリントする。
 - `MDEBUG_FINI - 1` ならば、`M17N_FINI()` が呼ばれた時点で、まだ解放されていないオブジェクトの参照数をプリントする。
 - `MDEBUG_CHARSET - 1` ならば、m17n データベースからロードされた文字セットについての情報をプリントする。
 - `MDEBUG_CODING - 1` ならば、m17n データベースからロードされたコード系についての情報をプリントする。
 - `MDEBUG_DATABASE - 1` ならば、m17n データベースからロードされたデータについての情報をプリントする。
 - `MDEBUG_FONT - 1` ならば、選択されてオープンされたフォントについての情報をプリントする。
 - `MDEBUG_FLT - 1, 2, もしくは 3` ならば、Font Layout Table のどのコマンドが実行中かについての情報をプリントする。より大きな値程より詳しい情報をプリントする。
 - `MDEBUG_INPUT - 1` ならば、実行中の入力メソッドの状態に付いての情報をプリントする。
 - `MDEBUG_ALL - 1` ならば、上記すべての変数を 1 にしたのと同じ効果を持つ。
 - `MDEBUG_OUTPUT_FILE` – もしファイル名なら、上記デバッグ情報はそのファイルに追加される。もし "stdout" ならその情報は標準出力に出力される。
- 種々のオブジェクトを人間に可読な形でプリントする関数。詳細は関数 `mdebug_dump_XXXX()` の説明参照。
- エラー発生時に呼ばれるフック関数。 `mdebug_hook()` の説明参照。

2.26.2 関数詳解

2.26.2.1 mdebug_dump_face()

```
MFace* mdebug_dump_face (  
    MFace * face,  
    int indent )
```

フェースをダンプする.

関数 `mdebug_dump_face()` はフェース `face` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。 `indent` は2行目以降のインデントを指定する。

戻り値:

この関数は `face` を返す。

2.26.2.2 mdebug_dump_im()

```
MInputMethod* mdebug_dump_im (  
    MInputMethod * im,  
    int indent )
```

入力メソッドをダンプする.

関数 `mdebug_dump_im()` は入力メソッド `im` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で出力する。 `indent` は2行目以降のインデントを指定する。

戻り値:

この関数は `im` を返す。

2.26.2.3 mdebug_hook()

```
int mdebug_hook (  
    void )
```

エラーの際に呼ばれるフック関数.

関数 `mdebug_hook()` はエラーが起こった際に呼ばれ、何もせずに-1を返す。デバッガ内でブレークポイントを設定するために用いることができる。

2.26.2.4 mdebug_dump_mtext()

```
MText* mdebug_dump_mtext (
    MText * mt,
    int indent,
    int fullp )
```

M-text をダンプする。

関数 `mdebug_dump_mtext()` は M-text `mt` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。 `indent` は 2 行目以降のインデントを指定する。 `fullp` が 0 ならば、文字コード列だけを印刷する。そうでなければ、内部バイト列とテキストプロパティも印刷する。

戻り値:

この関数は `mt` を返す。

2.26.2.5 mdebug_dump_symbol()

```
MSymbol mdebug_dump_symbol (
    MSymbol symbol,
    int indent )
```

シンボルをダンプする。

関数 `mdebug_dump_symbol()` はシンボル `$symbol` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。 `indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `symbol` を返す。

エラー:

`MERROR_DEBUG`

2.26.2.6 mdebug_dump_all_symbols()

```
MSymbol mdebug_dump_all_symbols (
    int indent )
```

すべてのシンボル名をダンプする。

関数 `mdebug_dump_all_symbols()` は、すべてのシンボルの名前を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに印刷する。 `indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `Mnil` を返す。

エラー:

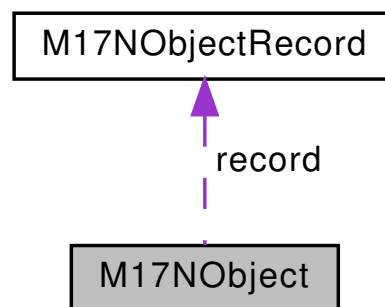
`MERROR_DEBUG`

Chapter 3

データ構造詳解

3.1 M17NObject 構造体

M17NObject 連携図



フィールド

- unsigned `ref_count`: 16
 - unsigned `ref_count_extended`: 1
 - unsigned `flag`: 15
 - union {
 - void(* `freer`)(void *)
 - `M17NObjectRecord` * `record`
- } `u`

3.1.1 フィールド詳解

構築: Doxygen

3.1.1.1 ref_count

```
unsigned M17NObject::ref_count
```

3.1.1.2 ref_count_extended

```
unsigned M17NObject::ref_count_extended
```

3.1.1.3 flag

```
unsigned M17NObject::flag
```

3.1.1.4 freer

```
void(* M17NObject::freer) (void *)
```

3.1.1.5 record

```
M17NObjectRecord* M17NObject::record
```

3.1.1.6

```
union { ... } M17NObject::u
```

3.2 M17NObjectArray 構造体

フィールド

- char * [name](#)
- int [count](#)
- int [size](#)
- int [inc](#)
- int [used](#)
- void ** [objects](#)
- M17NObjectArray * [next](#)

3.2.1 フィールド詳解

3.2.1.1 name

```
char* M17NObjectArray::name
```

3.2.1.2 count

```
int M17NObjectArray::count
```

3.2.1.3 size

```
int M17NObjectArray::size
```

3.2.1.4 inc

```
int M17NObjectArray::inc
```

3.2.1.5 used

```
int M17NObjectArray::used
```

3.2.1.6 objects

```
void** M17NObjectArray::objects
```

3.2.1.7 next

```
M17NObjectArray* M17NObjectArray::next
```

3.3 M17NObjectHead 構造体

管理下オブジェクトの最初のメンバ.

フィールド

- void * [filler](#) [2]

3.3.1 詳解

管理下オブジェクトの最初のメンバ.

アプリケーションプログラムが新しい構造体を管理下オブジェクトとして定義する際には、最初のメンバは [M17NObjectHead](#) 構造体型でなくてはならない。 [M17NObjectHead](#) の内容は m17n ライブラリが使用するので、アプリケーションプログラムは触れてはならない。

3.3.2 フィールド詳解

3.3.2.1 filler

```
void* M17NObjectHead::filler[2]
```

Hidden from applications.

3.4 M17NObjectRecord 構造体

フィールド

- void(* [freer](#))(void *)
- int [size](#)
- int [inc](#)
- int [used](#)
- unsigned * [counts](#)

3.4.1 フィールド詳解

3.4.1.1 freer

```
void(* M17NObjectRecord::freer) (void *)
```

3.4.1.2 size

```
int M17NObjectRecord::size
```

3.4.1.3 inc

```
int M17NObjectRecord::inc
```

3.4.1.4 used

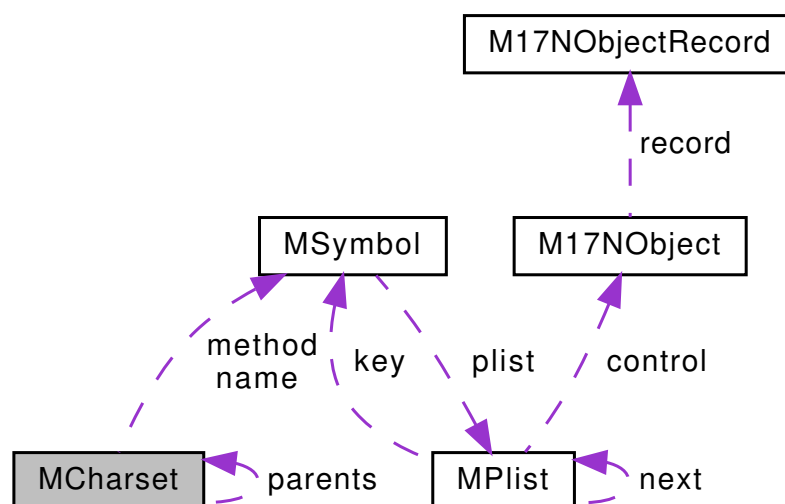
```
int M17NObjectRecord::used
```

3.4.1.5 counts

```
unsigned* M17NObjectRecord::counts
```

3.5 MCharset 構造体

MCharset 連携図



フィールド

- unsigned `ref_count`
- `MSymbol` `name`
- int `dimension`
- int `code_range` [16]
- int `code_range_min_code`
- int `no_code_gap`
- unsigned char `code_range_mask` [256]
- unsigned `min_code`
- unsigned `max_code`
- int `ascii_compatible`
- int `min_char`
- int `max_char`
- int `final_byte`
- int `revision`
- `MSymbol` `method`
- int * `decoder`
- `MCharTable` * `encoder`
- int `unified_max`
- `MCharset` * `parents` [8]
- int `nparents`
- unsigned `subset_min_code`
- unsigned `subset_max_code`
- int `subset_offset`
- int `simple`
- int `fully_loaded`

3.5.1 フィールド詳解

3.5.1.1 `ref_count`

```
unsigned MCharset::ref_count
```

3.5.1.2 `name`

```
MSymbol MCharset::name
```

3.5.1.3 `dimension`

```
int MCharset::dimension
```

3.5.1.4 code_range

```
int MCharset::code_range[16]
```

3.5.1.5 code_range_min_code

```
int MCharset::code_range_min_code
```

3.5.1.6 no_code_gap

```
int MCharset::no_code_gap
```

3.5.1.7 code_range_mask

```
unsigned char MCharset::code_range_mask[256]
```

3.5.1.8 min_code

```
unsigned MCharset::min_code
```

3.5.1.9 max_code

```
unsigned MCharset::max_code
```

3.5.1.10 ascii_compatible

```
int MCharset::ascii_compatible
```

3.5.1.11 min_char

```
int MCharset::min_char
```

3.5.1.12 max_char

```
int MCharset::max_char
```

3.5.1.13 final_byte

```
int MCharset::final_byte
```

3.5.1.14 revision

```
int MCharset::revision
```

3.5.1.15 method

```
MSymbol MCharset::method
```

3.5.1.16 decoder

```
int* MCharset::decoder
```

3.5.1.17 encoder

```
MCharTable\* MCharset::encoder
```


3.5.1.18 unified_max

```
int MCharset::unified_max
```

3.5.1.19 parents

```
MCharset* MCharset::parents[8]
```

3.5.1.20 nparents

```
int MCharset::nparents
```

3.5.1.21 subset_min_code

```
unsigned MCharset::subset_min_code
```

3.5.1.22 subset_max_code

```
unsigned MCharset::subset_max_code
```

3.5.1.23 subset_offset

```
int MCharset::subset_offset
```

3.5.1.24 simple

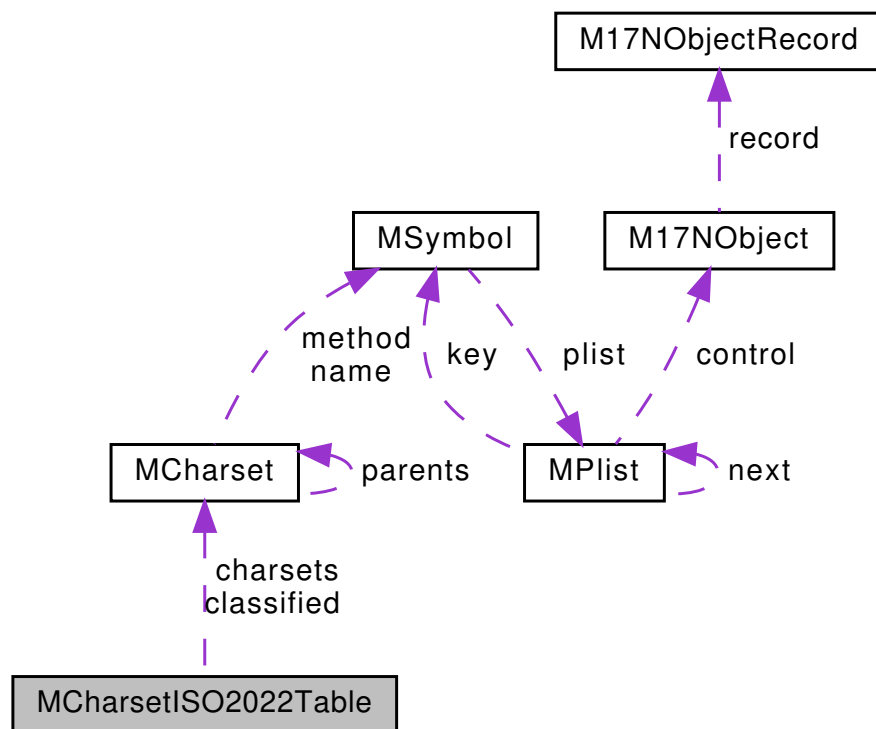
```
int MCharset::simple
```

3.5.1.25 fully_loaded

```
int MCharset::fully_loaded
```

3.6 MCharsetISO2022Table 構造体

MCharsetISO2022Table 連携図



フィールド

- int [size](#)
- int [inc](#)
- int [used](#)
- [MCharset](#) ** [charsets](#)
- [MCharset](#) * [classified](#) [[ISO_MAX_DIMENSION](#)][[ISO_MAX_CHARS](#)][[ISO_MAX_FINAL](#)]

3.6.1 フィールド詳解

3.6.1.1 size

```
int MCharsetISO2022Table::size
```

3.6.1.2 inc

```
int MCharsetISO2022Table::inc
```

3.6.1.3 used

```
int MCharsetISO2022Table::used
```

3.6.1.4 charsets

```
MCharset** MCharsetISO2022Table::charsets
```

3.6.1.5 classified

```
MCharset* MCharsetISO2022Table::classified[ISO_MAX_DIMENSION][ISO_MAX_CHARS][ISO_MAX_FINAL]
```

3.7 MCodingInfoISO2022 構造体

[MCODING_TYPE_ISO_2022](#) タイプのコード系に必要な付加情報用構造体.

フィールド

- int [initial_invocation](#) [2]
- char [designations](#) [32]
- unsigned [flags](#)

3.7.1 詳解

[MCODING_TYPE_ISO_2022](#) タイプのコード系に必要な付加情報用構造体.

[MCODING_TYPE_ISO_2022](#) タイプのコード系に必要な付加情報用を保持するための構造体.

3.7.2 フィールド詳解

3.7.2.1 initial_invocation

```
int MCodingInfoISO2022::initial_invocation[2]
```

各図形文字領域 (Graphic Left と Graphic Right) に呼び出されている、ISO2022 符合拡張要素の番号のテーブル。-1 はその領域にどの符合拡張要素も呼び出されていないことを示す。

3.7.2.2 designations

```
char MCodingInfoISO2022::designations[32]
```

符合拡張要素のテーブル。N 番目の要素は、charset_names の N 番目の文字セットに対応する。charset_names は関数 [mconv_define_coding\(\)](#) の引数となる。

値が 0..3 だったら、対応する文字セットを G0..G3 のそれぞれに指示すること、また初期状態ですでに G0..G3 に指示されていることを意味する。

値が -4..-1 だったら、対応する文字セットを G0..G3 のそれぞれに指示すること、しかし初期状態ではどこにも指示されていないことを意味する。

3.7.2.3 flags

```
unsigned MCodingInfoISO2022::flags
```

enum MCodingFlagISO2022 のビット単位での論理 OR

3.8 MCodingInfoUTF 構造体

[MCODING_TYPE_UTF](#) タイプのコード系に必要な付加情報用の構造体.

フィールド

- int [code_unit_bits](#)
- int [bom](#)
- int [endian](#)

3.8.1 詳解

[MCODING_TYPE_UTF](#) タイプのコード系に必要な付加情報用の構造体.

3.8.2 フィールド詳解

3.8.2.1 code_unit_bits

```
int MCodingInfoUTF::code_unit_bits
```

コード長（ビット数）の指定。値は 8, 16, 32 のいずれか。

3.8.2.2 bom

```
int MCodingInfoUTF::bom
```

先頭の BOM (バイトオーダーマーク) の取り扱いを指定する。値は 0, 1, 2 のいずれかであり、それぞれの意味は以下のようになる。

0: デコードの際に最初の 2 バイトを調べる。もしそれが BOM であれば、エンディアンをそれで判定する。そうでなければ、メンバ `endian` に従ってエンディアンを決定する。エンコードの際には `endian` に従ったバイト列を先頭に BOM 付で生成する。

1: デコードの際、最初の 2 バイトを BOM として扱わず、エンディアンは `endian` で判定する。エンコードの際には、BOM を出力せず、`endian` に応じたバイト列を生成する。

2: デコードの際に最初の 2 バイトを BOM として扱い、それによってエンディアンを判定する。エンコードの際には `endian` に応じたバイト列を先頭に BOM 付きで生成する。

3.8.2.3 endian

```
int MCodingInfoUTF::endian
```

エンディアンのタイプを指定する。値は 0 か 1 であり、0 ならばリトルエンディアン、1 ならばビッグエンディアンである。

<code_unit_bits> が 8 の場合には、この値は意味を持たない。

3.9 MConverter 構造体

コード変換に用いられる構造体。

フィールド

- int `lenient`
- int `last_block`
- unsigned `at_most`
- int `nchars`
- int `nbytes`
- enum `MConversionResult result`
- union {
 - void * `ptr`
 - double `dbl`
 - char `c` [256]
 } `status`
- void * `internal_info`

3.9.1 詳解

コード変換に用いられる構造体。

コード変換に用いられる構造体。最初の3つのメンバは変換を制御する。

3.9.2 フィールド詳解

3.9.2.1 lenient

```
int MConverter::lenient
```

厳密な変換が必要でない場合に値を0以外にする。デフォルトでは、変換は厳密である。

変換が厳密とは、デコードの際には最初の不正なバイトでコンバータが止まること、エンコードの際には変換されるコード系でサポートされない最初の文字でコンバータが止まることを指す。これらの場合、MConverter->result はそれぞれ MCONVERSION_RESULT_INVALID_BYTE か MCONVERSION_RESULT_INVALID_CHAR となる。

変換が厳密でない場合には、デコードの際の不正なバイトはそのバイトのまま残る。またエンコードの際には、不正な文字が Unicode 文字の場合には "<U+XXXX>" に、そうでない場合には "<M+XXXXXX>" に置き換えられる。

3.9.2.2 last_block

```
int MConverter::last_block
```

バイト列の終端のブロックをデコードする際、または文字列の終端のブロックをエンコードする際は、値を0以外にする。この値は以下のように変換に影響する。

デコーディングの際に最後の数バイトが正しいバイトシーケンスを形成するには短すぎる場合：

値が0でなければ、変換はそのシーケンスの最初のバイトにおいて、エラー (MCONVERSION_RESULT_INVALID_BYTE) で終る。

値が0ならば、変換は成功して終る。問題の数バイトはキャリーオーバーとしてコンバータに保持され、変換の続きを行う際に変換するバイト列の前に付けられる。

エンコーディングの際にコード系が文脈依存の場合、

値が0でなければ、コンテキストを最初に戻すためのバイト列がソースの文字とかかわりなく変換の結果生成されることがある。

値が0ならば、そのようなバイト列は生成されない。

3.9.2.3 at_most

```
unsigned MConverter::at_most
```

0 でなければ、変換される最大の文字数を指定する。

3.9.2.4 nchars

```
int MConverter::nchars
```

以下の 3 つのメンバは変換の結果を表すためのものである。

最近にデコード/エンコードされた文字数。

3.9.2.5 nbytes

```
int MConverter::nbytes
```

最近にデコード/エンコードされたバイト数。

3.9.2.6 result

```
enum MConversionResult MConverter::result
```

コード変換の結果を示すコード。

3.9.2.7 ptr

```
void* MConverter::ptr
```

3.9.2.8 dbl

```
double MConverter::dbl
```

3.9.2.9 c

```
char MConverter::c[256]
```

3.9.2.10

```
union { ... } MConverter::status
```

コード変換の状況に関する種々の情報。内容はコード系のタイプによって異なる。status はどのような型へのキャストに対しても安全なようにメモリアラインされており、また最低 256 バイトのメモリ領域が使えるようになっている。

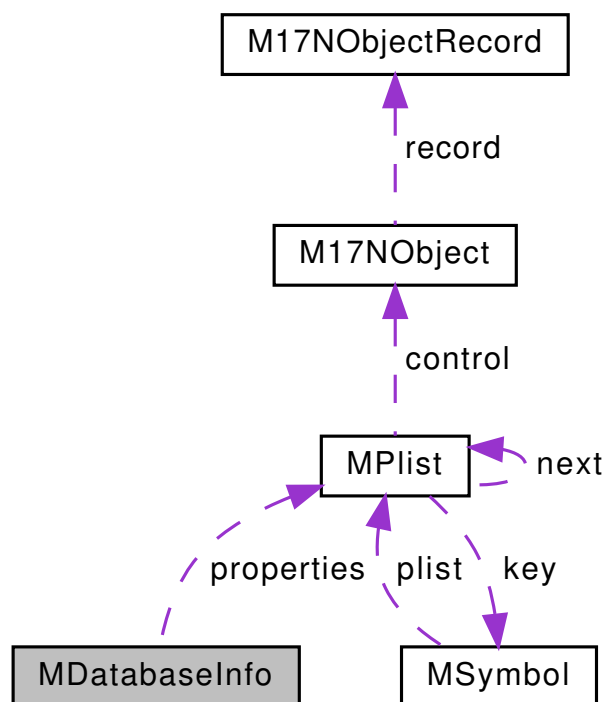
3.9.2.11 internal_info

```
void* MConverter::internal_info
```

このメンバは内部的に使用され、アプリケーションプログラムは触れてはならない。

3.10 MDatabaseInfo 構造体

MDatabaseInfo 連携図



フィールド

- char * filename
- int len
- char * absolute_filename
- enum MDatabaseStatus status
- time_t time
- char * lock_file
- char * uniq_file
- MPlist * properties

3.10.1 フィールド詳解

3.10.1.1 filename

```
char* MDatabaseInfo::filename
```

3.10.1.2 len

```
int MDatabaseInfo::len
```

3.10.1.3 absolute_filename

```
char* MDatabaseInfo::absolute_filename
```

3.10.1.4 status

```
enum MDatabaseStatus MDatabaseInfo::status
```

3.10.1.5 time

```
time_t MDatabaseInfo::time
```

3.10.1.6 lock_file

```
char* MDatabaseInfo::lock_file
```

3.10.1.7 uniq_file

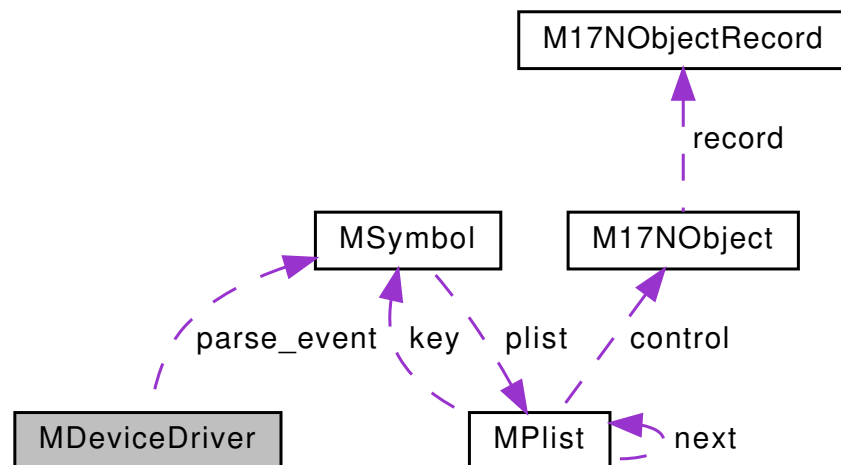
```
char * MDatabaseInfo::uniq_file
```

3.10.1.8 properties

```
MPlist* MDatabaseInfo::properties
```

3.11 MDeviceDriver 構造体

MDeviceDriver 連携図



フィールド

- void(* close)(MFrame *frame)
- void (* get_prop)(MFrame *frame, MSymbol key)
- void(* realize_face)(MRealizedFace *rface)
- void(* free_realized_face)(MRealizedFace *rface)
- void(* fill_space)(MFrame *frame, MDrawWindow win, MRealizedFace *rface, int reverse, int x, int y, int width, int height, MDrawRegion region)
- void(* draw_empty_boxes)(MDrawWindow win, int x, int y, MGlyphString *gstring, MGlyph *from, MGlyph *to, int reverse, MDrawRegion region)
- void(* draw_hline)(MFrame *frame, MDrawWindow win, MGlyphString *gstring, MRealizedFace *rface, int reverse, int x, int y, int width, MDrawRegion region)
- void(* draw_box)(MFrame *frame, MDrawWindow win, MGlyphString *gstring, MGlyph *g, int x, int y, int width, MDrawRegion region)
- void(* draw_points)(MFrame *frame, MDrawWindow win, MRealizedFace *rface, int intensity, MDrawPoint *points, int num, MDrawRegion region)
- MDrawRegion(* region_from_rect)(MDrawMetric *rect)

- void(* union_rect_with_region)(MDrawRegion region, MDrawMetric *rect)
- void(* intersect_region)(MDrawRegion region1, MDrawRegion region2)
- void(* region_add_rect)(MDrawRegion region, MDrawMetric *rect)
- void(* region_to_rect)(MDrawRegion region, MDrawMetric *rect)
- void(* free_region)(MDrawRegion region)
- void(* dump_region)(MDrawRegion region)
- MDrawWindow(* create_window)(MFrame *frame, MDrawWindow parent)
- void(* destroy_window)(MFrame *frame, MDrawWindow win)
- void(* map_window)(MFrame *frame, MDrawWindow win)
- void(* unmap_window)(MFrame *frame, MDrawWindow win)
- void(* window_geometry)(MFrame *frame, MDrawWindow win, MDrawWindow parent, MDrawMetric *geometry)
- void(* adjust_window)(MFrame *frame, MDrawWindow win, MDrawMetric *current, MDrawMetric *new)
- MSymbol(* parse_event)(MFrame *frame, void *arg, int *modifiers)

3.11.1 フィールド詳解

3.11.1.1 close

```
void(* MDeviceDriver::close) (MFrame *frame)
```

3.11.1.2 get_prop

```
void(* MDeviceDriver::get_prop) (MFrame *frame, MSymbol key)
```

3.11.1.3 realize_face

```
void(* MDeviceDriver::realize_face) (MRealizedFace *rface)
```

3.11.1.4 free_realized_face

```
void(* MDeviceDriver::free_realized_face) (MRealizedFace *rface)
```

3.11.1.5 fill_space

```
void(* MDeviceDriver::fill_space) (MFrame *frame, MDrawWindow win, MRealizedFace *rface, int reverse, int x, int y, int width, int height, MDrawRegion region)
```

3.11.1.6 draw_empty_boxes

```
void(* MDeviceDriver::draw_empty_boxes) (MDrawWindow win, int x, int y, MGlyphString *gstring, MGlyph *from, MGlyph *to, int reverse, MDrawRegion region)
```

3.11.1.7 draw_hline

```
void(* MDeviceDriver::draw_hline) (MFrame *frame, MDrawWindow win, MGlyphString *gstring, MRealizedFace *rface, int reverse, int x, int y, int width, MDrawRegion region)
```

3.11.1.8 draw_box

```
void(* MDeviceDriver::draw_box) (MFrame *frame, MDrawWindow win, MGlyphString *gstring, MGlyph *g, int x, int y, int width, MDrawRegion region)
```

3.11.1.9 draw_points

```
void(* MDeviceDriver::draw_points) (MFrame *frame, MDrawWindow win, MRealizedFace *rface, int intensity, MDrawPoint *points, int num, MDrawRegion region)
```

3.11.1.10 region_from_rect

```
MDrawRegion(* MDeviceDriver::region_from_rect) (MDrawMetric *rect)
```

3.11.1.11 union_rect_with_region

```
void(* MDeviceDriver::union_rect_with_region) (MDrawRegion region, MDrawMetric *rect)
```

3.11.1.12 intersect_region

```
void(* MDeviceDriver::intersect_region) (MDrawRegion region1, MDrawRegion region2)
```

3.11.1.13 region_add_rect

```
void(* MDeviceDriver::region_add_rect) (MDrawRegion region, MDrawMetric *rect)
```

3.11.1.14 region_to_rect

```
void(* MDeviceDriver::region_to_rect) (MDrawRegion region, MDrawMetric *rect)
```

3.11.1.15 free_region

```
void(* MDeviceDriver::free_region) (MDrawRegion region)
```

3.11.1.16 dump_region

```
void(* MDeviceDriver::dump_region) (MDrawRegion region)
```

3.11.1.17 create_window

```
MDrawWindow(* MDeviceDriver::create_window) (MFrame *frame, MDrawWindow parent)
```

3.11.1.18 destroy_window

```
void(* MDeviceDriver::destroy_window) (MFrame *frame, MDrawWindow win)
```

3.11.1.19 map_window

```
void(* MDeviceDriver::map_window) (MFrame *frame, MDrawWindow win)
```

3.11.1.20 unmap_window

```
void(* MDeviceDriver::unmap_window) (MFrame *frame, MDrawWindow win)
```

3.11.1.21 window_geometry

```
void(* MDeviceDriver::window_geometry) (MFrame *frame, MDrawWindow win, MDrawWindow parent,  
MDrawMetric *geometry)
```

3.11.1.22 adjust_window

```
void(* MDeviceDriver::adjust_window) (MFrame *frame, MDrawWindow win, MDrawMetric *current,  
MDrawMetric *new)
```

3.11.1.23 parse_event

```
MSymbol(* MDeviceDriver::parse_event) (MFrame *frame, void *arg, int *modifiers)
```

3.12 MDrawControl 構造体

テキスト表示制御の型宣言.

フィールド

- unsigned `as_image`: 1
- unsigned `align_head`: 1
- unsigned `two_dimensional`: 1
- unsigned `orientation_reversed`: 1
- unsigned `enable_bidi`: 1
- unsigned `ignore_formatting_char`: 1
- unsigned `fixed_width`: 1
- unsigned `anti_alias`: 1
- unsigned `disable_overlapping_adjustment`: 1
- unsigned int `min_line_ascent`
- unsigned int `min_line_descent`
- unsigned int `max_line_ascent`
- unsigned int `max_line_descent`
- unsigned int `max_line_width`
- unsigned int `tab_width`
- void(* `format`)(int line, int y, int *indent, int *width)
- int(* `line_break`)(MText *mt, int pos, int from, int to, int line, int y)
- int `with_cursor`
- int `cursor_pos`
- int `cursor_width`
- int `cursor_bidi`
- int `partial_update`
- int `disable_caching`
- MDrawRegion `clip_region`

3.12.1 詳解

テキスト表示制御の型宣言.

`MDrawControl` 型は、M-text をどう表示するかを制御する構造体である。

3.12.2 フィールド詳解

3.12.2.1 `as_image`

```
unsigned MDrawControl::as_image
```

0 でなければ、M-text を画像として、すなわち背景を M-text のフェースで指定されている背景色で埋めて表示する。そうでなければ背景は変わらない。

3.12.2.2 align_head

```
unsigned MDrawControl::align_head
```

0 でなく、各行の最初のグリフの `lbearing` が負ならば、グリフを水平に右にずらして、指定した位置より左にピクセルが描かれないようにする。

3.12.2.3 two_dimensional

```
unsigned MDrawControl::two_dimensional
```

0 でなければ、M-text を 2 次元的に、すなわち M-text 中の `newline` で改行し、続く文字は次の行に表示する。もし `<format>` が NULL でなく、その関数が 0 でない行幅を返せば、その幅より長い行も改行される。

3.12.2.4 orientation_reversed

```
unsigned MDrawControl::orientation_reversed
```

0 でなければ、M-text を指定した位置の右に表示する。

3.12.2.5 enable_bidi

```
unsigned MDrawControl::enable_bidi
```

0 なければ、bidi テキスト用にグリフを正しく整列する。

3.12.2.6 ignore_formatting_char

```
unsigned MDrawControl::ignore_formatting_char
```

0 でなければ、ユニコードに置ける一般カテゴリが Cf (Other, format) である文字を表示しない。

3.12.2.7 fixed_width

```
unsigned MDrawControl::fixed_width
```

0 でなければ、端末用のグリフを表示する。未実装。

3.12.2.8 anti_alias

```
unsigned MDrawControl::anti_alias
```

0 でなければ、アンチエイリアスでグリフを表示する。（バックエンドのフォントドライバがアンチエイリアス機能を持つ場合のみ。）

3.12.2.9 disable_overlapping_adjustment

unsigned MDrawControl::disable_overlapping_adjustment

0 でなければ、フォント境界での水平方向のグリフの重なりを避けるためのグリフ位置の調整を無効にする。

3.12.2.10 min_line_ascent

unsigned int MDrawControl::min_line_ascent

0 でなければ、値は行の ascent の最小値を示す。

3.12.2.11 min_line_descent

unsigned int MDrawControl::min_line_descent

0 でなければ、値は行の descent の最小値を示す。

3.12.2.12 max_line_ascent

unsigned int MDrawControl::max_line_ascent

0 でなければ、値は行の ascent の最大値を示す。

3.12.2.13 max_line_descent

unsigned int MDrawControl::max_line_descent

0 でなければ、値は行の descent の最大値を示す。

3.12.2.14 max_line_width

unsigned int MDrawControl::max_line_width

0 でなければ、値はこのディスプレイ上で各行が占めることのできるピクセル数を示す。0 は限定されないことを意味する。<format> が NULL でなければ無視される。

3.12.2.15 tab_width

unsigned int MDrawControl::tab_width

0 でなければ、値はタブストップ間の距離をコラム単位（コラムはフレームのデフォルトフォントにおける空白文字の幅である）で示す。0 は 8 を意味する。

3.12.2.16 format

```
void(* MDrawControl::format) (int line, int y, int *indent, int *width)
```

0 でなければ、値は関数であり、その関数は行番号 **LINE** と座標 **Y** に基づいて各行のインデントと最大幅を計算し、それぞれを **INDENT** と **WIDTH** で指される場所に保存する。

インデントは、各行の最初のグリフを右（メンバ `<orientation_reversed>` が 0 の時）あるいは左（それ以外の時）に何ピクセルずらすかを指定する。値が負ならば逆方向にずらす。

最大幅は、各行がディスプレイ上で占めることのできるピクセル数の最大値である。値が 0 の場合は制限を受けないことを意味する。

LINE と **Y** は改行文字によって行が改まった際には 0 にリセットされ、長い行が最大幅の制限によって改行されるたびに 1 増やされる。

これは `<two_dimensional>` が 0 でない場合にのみ有効である。

3.12.2.17 line_break

```
int(* MDrawControl::line_break) (MText *mt, int pos, int from, int to, int line, int y)
```

NULL でなければ、値は行が最大幅中に収まらない場合に行を改める位置を計算する関数である。POS は最大幅に収まる最後の文字の次の文字の位置である。FROM は行の最初の文字の位置、TO は最大幅が指定されていなければその行に表示される最後の文字の位置である。LINE と Y は `<format>` の引数と同様である。

この関数は行を改める文字位置を返さなくてはならない。また MT を変更してはならない。

関数 `mdraw_default_line_break()` は、空白を語の区切りとして用いるスクリプト用として有用である。

3.12.2.18 with_cursor

```
int MDrawControl::with_cursor
```

ゼロでなければ `<cursor_width>` にしたがってカーソルを表示する。

3.12.2.19 cursor_pos

```
int MDrawControl::cursor_pos
```

カーソルを表示する文字位置を示す。最大の文字位置より大きければ、カーソルは M-text の最後の文字の隣に表示される。負ならば、`<cursor_width>` が 0 でなくてもカーソルは表示されない。

3.12.2.20 cursor_width

```
int MDrawControl::cursor_width
```

0 でなければ、`<cursor_pos>` にカーソルを表示する。値が正ならば、カーソルの幅はその値（ピクセル単位）である。負ならば、カーソルのあるグリフと同じ幅である。

3.12.2.21 cursor_bidi

```
int MDrawControl::cursor_bidi
```

If 0 でなく、かつ <cursor_width> も 0 でなければ、バーカーソルを文字位置 <cursor_pos> と論理的にそれの前にある文字の 2 ヶ所に表示する。双方とも 1 ピクセル幅で、上か下に水平の飾りがつく。

3.12.2.22 partial_update

```
int MDrawControl::partial_update
```

0 でなければ、テキストの一部分を表示する際に、前後のテキストのうちその表示領域に侵入する部分も表示する。たとえば、タイ語テキスト 子音-母音-子音 というシークエンスのいくつかは、母音が二つの子音の間に上にのるように描かれる。このようなシークエンスがすでに描かれており、最後の子音だけを描き直す場合（たとえば、カーソル位置を更新する際など）このメンバが 0 であれば、母音の右半分が消されてしまう。これを 0 以外にすることによって、そのような際にも 子音-母音-子音 のシークエンスを正しく表示し続けることができる。

3.12.2.23 disable_caching

```
int MDrawControl::disable_caching
```

0 でなければ、M-text の表示に関する情報をキャッシュしない。

3.12.2.24 clip_region

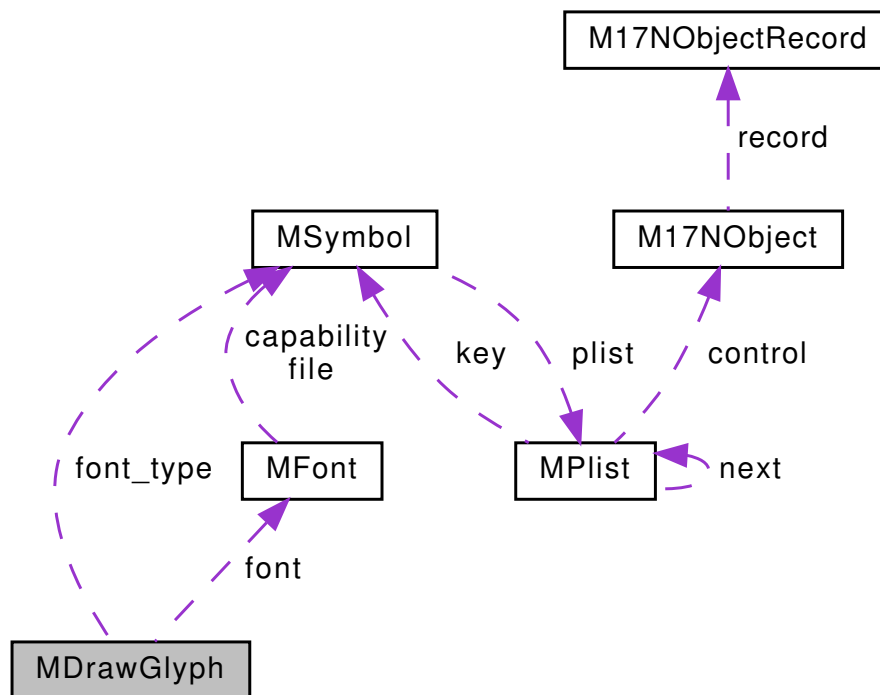
```
MDrawRegion MDrawControl::clip_region
```

NULL でなければ表示エリアを指定された領域に限定する。

3.13 MDrawGlyph 構造体

グリフの寸法とフォントに関する情報の型宣言。

MDrawGlyph 連携図



フィールド

- int [from](#)
- int [to](#)
- int [glyph_code](#)
- int [x_advance](#)
- int [y_advance](#)
- int [x_off](#)
- int [y_off](#)
- int [lbearing](#)
- int [rbearing](#)
- int [ascent](#)
- int [descent](#)
- [MFont](#) * [font](#)
- [MSymbol](#) [font_type](#)
- void * [fontp](#)

3.13.1 詳解

グリフの寸法とフォントに関する情報の型宣言.

[MDrawGlyph](#) 型はグリフの寸法とフォントに関する情報を含む構造体である。 [mdraw_glyph_list\(\)](#) はこれを用いる。

3.13.2 フィールド詳解

3.13.2.1 from

```
int MDrawGlyph::from
```

グリフに対応する文字の範囲.

3.13.2.2 to

```
int MDrawGlyph::to
```

3.13.2.3 glyph_code

```
int MDrawGlyph::glyph_code
```

フォント内のグリフコード。

3.13.2.4 x_advance

```
int MDrawGlyph::x_advance
```

グリフの論理的幅。次のグリフとの名目上の距離。

3.13.2.5 y_advance

```
int MDrawGlyph::y_advance
```

グリフの論理的高さ。次のグリフとの名目上の距離。

3.13.2.6 x_off

```
int MDrawGlyph::x_off
```

グリフの位置に対する X オフセット.

3.13.2.7 y_off

```
int MDrawGlyph::y_off
```

グリフの位置に対する Y オフセット.

3.13.2.8 lbearing

```
int MDrawGlyph::lbearing
```

グリフの寸法 (left-bearing).

3.13.2.9 rbearing

```
int MDrawGlyph::rbearing
```

グリフの寸法 (right-bearing).

3.13.2.10 ascent

```
int MDrawGlyph::ascent
```

グリフの寸法 (ascent).

3.13.2.11 descent

```
int MDrawGlyph::descent
```

グリフの寸法 (descent).

3.13.2.12 font

```
MFont* MDrawGlyph::font
```

グリフに使われるフォント。見つからなければ NULL。

3.13.2.13 font_type

```
MSymbol MDrawGlyph::font_type
```

フォントのタイプ。Mx、Mfreetype、Mxft のいずれか。

3.13.2.14 fontp

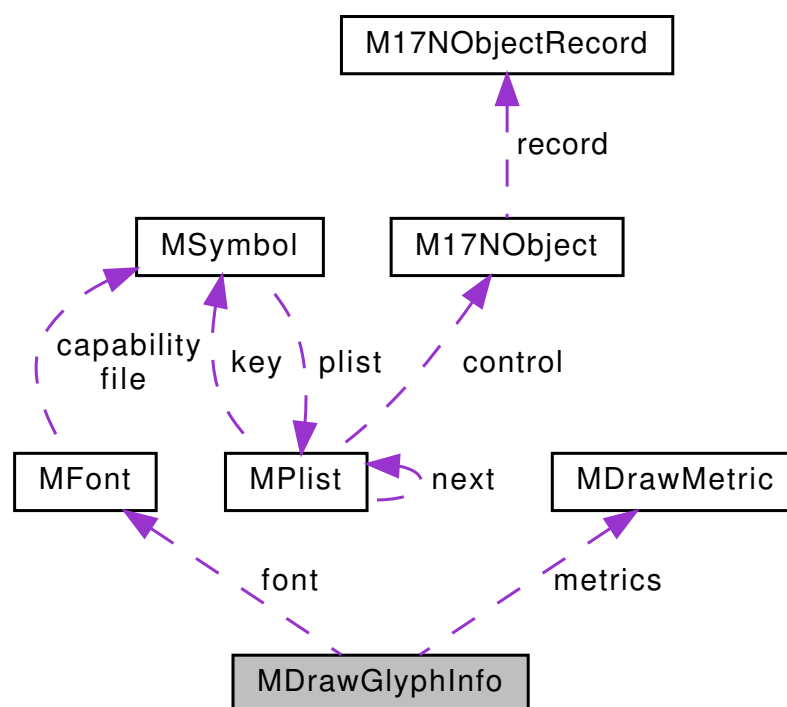
```
void* MDrawGlyph::fontp
```

フォントの構造体へのポインタ。実際の型は <font_type> メンバが Mx なら (XFontStruct *)、Mfreetype なら FT_Face、Mxft なら (XftFont *)。

3.14 MDrawGlyphInfo 構造体

グリフに関する情報の型宣言。

MDrawGlyphInfo 連携図



フィールド

- int from
- int to
- int line_from
- int line_to
- int x
- int y
- MDrawMetric metrics
- MFont * font
- int prev_from
- int next_to
- int left_from
- int left_to
- int right_from
- int right_to
- int logical_width

3.14.1 詳解

グリフに関する情報の型宣言.

`MDrawGlyphInfo` 型はグリフに関する情報を含む構造体である。 `mdraw_glyph_info()` はこれを用いる。

3.14.2 フィールド詳解

3.14.2.1 from

```
int MDrawGlyphInfo::from
```

グリフに対応する文字の範囲の開始位置.

3.14.2.2 to

```
int MDrawGlyphInfo::to
```

グリフに対応する文字の範囲の終了位置.

3.14.2.3 line_from

```
int MDrawGlyphInfo::line_from
```

一行のグリフの列に対応する文字の範囲の開始位置.

3.14.2.4 line_to

```
int MDrawGlyphInfo::line_to
```

一行のグリフの列に対応する文字の範囲の終了位置.

3.14.2.5 x

```
int MDrawGlyphInfo::x
```

グリフの X 座標.

3.14.2.6 y

```
int MDrawGlyphInfo::y
```

グリフの Y 座標.

3.14.2.7 metrics

```
MDrawMetric MDrawGlyphInfo::metrics
```

グリフの寸法.

3.14.2.8 font

```
MFont* MDrawGlyphInfo::font
```

グリフに使われるフォント。見つからなければ NULL。

3.14.2.9 prev_from

```
int MDrawGlyphInfo::prev_from
```

論理的な前のグリフに対応する文字の範囲。メンバ `prev_to` は、メンバ `from` と同じであるはずなので不要である。

3.14.2.10 next_to

```
int MDrawGlyphInfo::next_to
```

論理的な後のグリフに対応する文字の範囲。メンバ `next_from` はメンバ `to` と同じであるはずなので不要である。

3.14.2.11 left_from

```
int MDrawGlyphInfo::left_from
```

表示上の左のグリフに対応する文字の範囲の開始位置。

3.14.2.12 left_to

```
int MDrawGlyphInfo::left_to
```

表示上の左のグリフに対応する文字の範囲の終了位置。

3.14.2.13 right_from

```
int MDrawGlyphInfo::right_from
```

表示上の右のグリフに対応する文字の範囲の開始位置。

3.14.2.14 right_to

```
int MDrawGlyphInfo::right_to
```

表示上の右のグリフに対応する文字の範囲の終了位置。

3.14.2.15 logical_width

```
int MDrawGlyphInfo::logical_width
```

グリフの論理的幅。次のグリフとの名目上の距離。

3.15 MDrawMetric 構造体

グリフとテキストの寸法の型宣言。

フィールド

- int `x`
- int `y`
- unsigned int `width`
- unsigned int `height`

3.15.1 詳解

グリフとテキストの寸法の型宣言。

`MDrawMetric` はグリフと表示されたテキストの寸法用の型である。また、表示デバイスの矩形領域を表すのにも用いられる。

3.15.2 フィールド詳解

3.15.2.1 x

```
int MDrawMetric::x
```

X coordinates of a glyph or a text.

3.15.2.2 y

```
int MDrawMetric::y
```

Y coordinates of a glyph or a text.

3.15.2.3 width

```
unsigned int MDrawMetric::width
```

Pixel width of a glyph or a text.

3.15.2.4 height

```
unsigned int MDrawMetric::height
```

Pixel height of a glyph or a text.

3.16 MDrawPoint 構造体

フィールド

- short [x](#)
- short [y](#)

3.16.1 フィールド詳解

3.16.1.1 x

```
short MDrawPoint::x
```

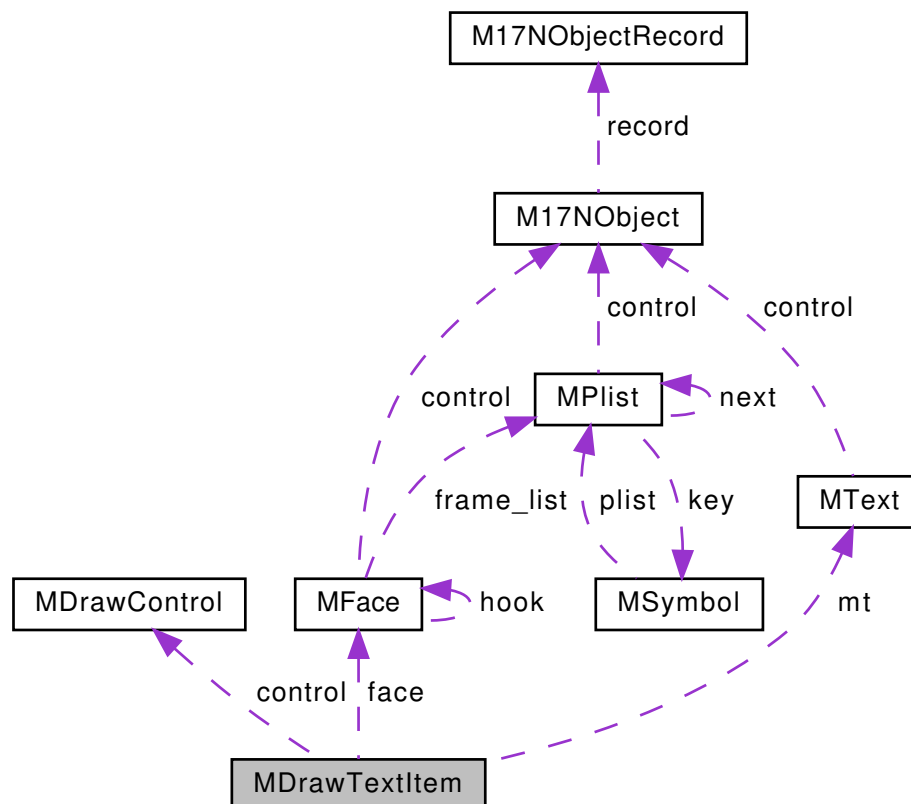
3.16.1.2 y

```
short MDrawPoint::y
```

3.17 MDrawTextItem 構造体

textitem の型宣言.

MDrawTextItem 連携図



フィールド

- [MText](#) * [mt](#)
- int [delta](#)
- [MFace](#) * [face](#)
- [MDrawControl](#) * [control](#)

3.17.1 詳解

textitem の型宣言.

[MDrawTextItem](#) はテキストアイテム オブジェクト用の型である。各テキストアイテムは、1 個の [M-text](#) と、その表示を制御するための情報を含んでいる。

3.17.2 フィールド詳解

3.17.2.1 mt

```
MText* MDrawTextItem::mt
```

M-text.

3.17.2.2 delta

```
int MDrawTextItem::delta
```

M-text 表示前に行なうX 軸方向の位置調整 (ピクセル単位)

3.17.2.3 face

```
MFace* MDrawTextItem::face
```

フェースオブジェクトへのポインタ。フェースの各プロパティは Mnil でなければ <mt> で指定されたフェースの同じプロパティに優先する

3.17.2.4 control

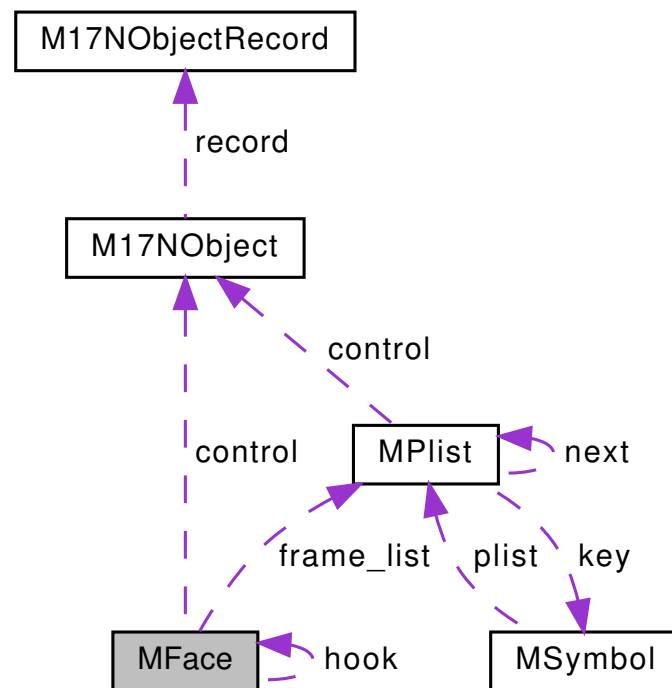
```
MDrawControl* MDrawTextItem::control
```

表示制御オブジェクトへのポインタ。 [mdraw_text_with_control\(\)](#) はこのオブジェクトを用いて M-text <mt> を表示する。

3.18 MFace 構造体

フェースの型宣言.

MFace 連携図



フィールド

- [M17NObject control](#)
- `void * property [MFACE_PROPERTY_MAX]`
- [MFaceHookFunc hook](#)
- [MPList * frame_list](#)

3.18.1 詳解

フェースの型宣言.

[MFace](#) 型はフェースオブジェクトのための構造体である。内部構造はアプリケーションプログラムからは見えない。

3.18.2 フィールド詳解

3.18.2.1 control

[M17NObject](#) `MFace::control`

3.18.2.2 property

```
void* MFace::property[MFACE_PROPERTY_MAX]
```

3.18.2.3 hook

```
MFaceHookFunc MFace::hook
```

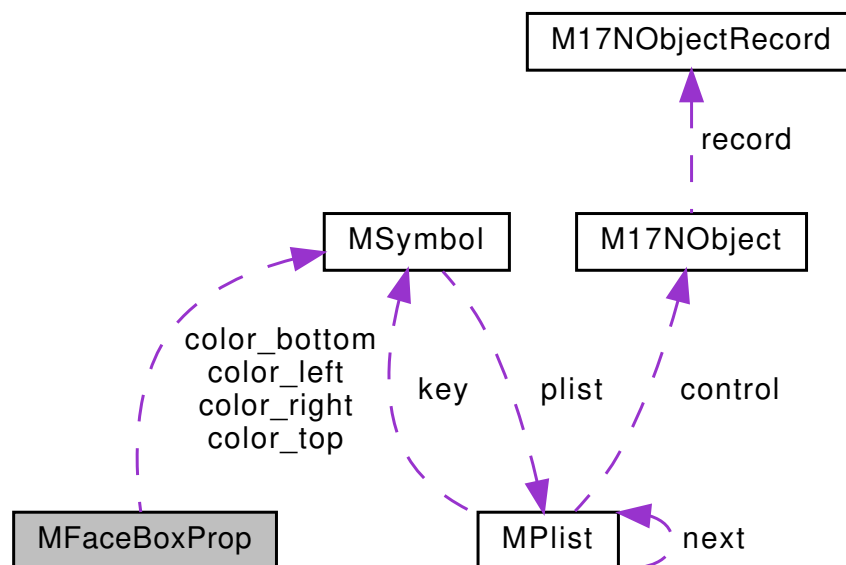
3.18.2.4 frame_list

```
MPlist* MFace::frame_list
```

3.19 MFaceBoxProp 構造体

フェースの囲み枠指定用型宣言.

MFaceBoxProp 連携図



フィールド

- unsigned width
- MSymbol color_top
- MSymbol color_bottom
- MSymbol color_left
- MSymbol color_right
- unsigned inner_hmargin
- unsigned inner_vmargin
- unsigned outer_hmargin
- unsigned outer_vmargin

3.19.1 詳解

フェースの囲み枠指定用型宣言.

`MFaceBoxProp` はフェースの `Mbox` プロパティの詳細を指定する型である。このプロパティの値はこの型のオブジェクトへのポインタでなくてはならない。

3.19.2 フィールド詳解

3.19.2.1 width

```
unsigned MFaceBoxProp::width
```

線幅（ピクセル単位）.

3.19.2.2 color_top

```
MSymbol MFaceBoxProp::color_top
```

Colors of borders.

3.19.2.3 color_bottom

```
MSymbol MFaceBoxProp::color_bottom
```

3.19.2.4 color_left

```
MSymbol MFaceBoxProp::color_left
```

3.19.2.5 color_right

```
MSymbol MFaceBoxProp::color_right
```


3.19.2.6 inner_hmargin

```
unsigned MFaceBoxProp::inner_hmargin
```

Margins

3.19.2.7 inner_vmargin

```
unsigned MFaceBoxProp::inner_vmargin
```

3.19.2.8 outer_hmargin

```
unsigned MFaceBoxProp::outer_hmargin
```

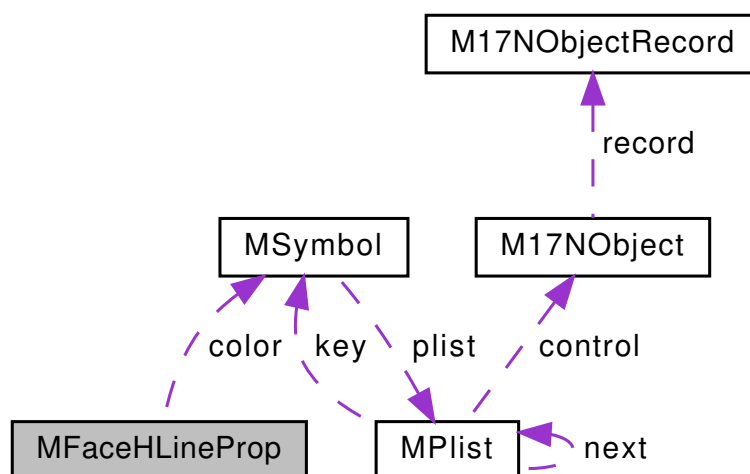
3.19.2.9 outer_vmargin

```
unsigned MFaceBoxProp::outer_vmargin
```

3.20 MFaceHLineProp 構造体

フェースの水平線指定用型宣言.

MFaceHLineProp 連携図



公開型

- enum `MFaceHLineType` {
 `MFACE_HLINE_BOTTOM` ,
 `MFACE_HLINE_UNDER` ,
 `MFACE_HLINE_STRIKE_THROUGH` ,
 `MFACE_HLINE_OVER` ,
 `MFACE_HLINE_TOP` }

フィールド

- enum `MFaceHLineProp::MFaceHLineType type`
- unsigned `width`
- `MSymbol color`

3.20.1 詳解

フェースの水平線指定用型宣言.

`MFaceHLineProp` はフェースの `Mhline` プロパティの詳細を指定する型である。このプロパティの値はこの型のオブジェクトでなくてはならない。

3.20.2 列挙型メンバ詳解

3.20.2.1 MFaceHLineType

enum `MFaceHLineProp::MFaceHLineType`

水平線のタイプ.

列挙値

<code>MFACE_HLINE_BOTTOM</code>	
<code>MFACE_HLINE_UNDER</code>	
<code>MFACE_HLINE_STRIKE_THROUGH</code>	
<code>MFACE_HLINE_OVER</code>	
<code>MFACE_HLINE_TOP</code>	

3.20.3 フィールド詳解

3.20.3.1 type

```
enum MFaceHLineProp::MFaceHLineType MFaceHLineProp::type
```

3.20.3.2 width

```
unsigned MFaceHLineProp::width
```

線幅（ピクセル単位）。

3.20.3.3 color

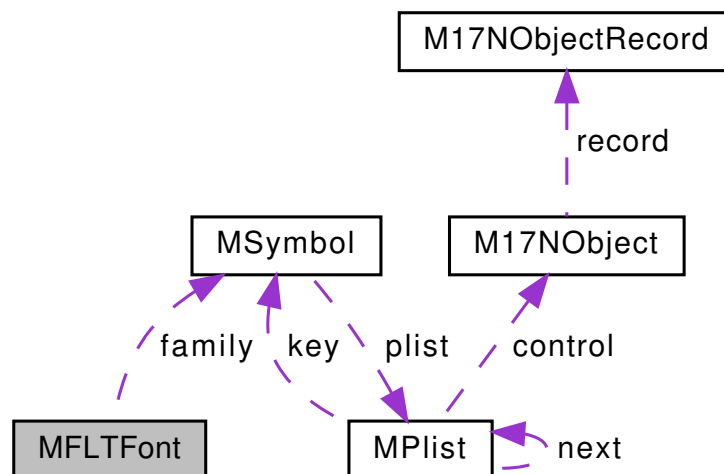
```
MSymbol MFaceHLineProp::color
```

線の色. Mnil ならば、統合したフェースの前景色が使われる。

3.21 MFLTFont 構造体

FLT ドライバが使うフォントの型.

MFLTFont 連携図



フィールド

- `MSymbol family`
- `int x_ppem`
- `int y_ppem`
- `int(* get_glyph_id)(struct _MFLTFont *font, MFLTGlyphString *gstring, int from, int to)`
- `int(* get_metrics)(struct _MFLTFont *font, MFLTGlyphString *gstring, int from, int to)`
- `int(* check_otf)(struct _MFLTFont *font, MFLTOtfSpec *spec)`
- `int(* drive_otf)(struct _MFLTFont *font, MFLTOtfSpec *spec, MFLTGlyphString *in, int from, int to, MFLTGlyphString *out, MFLTGlyphAdjustment *adjustment)`
- `void * internal`

3.21.1 詳解

FLT ドライバが使うフォントの型.

型 `MFLTFont` は、FLT ドライバが使うフォントに関する情報を格納するための構造体である。通常アプリケーションは最初の要素が `MFLTFont` で、残りの要素に `callback` 関数が利用するフォント情報を持った、より大きな構造体を用意し、それを `MFLTFont` に `coerce` して `mflt` の各関数に渡す。各 `callback` 関数は `MFLTFont` を元の構造体に `coerce` し直すことができることが保証されている。

3.21.2 フィールド詳解

3.21.2.1 family

`MSymbol MFLTFont::family`

フォントのファミリー名。フォントに適した FLT を探す際に重要でない場合 (たとえば OpenType フォントの場合など) は、`::Mnil` でよい。

3.21.2.2 x_ppem

`int MFLTFont::x_ppem`

フォントの水平サイズを `pixels per EM` で表現したもの。

3.21.2.3 y_ppem

`int MFLTFont::y_ppem`

フォントの垂直サイズを `pixels per EM` で表現したもの。

3.21.2.4 get_glyph_id

`int (* MFLTFont::get_glyph_id) (struct _MFLTFont *font, MFLTGlyphString *gstring, int from, int to)`

GSTRING 内の FROM から TO 直前までの各グリフに対応するグリフ ID を取得するための `callback` 関数。もしあるグリフのメンバー `<encoded>` がゼロならば、そのグリフのメンバー `<code>` は文字コードである。この関数はその文字コードを FONT のグリフ ID に変換しなくてはならない。

3.21.2.5 get_metrics

```
int(* MFLTFont::get_metrics) (struct _MFLTFont *font, MFLTGlyphString *gstring, int from, int to)
```

GSTRING 内の FROM から TO 直前までの各グリフに対応するメトリックを取得するための callback 関数。もしあるグリフのメンバー <measured> がゼロならば、この関数はそのグリフのメンバー <xadv>, <yadv>, <ascent>, <descent>, <lbearing>, および <rbearing> をセットしなければならない。

3.21.2.6 check_otf

```
int(* MFLTFont::check_otf) (struct _MFLTFont *font, MFLTOfSpec *spec)
```

フォントがある特定のスクリプト/言語に対する GSUB/GPOS OpenType フィーチャーを持つか否かを調べる callback 関数。この関数はフォントが SPEC を満たすときは 1 を、そうでないときは 0 を返さなければならない。フォントが OpenType テーブルを持たないときは NULL でなければならない。

3.21.2.7 drive_otf

```
int(* MFLTFont::drive_otf) (struct _MFLTFont *font, MFLTOfSpec *spec, MFLTGlyphString *in, int from, int to, MFLTGlyphString *out, MFLTGlyphAdjustment *adjustment)
```

IN 内の FROM から TO 直前までの各グリフに SPEC 内の各 OpenType フィーチャーを適用するための callback 関数。適用結果のグリフ列は OUT の末尾に追加される。OUT が短か過ぎて結果を追加し切れない場合は -2 を返さなくてはならない。フォントが OpenType テーブルを持たない場合は NULL でなければならない。

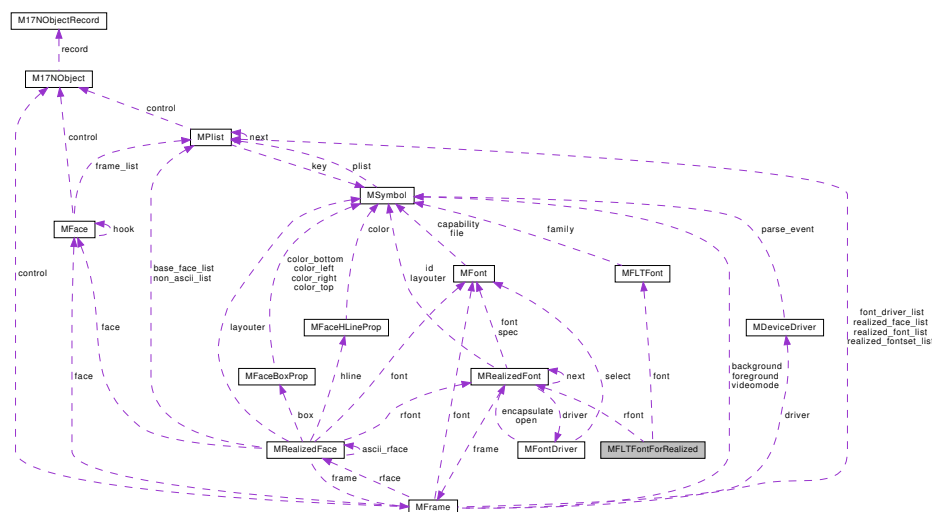
3.21.2.8 internal

```
void* MFLTFont::internal
```

m17n-lib の内部作業用。NULL に初値化される。

3.22 MFLTFontForRealized 構造体

MFLTFontForRealized 連携図



フィールド

- [MFLTFont font](#)
- [MRealizedFont * rfont](#)

3.22.1 フィールド詳解

3.22.1.1 font

[MFLTFont](#) MFLTFontForRealized::font

3.22.1.2 rfont

[MRealizedFont*](#) MFLTFontForRealized::rfont

3.23 MFLTGlyph 構造体

グリフに関する情報の型.

フィールド

- int [c](#)
- unsigned int [code](#)
- int [from](#)
- int [to](#)
- int [xadv](#)
- int [yadv](#)
- int [ascent](#)
- int [descent](#)
- int [lbearing](#)
- int [rbearing](#)
- int [xoff](#)
- int [yoff](#)
- unsigned [encoded](#): 1
- unsigned [measured](#): 1
- unsigned [adjusted](#): 1
- unsigned [internal](#): 30

3.23.1 詳解

グリフに関する情報の型.

型 **MFLTGlyph** は、グリフに関する情報を格納する構造体である。関数 **mflt_find()** と **mflt_run()** を呼ぶ前にはメンバー **<c>** と **<encoded>** を適切に設定しておかねばならず、もし **<encoded>** を 1 とした場合は **<code>** も設定しておかねばならない。

3.23.2 フィールド詳解

3.23.2.1 c

```
int MFLTGlyph::c
```

グリフの (Unicode における) 文字コード。

3.23.2.2 code

```
unsigned int MFLTGlyph::code
```

フォント内におけるそのグリフの ID。

3.23.2.3 from

```
int MFLTGlyph::from
```

MFLTGlyphString の中で、このグリフによって置き換えられる部分の先頭のインデクス。

3.23.2.4 to

```
int MFLTGlyph::to
```

MFLTGlyphString の中で、このグリフによって置き換えられる部分の末尾のインデクス。

3.23.2.5 xadv

```
int MFLTGlyph::xadv
```

横書き時の送り幅を 26.6 fractional pixel format で表現したもの。

3.23.2.6 yadv

```
int MFLTGlyph::yadv
```

縦書き時の送り高を 26.6 fractional pixel format で表現したもの。

3.23.2.7 ascent

```
int MFLTGlyph::ascent
```

このグリフのインクメトリックを 26.6 fractional pixel format で表現したもの。

3.23.2.8 descent

```
int MFLTGlyph::descent
```

3.23.2.9 lbearing

```
int MFLTGlyph::lbearing
```

3.23.2.10 rbearing

```
int MFLTGlyph::rbearing
```

3.23.2.11 xoff

```
int MFLTGlyph::xoff
```

グリフ位置決めの際の水平・垂直調整値を、26.6 fractional pixel format で表現したもの。

3.23.2.12 yoff

```
int MFLTGlyph::yoff
```


3.23.2.13 encoded

`unsigned MFLTGlyph::encoded`

メンバー `<code>` に既にグリフ ID がセットされているか否かを示すフラグ。

3.23.2.14 measured

`unsigned MFLTGlyph::measured`

メンバー `<xadv>` から `<rbearing>` までの各メトリックが既に計算済か否かを示すフラグ。

3.23.2.15 adjusted

`unsigned MFLTGlyph::adjusted`

グリフのメトリックが調整済みか否か、すなわち以下のうち 1 つ以上が成立していることを示すフラグ。
`<xadv>` が標準の値と異なる、`<yadv>` が標準の値と異なる、`<xoff>` がゼロでない、`<yoff>` がゼロでない。

3.23.2.16 internal

`unsigned MFLTGlyph::internal`

m17n-lib 内部作業用。

3.24 MFLTGlyphAdjustment 構造体

グリフ位置調整情報のための型。

フィールド

- `int xadv`
- `int yadv`
- `int xoff`
- `int yoff`
- `short back`
- `unsigned advance_js_absolute: 1`
- `unsigned set: 1`

3.24.1 詳解

グリフ位置調整情報のための型。

型 `MFLTGlyphAdjustment` は、グリフのメトリック/位置の調整に関する情報を格納するための構造体であり、`MFLTFont` の callback 関数 `drive_otf` に渡される。

3.24.2 フィールド詳解

3.24.2.1 xadv

```
int MFLTGlyphAdjustment::xadv
```

水平・垂直方向の送り量の調整値を 26.6 fractional pixel format で表現したもの。

3.24.2.2 yadv

```
int MFLTGlyphAdjustment::yadv
```

3.24.2.3 xoff

```
int MFLTGlyphAdjustment::xoff
```

グリフ位置決めのための水平・垂直調整値を 26.6 fractional pixel format で表現したもの。

3.24.2.4 yoff

```
int MFLTGlyphAdjustment::yoff
```

3.24.2.5 back

```
short MFLTGlyphAdjustment::back
```

グリフ描画のために戻るべきグリフ数。

3.24.2.6 advance_is_absolute

```
unsigned MFLTGlyphAdjustment::advance_is_absolute
```

非ゼロのとき、メンバー <xadv> と <yadv> は絶対値である。すなわちその値をグリフ本来の送り幅に加算してはならない。

3.24.2.7 set

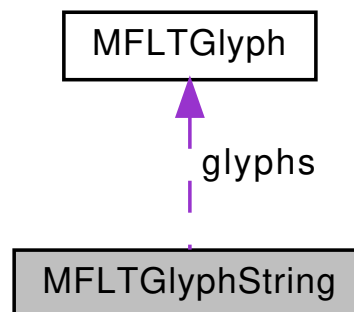
```
unsigned MFLTGlyphAdjustment::set
```

他のメンバーのうち最低 1 個が非ゼロのときのみ、1 にセットされる。

3.25 MFLTGlyphString 構造体

グリフ列の情報のための型.

MFLTGlyphString 連携図



フィールド

- int `glyph_size`
- `MFLTGlyph * glyphs`
- int `allocated`
- int `used`
- unsigned int `r2l`

3.25.1 詳解

グリフ列の情報のための型.

型 `MFLTGlyphString` は、グリフ列の情報を格納するための構造体である。

3.25.2 フィールド詳解

3.25.2.1 glyph_size

```
int MFLTGlyphString::glyph_size
```

メンバー `glyphs` の指す配列の要素が占める実バイト数。この値は "sizeof (MFLTGlyph)" 以上でなければならない。

3.25.2.2 glyphs

```
MFLTGlyph* MFLTGlyphString::glyphs
```

グリフの配列。

3.25.2.3 allocated

```
int MFLTGlyphString::allocated
```

`glyphs` 内に配置されている要素の数。

3.25.2.4 used

```
int MFLTGlyphString::used
```

`glyphs` 内で使用中の要素の数。

3.25.2.5 r2l

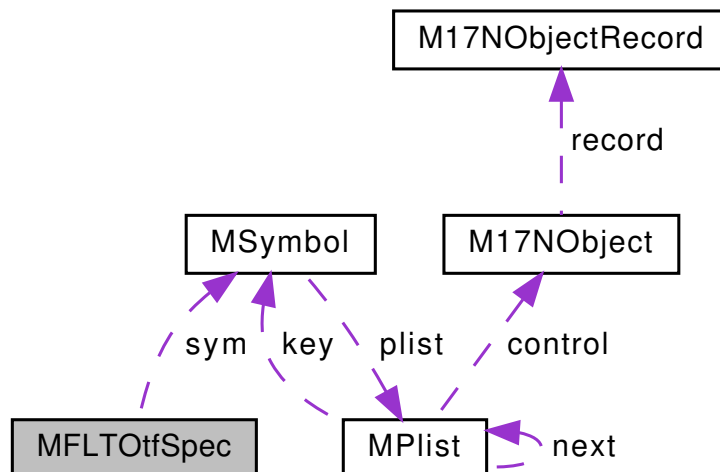
```
unsigned int MFLTGlyphString::r2l
```

グリフが右から左へと描かれるべきか否かを示すフラグ。

3.26 MFLTOfSpec 構造体

GSUB および GPOS OpenType テーブルの仕様のための型。

MFLTOfSpec 連携図



フィールド

- [MSymbol sym](#)
- unsigned int [script](#)
- unsigned int [langsys](#)
- unsigned int * [features](#) [2]

3.26.1 詳解

GSUB および GPOS OpenType テーブルの仕様のための型.

型 [MFLTOfSpec](#) は、GSUB および GPOS フィーチャーの情報を格納するための構造体である。これらフィーチャーは特定のスクリプトおよび言語システムのものである。この情報は、どのフィーチャーをグリフ列に適用するか、あるいは特定の FLT が特定のフォントに対して有効かどうかの決定に使用される。

3.26.2 フィールド詳解

3.26.2.1 sym

[MSymbol](#) MFLTOfSpec::sym

この仕様を表わすユニークなシンボル。FLT の [OTF-SPEC](#) と同一の値である。

3.26.2.2 script

unsigned int MFLTOfSpec::script

スクリプトおよび言語システムのタグ。

3.26.2.3 langsys

unsigned int MFLTOfSpec::langsys

3.26.2.4 features

```
unsigned int* MFLTOtfSpec::features[2]
```

GSUB フィーチャータグの配列を第 1 要素、GPOS フィーチャータグの配列を第 2 要素とする配列。各配列の末尾は 0 で示される。フィーチャー の指定が 1 つもない場合はこの配列の要素は NULL でもよい。

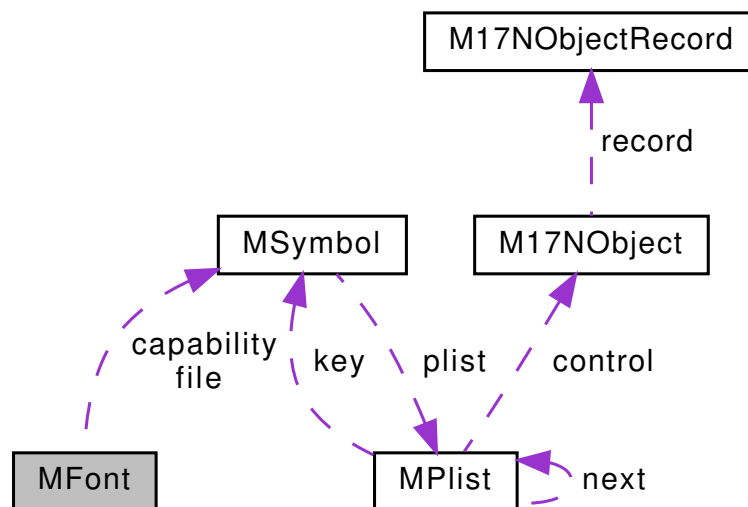
(1) この情報がグリフ列に適用すべきフィーチャーの選択に使われる場合。もし配列自身がNULL なら、どのフィーチャーも適用しない。もし最初の要素が 0xFFFFFFFF なら、2 番目以降のフィーチャー（もしあれば）を除くすべての適用可能なフィーチャーを適用する。それ以外の場合 リストされたすべてのフィーチャーを適用する。

(2) この情報が特定の FLT が特定のフォントに有効かどうかの決定に使われる場合。もし配列自身がNULL なら、フォントはフィーチャーを一つも持っていないといけない。もし最初の要素が 0xFFFFFFFF なら、フォントは 2 番目の要素以降のフォントを持っていないといけない。それ以外の場合、フォントは 0xFFFFFFFF 以前のすべてのフィーチャーを持ち、かつ 0xFFFFFFFF 以降のフィーチャーは一つも持っていないといけない。

3.27 MFont 構造体

フォントの型宣言.

MFont 連携図



フィールド

- unsigned short [property](#) [MFONT_PROPERTY_MAX]
- unsigned [type](#): 2
- unsigned [source](#): 2
- unsigned [spacing](#): 2
- unsigned [for_full_width](#): 1
- unsigned [multiple_sizes](#): 1
- unsigned [size](#): 24
- [MSymbol file](#)
- [MSymbol capability](#)
- [MFontEncoding](#) * [encoding](#)

3.27.1 詳解

フォントの型宣言.

MFont 型はフォント指定用の構造体であり、フォントのプロパティである `foundry`, `family`, `weight`, `style`, `stretch`, `adstyle`, `registry`, `size`, `resolution` に関する情報を含む。

この構造体はフォントセット内のフォントを指定する際と、使用可能なシステムフォントの情報を格納する際の両方に用いられる。

内部構造はアプリケーションプログラムからは見えない。

参照:

[mfont\(\)](#), [mfont_from_name\(\)](#), [mfont_find\(\)](#).

3.27.2 フィールド詳解

3.27.2.1 property

```
unsigned short MFont::property[MFONT_PROPERTY_MAX]
```

3.27.2.2 type

```
unsigned MFont::type
```

3.27.2.3 source

```
unsigned MFont::source
```

3.27.2.4 spacing

```
unsigned MFont::spacing
```

3.27.2.5 for_full_width

`unsigned MFont::for_full_width`

3.27.2.6 multiple_sizes

`unsigned MFont::multiple_sizes`

3.27.2.7 size

`unsigned MFont::size`

3.27.2.8 file

`MSymbol MFont::file`

3.27.2.9 capability

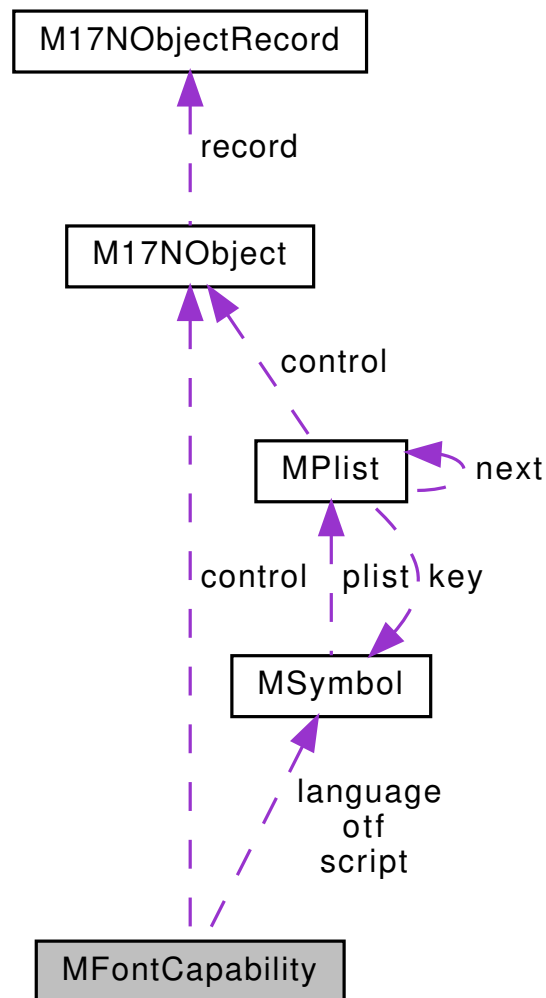
`MSymbol MFont::capability`

3.27.2.10 encoding

`MFontEncoding* MFont::encoding`

3.28 MFontCapability 構造体

MFontCapability 連携図



フィールド

- M17NObject control
- MSymbol language
- MSymbol script
- MSymbol otf
- OTF_Tag script_tag
- OTF_Tag langsys_tag
- struct {
 - char * str
 - int nfeatures
 - OTF_Tag * tags
 } features [MFONT_OTT_MAX]

3.28.1 フィールド詳解

3.28.1.1 control

`M17NObject` MFontCapability::control

3.28.1.2 language

`MSymbol` MFontCapability::language

3.28.1.3 script

`MSymbol` MFontCapability::script

3.28.1.4 otf

`MSymbol` MFontCapability::otf

3.28.1.5 script_tag

`OTF_Tag` MFontCapability::script_tag

3.28.1.6 langsys_tag

`OTF_Tag` MFontCapability::langsys_tag

3.28.1.7 str

```
char* MFontCapability::str
```

3.28.1.8 nfeatures

```
int MFontCapability::nfeatures
```

3.28.1.9 tags

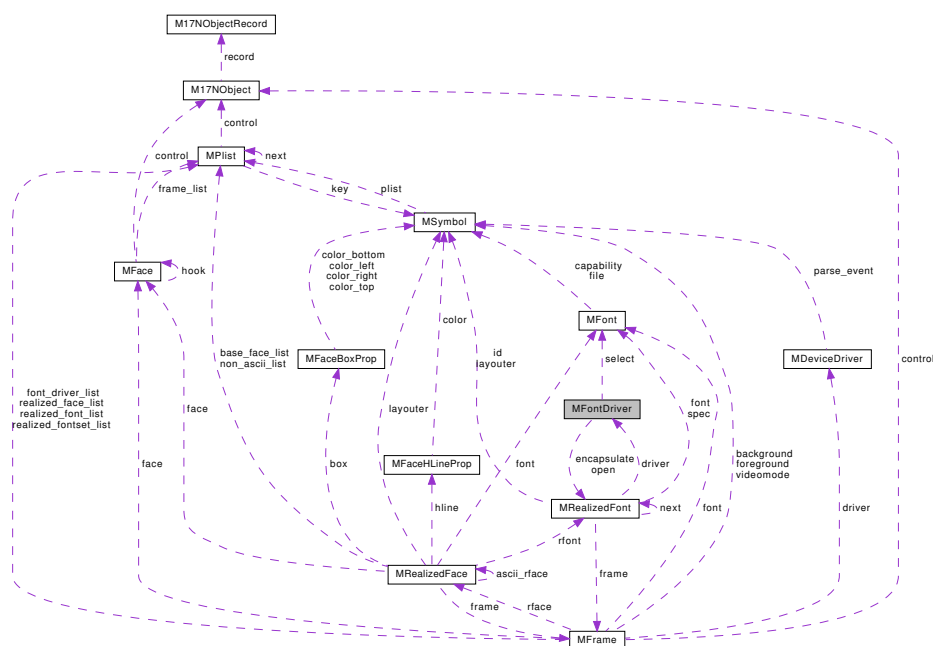
```
OTF_Tag* MFontCapability::tags
```

3.28.1.10

```
struct { ... } MFontCapability::features[MFONT_OTT_MAX]
```

3.29 MFontDriver 構造体

MFontDriver 連携図



フィールド

- `MFont *(* select)(MFrame *frame, MFont *font, int limited_size)`
- `MRealizedFont *(* open)(MFrame *frame, MFont *font, MFont *spec, MRealizedFont *rfont)`
- `void(* find_metric)(MRealizedFont *rfont, MGlyphString *gstring, int from, int to)`
- `int(* has_char)(MFrame *frame, MFont *font, MFont *spec, int c, unsigned code)`
- `unsigned(* encode_char)(MFrame *frame, MFont *font, MFont *spec, unsigned code)`
- `void(* render)(MDrawWindow win, int x, int y, MGlyphString *gstring, MGlyph *from, MGlyph *to, int reverse, MDrawRegion region)`
- `int(* list)(MFrame *frame, MPlist *plist, MFont *font, int maxnum)`
- `void(* list_family_names)(MFrame *frame, MPlist *plist)`
- `int(* check_capability)(MRealizedFont *rfont, MSymbol capability)`
- `MRealizedFont *(* encapsulate)(MFrame *frame, MSymbol source, void *data)`
- `void(* close)(MRealizedFont *rfont)`
- `int(* check_otf)(MFLTFont *font, MFLTOtfSpec *spec)`
- `int(* drive_otf)(MFLTFont *font, MFLTOtfSpec *spec, MFLTGlyphString *in, int from, int to, MFLTGlyphString *out, MFLTGlyphAdjustment *adjustment)`
- `int(* try_otf)(MFLTFont *font, MFLTOtfSpec *spec, MFLTGlyphString *in, int from, int to)`
- `int(* iterate_otf_feature)(struct _MFLTFont *font, MFLTOtfSpec *spec, int from, int to, unsigned char *table)`

3.29.1 フィールド詳解

3.29.1.1 select

```
MFont *(* MFontDriver::select) (MFrame *frame, MFont *font, int limited_size)
```

3.29.1.2 open

```
MRealizedFont *(* MFontDriver::open) (MFrame *frame, MFont *font, MFont *spec, MRealizedFont *rfont)
```

3.29.1.3 find_metric

```
void(* MFontDriver::find_metric) (MRealizedFont *rfont, MGlyphString *gstring, int from, int to)
```

3.29.1.4 has_char

```
int(* MFontDriver::has_char) (MFrame *frame, MFont *font, MFont *spec, int c, unsigned code)
```

3.29.1.5 encode_char

```
unsigned(* MFontDriver::encode_char) (MFrame *frame, MFont *font, MFont *spec, unsigned code)
```

3.29.1.6 render

```
void(* MFontDriver::render) (MDrawWindow win, int x, int y, MGlyphString *gstring, MGlyph *from, MGlyph *to, int reverse, MDrawRegion region)
```

3.29.1.7 list

```
int(* MFontDriver::list) (MFrame *frame, MPlist *plist, MFont *font, int maxnum)
```

3.29.1.8 list_family_names

```
void(* MFontDriver::list_family_names) (MFrame *frame, MPlist *plist)
```

3.29.1.9 check_capability

```
int(* MFontDriver::check_capability) (MRealizedFont *rfont, MSymbol capability)
```

3.29.1.10 encapsulate

```
MRealizedFont*(* MFontDriver::encapsulate) (MFrame *frame, MSymbol source, void *data)
```

3.29.1.11 close

```
void(* MFontDriver::close) (MRealizedFont *rfont)
```

3.29.1.12 check_otf

```
int(* MFontDriver::check_otf) (MFLTFont *font, MFLTOfSpec *spec)
```

3.29.1.13 drive_otf

```
int(* MFontDriver::drive_otf) (MFLTFont *font, MFLTOfSpec *spec, MFLTGlyphString *in, int  
from, int to, MFLTGlyphString *out, MFLTGlyphAdjustment *adjustment)
```

3.29.1.14 try_otf

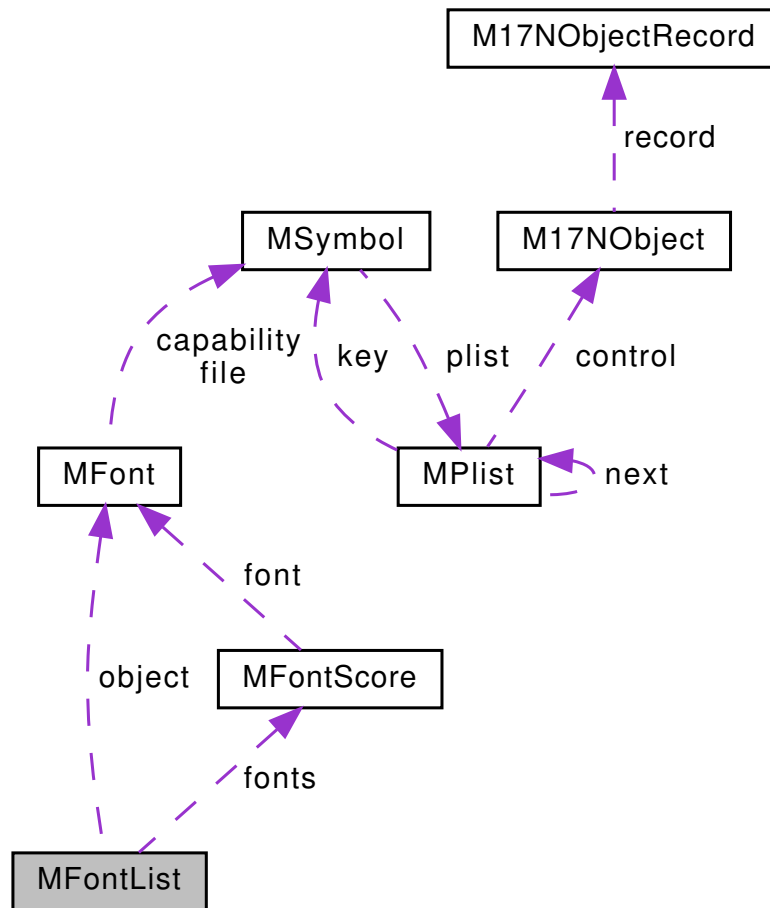
```
int(* MFontDriver::try_otf) (MFLTFont *font, MFLTOfSpec *spec, MFLTGlyphString *in, int from,  
int to)
```

3.29.1.15 iterate_otf_feature

```
int(* MFontDriver::iterate_otf_feature) (struct _MFLTFont *font, MFLTOfSpec *spec, int from,  
int to, unsigned char *table)
```

3.30 MFontList 構造体

MFontList 連携図



フィールド

- [MFont object](#)
- [MFontScore * fonts](#)
- [int nfonts](#)

3.30.1 フィールド詳解

3.30.1.1 object

[MFont](#) MFontList::object

3.30.1.2 fonts

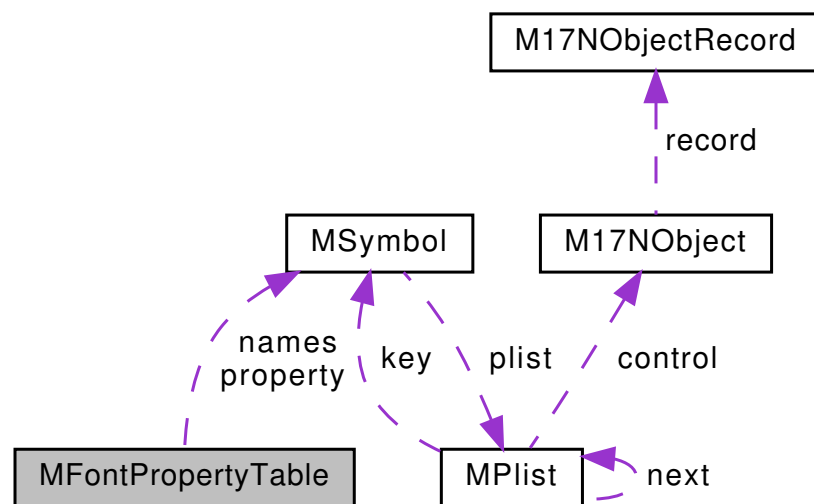
```
MSymbol* MFontList::fonts
```

3.30.1.3 nfonts

```
int MFontList::nfonts
```

3.31 MFontPropertyTable 構造体

MFontPropertyTable 連携図



フィールド

- int [size](#)
- int [inc](#)
- int [used](#)
- [MSymbol](#) [property](#)
- [MSymbol](#) * [names](#)

3.31.1 フィールド詳解

3.31.1.1 size

```
int MFontPropertyTable::size
```

3.31.1.2 inc

```
int MFontPropertyTable::inc
```

3.31.1.3 used

```
int MFontPropertyTable::used
```

3.31.1.4 property

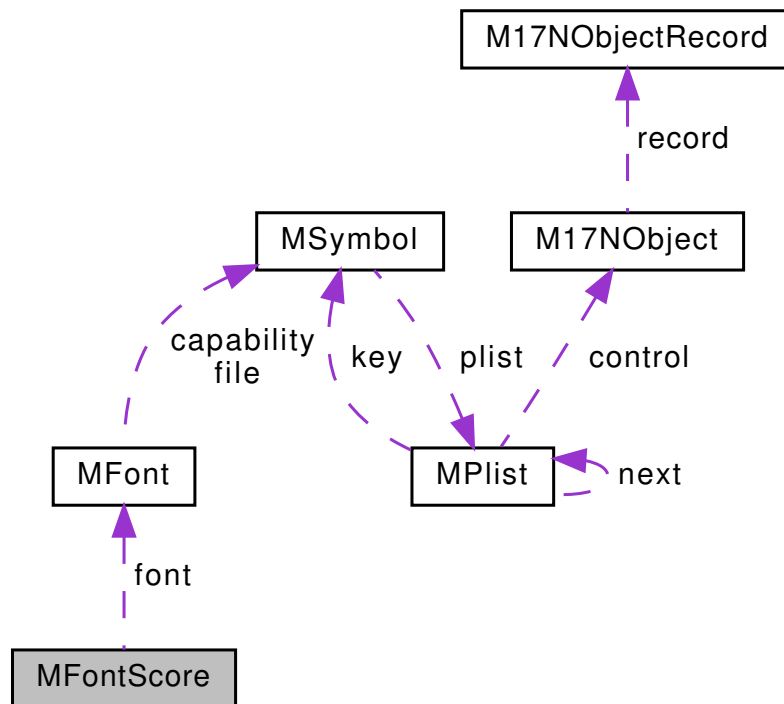
```
MSymbol MFontPropertyTable::property
```

3.31.1.5 names

```
MSymbol* MFontPropertyTable::names
```

3.32 MFontScore 構造体

MFontScore 連携図



フィールド

- **MFont** * **font**
- int **score**

3.32.1 フィールド詳解

3.32.1.1 font

MFont* MFontScore::font

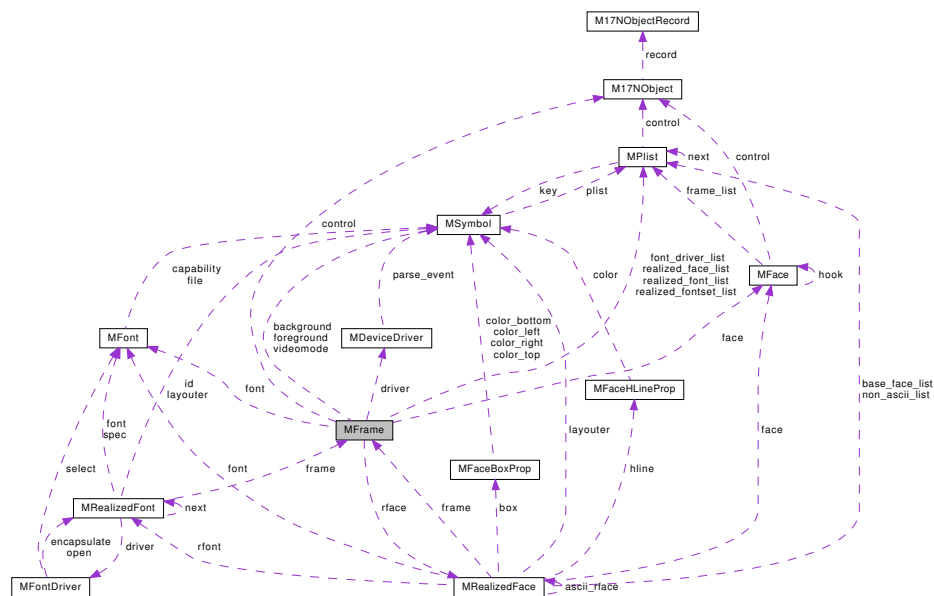
3.32.1.2 score

int MFontScore::score

3.33 MFrame 構造体

フレームの型宣言.

MFrame 連携図



フィールド

- M17Object control
- MSymbol foreground
- MSymbol background
- MSymbol videomode
- MFont * font
- MFace * face
- MRealizedFace * rface
- int space_width
- int average_width
- int ascent
- int descent
- unsigned tick
- void * device
- int device_type
- int dpi
- MDeviceDriver * driver
- MPlist * font_driver_list
- MPlist * realized_font_list
- MPlist * realized_face_list
- MPlist * realized_fontset_list

3.33.1 詳解

フレームの型宣言.

MFrame は、フレーム オブジェクト用の型である。個々のフレームは、それに対応する物理的な表示／入力デバイスの各種情報を保持する。

MFrame 型の内部構造は、アプリケーションプログラムからは見えない。またその内容は使用するウィンドウシステムに依存する。また **m17n-X** ライブラリにおけるフレームは、X ウィンドウの **display** と **screen** に関する情報を持つ。

3.33.2 フィールド詳解

3.33.2.1 control

M17NObject MFrame::control

3.33.2.2 foreground

MSymbol MFrame::foreground

3.33.2.3 background

MSymbol MFrame::background

3.33.2.4 videomode

MSymbol MFrame::videomode

3.33.2.5 font

MFont* MFrame::font

3.33.2.6 face

```
MFace* MFrame::face
```

3.33.2.7 rface

```
MRealizedFace* MFrame::rface
```

3.33.2.8 space_width

```
int MFrame::space_width
```

3.33.2.9 average_width

```
int MFrame::average_width
```

3.33.2.10 ascent

```
int MFrame::ascent
```

3.33.2.11 descent

```
int MFrame::descent
```

3.33.2.12 tick

```
unsigned MFrame::tick
```

3.33.2.13 device

```
void* MFrame::device
```

3.33.2.14 device_type

```
int MFrame::device_type
```

3.33.2.15 dpi

```
int MFrame::dpi
```

3.33.2.16 driver

```
MDeviceDriver* MFrame::driver
```

3.33.2.17 font_driver_list

```
MList* MFrame::font_driver_list
```

3.33.2.18 realized_font_list

```
MList* MFrame::realized_font_list
```

3.33.2.19 realized_face_list

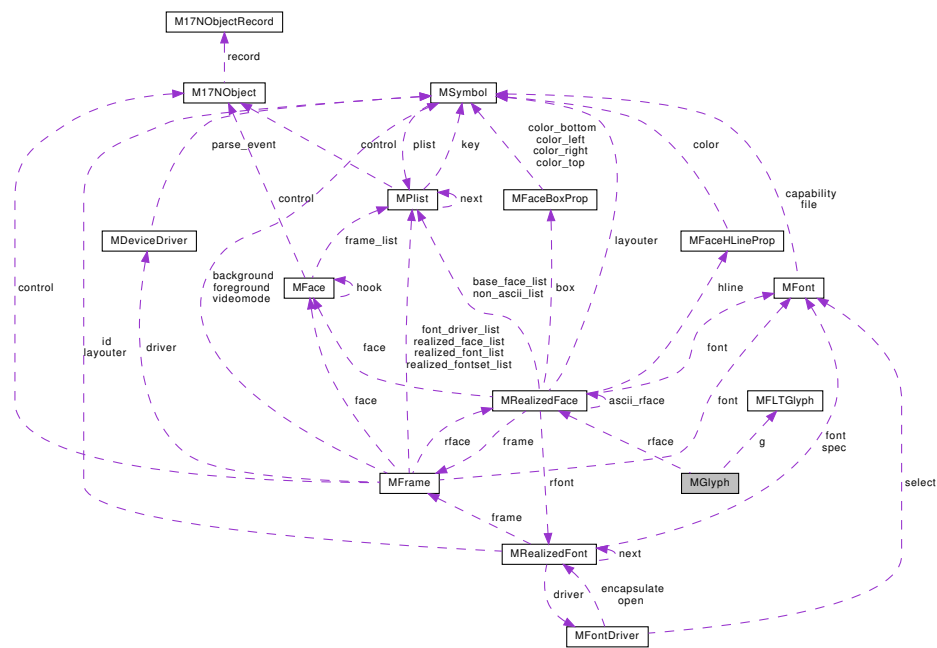
```
MList* MFrame::realized_face_list
```

3.33.2.20 realized_fontset_list

```
MPlist* MFrame::realized_fontset_list
```

3.34 MGlyph 構造体

MGlyph 連携図



フィールド

- [MFLTGlyph g](#)
- [MRealizedFace * rface](#)
- unsigned [left_padding](#): 1
- unsigned [right_padding](#): 1
- unsigned [enabled](#): 1
- unsigned [bidi_level](#): 6
- unsigned [category](#): 2
- unsigned [type](#): 3
- unsigned [libotf_positioning_type](#)

3.34.1 フィールド詳解

3.34.1.1 g

`MFLTGlyph` `MGlyph::g`

3.34.1.2 rface

`MRealizedFace*` `MGlyph::rface`

3.34.1.3 left_padding

`unsigned` `MGlyph::left_padding`

3.34.1.4 right_padding

`unsigned` `MGlyph::right_padding`

3.34.1.5 enabled

`unsigned` `MGlyph::enabled`

3.34.1.6 bidi_level

`unsigned` `MGlyph::bidi_level`

3.34.1.7 category

`unsigned` `MGlyph::category`

3.34.1.8 type

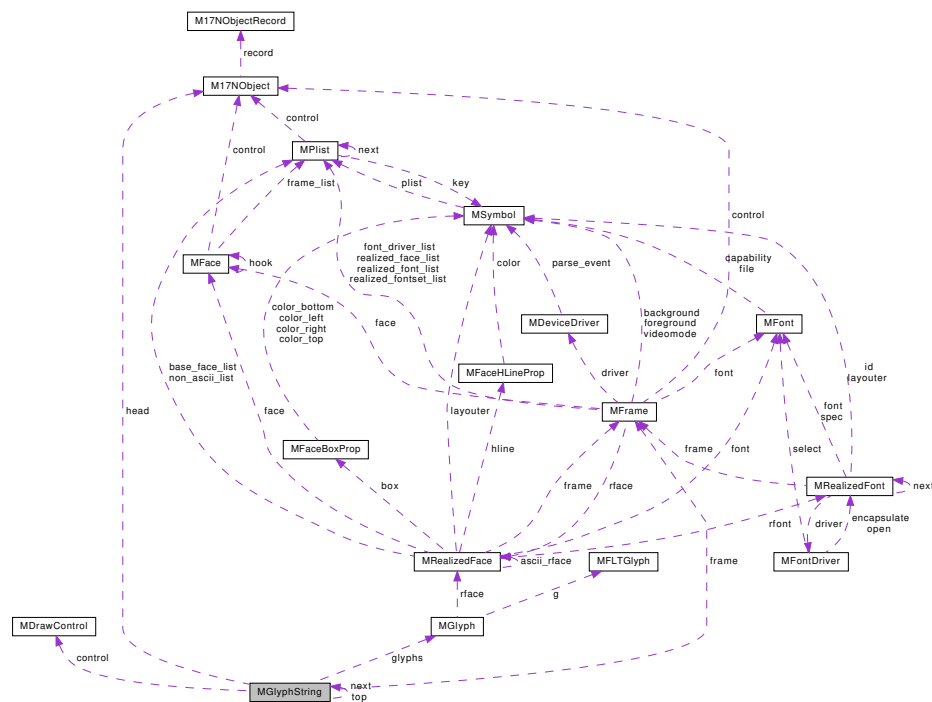
```
unsigned MGlyph::type
```

3.34.1.9 libotf_positioning_type

```
unsigned MGlyph::libotf_positioning_type
```

3.35 MGlyphString 構造体

MGlyphString 連携図



フィールド

- M17NObject head
- MFrame * frame
- int tick
- int size
- int inc
- int used
- MGlyph * glyphs
- int from
- int to

- short [width](#)
- short [height](#)
- short [ascent](#)
- short [descent](#)
- short [physical_ascent](#)
- short [physical_descent](#)
- short [lbearing](#)
- short [rbearing](#)
- short [text_ascent](#)
- short [text_descent](#)
- short [line_ascent](#)
- short [line_descent](#)
- int [indent](#)
- int [width_limit](#)
- unsigned [anti_alias](#): 1
- [MDrawControl](#) [control](#)
- struct [MGlyphString](#) * [next](#)
- struct [MGlyphString](#) * [top](#)

3.35.1 フィールド詳解

3.35.1.1 head

[M17NObject](#) [MGlyphString::head](#)

3.35.1.2 frame

[MFrame*](#) [MGlyphString::frame](#)

3.35.1.3 tick

int [MGlyphString::tick](#)

3.35.1.4 size

int [MGlyphString::size](#)

3.35.1.5 inc

```
int MGlyphString::inc
```

3.35.1.6 used

```
int MGlyphString::used
```

3.35.1.7 glyphs

```
MGlyph* MGlyphString::glyphs
```

3.35.1.8 from

```
int MGlyphString::from
```

3.35.1.9 to

```
int MGlyphString::to
```

3.35.1.10 width

```
short MGlyphString::width
```

3.35.1.11 height

```
short MGlyphString::height
```

3.35.1.12 ascent

```
short MGlyphString::ascent
```

3.35.1.13 descent

```
short MGlyphString::descent
```

3.35.1.14 physical_ascent

```
short MGlyphString::physical_ascent
```

3.35.1.15 physical_descent

```
short MGlyphString::physical_descent
```

3.35.1.16 lbearing

```
short MGlyphString::lbearing
```

3.35.1.17 rbearing

```
short MGlyphString::rbearing
```

3.35.1.18 text_ascent

```
short MGlyphString::text_ascent
```

3.35.1.19 text_descent

```
short MGlyphString::text_descent
```

3.35.1.20 line_ascent

```
short MGlyphString::line_ascent
```

3.35.1.21 line_descent

```
short MGlyphString::line_descent
```

3.35.1.22 indent

```
int MGlyphString::indent
```

3.35.1.23 width_limit

```
int MGlyphString::width_limit
```

3.35.1.24 anti_alias

```
unsigned MGlyphString::anti_alias
```

3.35.1.25 control

```
MDrawControl MGlyphString::control
```

3.35.1.26 next

```
struct MGlyphString* MGlyphString::next
```

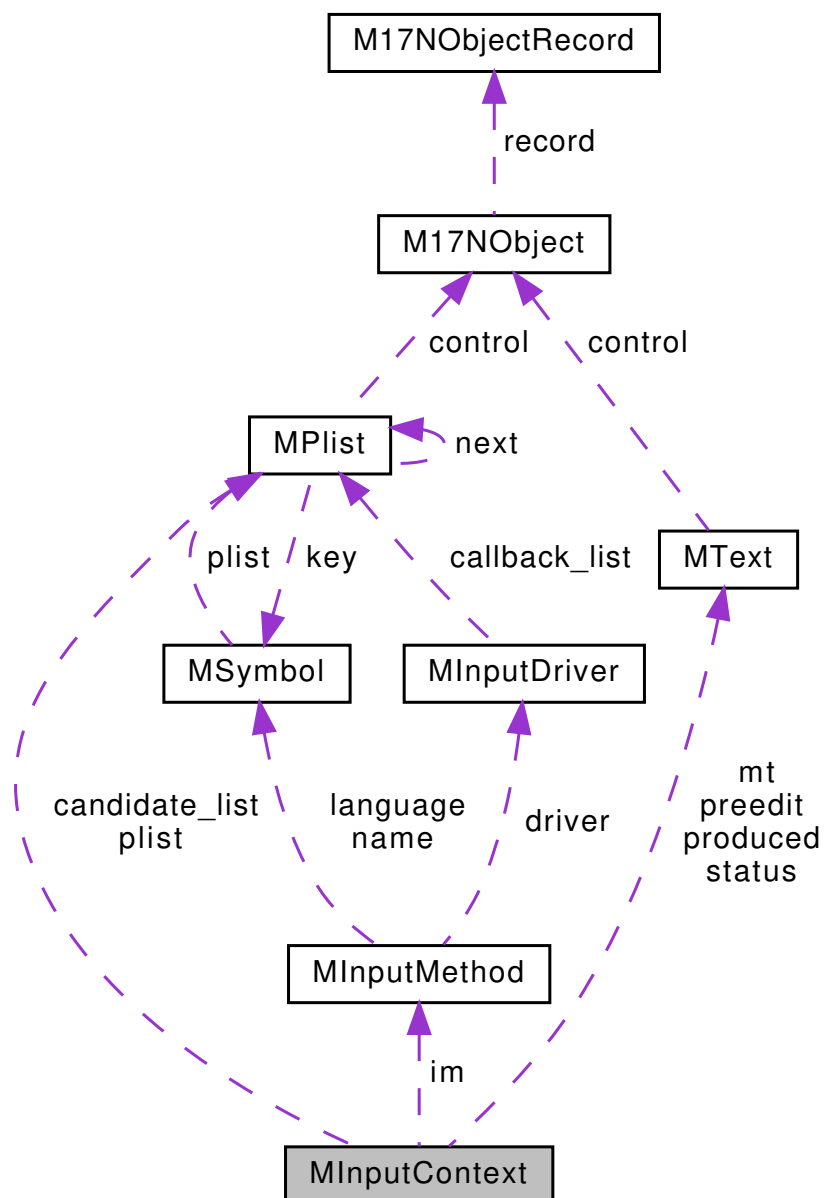
3.35.1.27 top

```
struct MGlyphString * MGlyphString::top
```

3.36 MInputContext 構造体

入力コンテキスト用構造体.

MInputContext 連携図



フィールド

- [MInputMethod](#) * [im](#)
- [MText](#) * [produced](#)
- void * [arg](#)
- int [active](#)
- struct {
 - int [x](#)
 - int [y](#)
 - int [ascent](#)
 - int [descent](#)
 - int [fontsize](#)
 - [MText](#) * [mt](#)
 - int [pos](#)
- } [spot](#)
- void * [info](#)
- [MText](#) * [status](#)
- int [status_changed](#)
- [MText](#) * [preedit](#)
- int [preedit_changed](#)
- int [cursor_pos](#)
- int [cursor_pos_changed](#)
- [MPlist](#) * [candidate_list](#)
- int [candidate_index](#)
- int [candidate_from](#)
- int [candidate_to](#)
- int [candidate_show](#)
- int [candidates_changed](#)
- [MPlist](#) * [plist](#)

3.36.1 詳解

入力コンテキスト用構造体.

See struct [MInputContext](#)

[MInputContext](#) は、入力コンテキストオブジェクト用の構造体の型である。

3.36.2 フィールド詳解

3.36.2.1 im

[MInputMethod](#)* [MInputContext::im](#)

入力メソッドへの逆ポインタ。関数 [minput_create_ic\(\)](#) によって設定される。

3.36.2.2 produced

```
MText* MInputContext::produced
```

入力メソッドによって生成される M-text。関数 `minput_filter()` によって設定される。

3.36.2.3 arg

```
void* MInputContext::arg
```

関数 `minput_create_ic()` に渡される引数。

3.36.2.4 active

```
int MInputContext::active
```

入力コンテキストがアクティブかどうかを示すフラグ。入力コンテキストが生成された時点では値は 1（アクティブ）であり、関数 `minput_toggle()` によってトグルされる。

3.36.2.5 x

```
int MInputContext::x
```

スポットの X, Y 座標。

3.36.2.6 y

```
int MInputContext::y
```

3.36.2.7 ascent

```
int MInputContext::ascent
```

スポットのアセントとディセントのピクセル数。

3.36.2.8 descent

```
int MInputContext::descent
```


3.36.2.9 fontsize

```
int MInputContext::fontsize
```

preedit テキスト用のフォントサイズ (1/10 ポイント単位).

3.36.2.10 mt

```
MText* MInputContext::mt
```

スポット上の M-text、または NULL.

3.36.2.11 pos

```
int MInputContext::pos
```

<mt> におけるスポットの文字位置.

3.36.2.12

```
struct { ... } MInputContext::spot
```

入力コンテキストのスポットの位置と大きさ.

3.36.2.13 info

```
void* MInputContext::info
```

以下のメンバの使用法は入力メソッドドライバによって異なる。以下の説明は、内部入力メソッド用の入力ドライバに対するものである。これらは関数 <im>->driver.filter() によって設定される。

<im>->driver.create_ic() が設定する追加情報へのポインタ。入力コンテキストの内部状態を記録するために用いられる。

3.36.2.14 status

```
MText* MInputContext::status
```

入力コンテキストの現在の状態を表す M-text

3.36.2.15 status_changed

```
int MInputContext::status_changed
```

関数 <im>->driver.filter() は、<status> を変えた際にこの値を 1 に設定する。

3.36.2.16 preedit

```
MText* MInputContext::preedit
```

現在の preedit テキストを含む M-text。関数 <im>->driver.filter() によって設定される。

3.36.2.17 preedit_changed

```
int MInputContext::preedit_changed
```

関数 <im>->driver.filter() は、<preedit> を変えた際にこの値を 1 に設定する。

3.36.2.18 cursor_pos

```
int MInputContext::cursor_pos
```

<preedit> のカーソル位置

3.36.2.19 cursor_pos_changed

```
int MInputContext::cursor_pos_changed
```

関数 <im>->driver.filter() は、<cursor_pos> を変えた際にこの値を 1 に設定する。

3.36.2.20 candidate_list

```
MPlist* MInputContext::candidate_list
```

現在の候補グループの Plist。各要素は M-text か plist である。要素が M-text の場合（キーが Mtext である場合）には、そのグループの候補はその M-text 中の各文字である。要素が plist の場合（キーが Mplist である場合）には、そのリストの各要素は M-text であり、それらがそのグループの候補となる。

3.36.2.21 candidate_index

```
int MInputContext::candidate_index
```

現在選択されている候補が全候補中で何番目かを示すインデックス。最初の候補のインデックスは 0。最初の候補グループに七つの候補が含まれており、この値が 8 ならば、現在の候補は二番目の候補グループの二番目の要素ということになる。

3.36.2.22 candidate_from

```
int MInputContext::candidate_from
```

preedit テキスト中で、<candidate_list> に対応する最初と最後の位置。

3.36.2.23 candidate_to

```
int MInputContext::candidate_to
```

3.36.2.24 candidate_show

```
int MInputContext::candidate_show
```

現在の候補グループを表示するかどうかを示すフラグ。関数 `<im>->driver.filter()` は、入力メソッドが候補の表示を要求した時この値を 1 に、それ以外の時 0 に設定する。

3.36.2.25 candidates_changed

```
int MInputContext::candidates_changed
```

関数 `<im>->driver.filter()` は、上記のメンバ `<candidate_XXX>` の 1 つでも変更した際には、この値を enum `MInputCandidatesChanged` のビット単位での論理 OR に設定する。そうでなければ 0 に設定する。

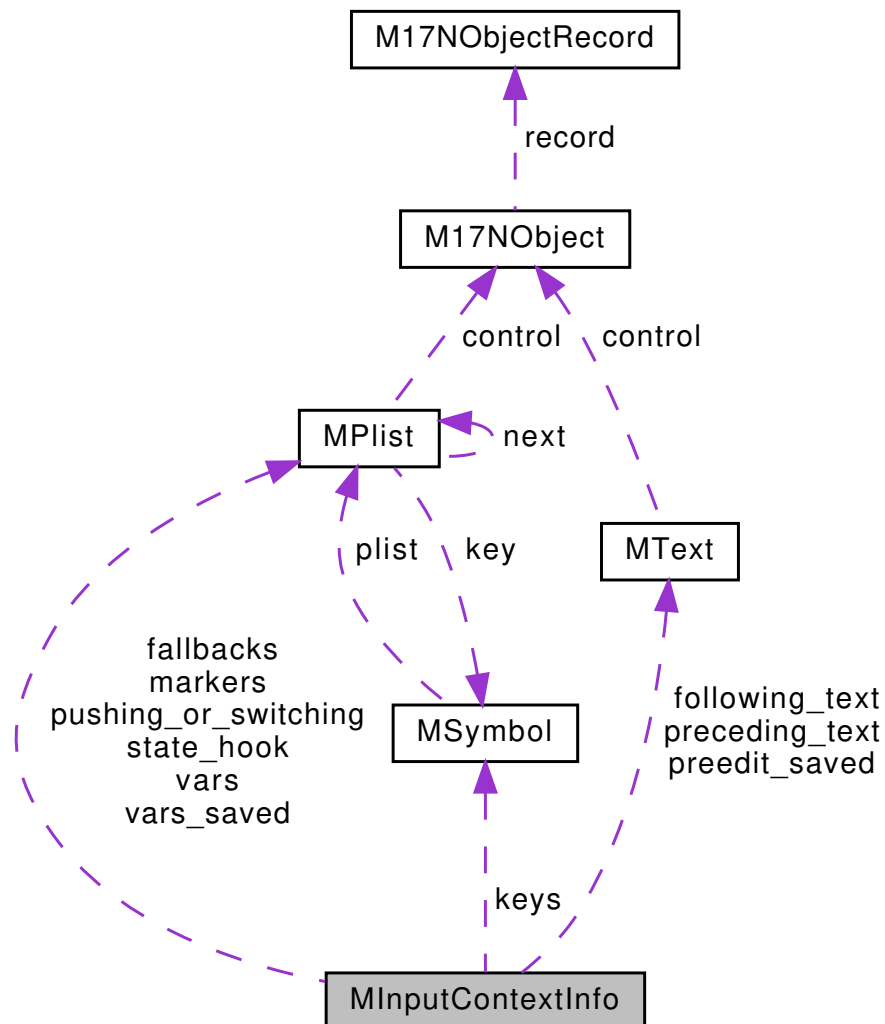
3.36.2.26 plist

```
MList* MInputContext::plist
```

`<im>->driver` の関数群によって自由に使用できる `plist`。内部入力メソッド用ドライバはこれをコールバック関数との引数や返値の受渡しに使用する。関数 `<im>->driver.create_ic()` はこの `plist` を空に設定する。関数 `<im>->driver.destroy_ic()` は `m17n_object_unref()` を用いてこの `plist` を解放する。

3.37 MInputContextInfo 構造体

MInputContextInfo 連携図



フィールド

- MIMState * state
- MIMState * prev_state
- MIMMap * map
- int size
- int inc
- int used
- MSymbol * keys
- int state_key_head
- int key_head
- int commit_key_head
- MText * preedit_saved
- int state_pos

- `MList * markers`
- `MList * vars`
- `MList * vars_saved`
- `MText * preceding_text`
- `MText * following_text`
- `int key_unhandled`
- `void * win_info`
- `MList * state_hook`
- `unsigned long tick`
- `MList * pushing_or_switching`
- `MList * fallbacks`
- `MIMInputStack * stack`

3.37.1 フィールド詳解

3.37.1.1 state

`MIMState*` `MInputContextInfo::state`

3.37.1.2 prev_state

`MIMState*` `MInputContextInfo::prev_state`

3.37.1.3 map

`MIMMap*` `MInputContextInfo::map`

3.37.1.4 size

`int` `MInputContextInfo::size`

3.37.1.5 inc

`int` `MInputContextInfo::inc`

3.37.1.6 used

```
int MInputContextInfo::used
```

3.37.1.7 keys

```
MSymbol* MInputContextInfo::keys
```

3.37.1.8 state_key_head

```
int MInputContextInfo::state_key_head
```

3.37.1.9 key_head

```
int MInputContextInfo::key_head
```

3.37.1.10 commit_key_head

```
int MInputContextInfo::commit_key_head
```

3.37.1.11 preedit_saved

```
MText* MInputContextInfo::preedit_saved
```

3.37.1.12 state_pos

```
int MInputContextInfo::state_pos
```

3.37.1.13 markers

```
MPlist* MInputContextInfo::markers
```

3.37.1.14 vars

```
MPlist* MInputContextInfo::vars
```

3.37.1.15 vars_saved

```
MPlist* MInputContextInfo::vars_saved
```

3.37.1.16 preceding_text

```
MText* MInputContextInfo::preceding_text
```

3.37.1.17 following_text

```
MText * MInputContextInfo::following_text
```

3.37.1.18 key_unhandled

```
int MInputContextInfo::key_unhandled
```

3.37.1.19 win_info

```
void* MInputContextInfo::win_info
```

3.37.1.20 state_hook

```
MPLIST* MInputContextInfo::state_hook
```

3.37.1.21 tick

```
unsigned long MInputContextInfo::tick
```

3.37.1.22 pushing_or_switching

```
MPLIST* MInputContextInfo::pushing_or_switching
```

3.37.1.23 fallbacks

```
MPLIST* MInputContextInfo::fallbacks
```

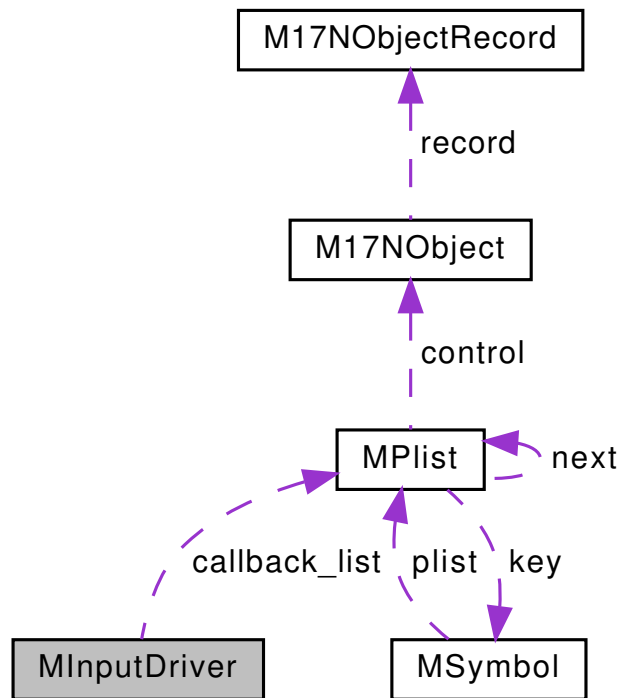
3.37.1.24 stack

```
MIMInputStack* MInputContextInfo::stack
```

3.38 MInputDriver 構造体

入力ドライバ用構造体.

MInputDriver 連携図



フィールド

- `int(* open_im)(MInputMethod *im)`
入力メソッドをオープンする。
- `void(* close_im)(MInputMethod *im)`
入力メソッドをクローズする。
- `int(* create_ic)(MInputContext *ic)`
入力コンテキストを生成する。
- `void(* destroy_ic)(MInputContext *ic)`
入力コンテキストを破壊する。
- `int(* filter)(MInputContext *ic, MSymbol key, void *arg)`
入力キーをフィルタする。
- `int(* lookup)(MInputContext *ic, MSymbol key, void *arg, MText *mt)`
入力コンテキストで生成されるテキストの獲得。
- `MPList * callback_list`
コールバック関数のリスト。

3.38.1 詳解

入力ドライバ用構造体。

`MInputDriver` は、入力メソッドを取り扱う関数を含む入力メソッドドライバの構造体の型である。

3.38.2 フィールド詳解

3.38.2.1 open_im

```
int(* MInputDriver::open_im) (MInputMethod *im)
```

入力メソッドをオープンする.

この関数は、入力メソッド `im` をオープンする。`im` の `<info>` 以外の全メンバーがセットされた後で、関数 `minput_open_im()` から呼ばれる。`im` をオープンできれば `0` を、できなければ `-1` を返す。この関数は `im->info` を設定して、他のドライバ関数から参照される情報を保持することができる。

3.38.2.2 close_im

```
void(* MInputDriver::close_im) (MInputMethod *im)
```

入力メソッドをクローズする.

この関数は、入力メソッド `im` をクローズする。関数 `minput_close_im()` から呼ばれる。入力メソッドのクローズがすべて終了した時点で、この関数は `im->info` に割り当てられているメモリを (あれば) すべて開放する。ただし、`im` の他のメンバに影響を与えてはならない。

3.38.2.3 create_ic

```
int(* MInputDriver::create_ic) (MInputContext *ic)
```

入力コンテキストを生成する.

この関数は入力コンテキスト `ic` を生成する。`ic` の `<info>` 以外の全メンバーがセットされた後で、関数 `minput_create_ic()` から呼ばれる。`ic` を生成できれば `0` を、できなければ `-1` を返す。この関数は `ic->info` を設定して、他のドライバ関数から参照される情報を保持することができる。

3.38.2.4 destroy_ic

```
void(* MInputDriver::destroy_ic) (MInputContext *ic)
```

入力コンテキストを破壊する.

関数 `minput_destroy_ic()` から呼ばれ、入力コンテキスト `ic` を破壊する。入力コンテキストの破壊がすべて終了した時点で、`ic->info` に割り当てられているメモリを (あれば) すべて開放する。ただし、`ic` の他のメンバに影響を与えてはならない。

3.38.2.5 filter

```
int(* MInputDriver::filter) (MInputContext *ic, MSymbol key, void *arg)
```

入力キーをフィルタする.

関数 `minput_filter()` から呼ばれ、入力キーをフィルタする。引数 `key`, `arg` は関数 `minput_filter()` のものと同じ。

この関数は `key` を処理し、`ic` の内部状態を更新する。`key` が入力メソッドに吸収されてテキストが生成されなかった場合には、1 を返す。そうでなければ 0 を返す。

メンバ `<callback>` に必要であれば、`ic->status`, `ic->preedit`, `ic->cursor_pos`, `ic->ncandidates`, `ic->candidates`, `ic->produced` を更新できる。

`arg` の意味は入力メソッドドライバに依存する。例は `minput_default_driver` または `minput_gui_driver` の説明を参照のこと。

3.38.2.6 lookup

```
int(* MInputDriver::lookup) (MInputContext *ic, MSymbol key, void *arg, MText *mt)
```

入力コンテキストで生成されるテキストの獲得.

関数 `minput_lookup()` から呼ばれ、入力コンテキスト `ic` で生成されるテキストを検索する。入力キー `key` によって生成されるテキストがあれば、`M-text mt` に追加する。`key` が入力メソッド `ic` によって正しく処理されれば 0 を返す。そうでなければ 1 を返す。

`arg` の意味は入力メソッドドライバに依存する。例は `minput_default_driver` または `minput_gui_driver` の説明を参照のこと。

3.38.2.7 callbackList

```
MPList* MInputDriver::callback_list
```

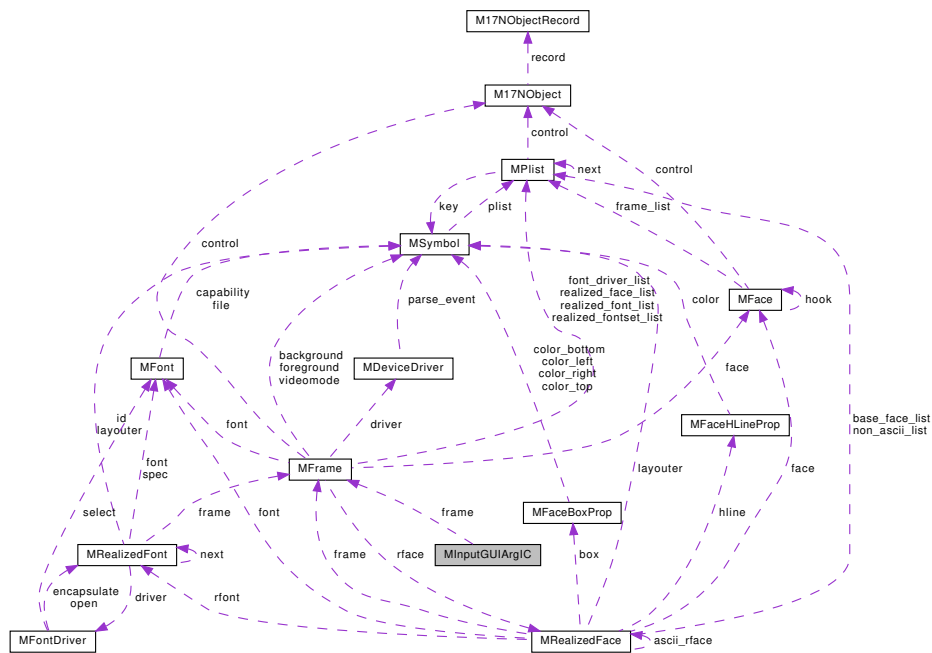
コールバック関数のリスト.

コールバック関数のリスト。キーは次のいずれか。 `Minput_preedit_start`, `Minput_preedit_draw`, `Minput_preedit_done`, `Minput_status_start`, `Minput_status_draw`, `Minput_status_done`, `Minput_candidates_start`, `Minput_candidates_draw`, `Minput_candidates_done`, `Minput_set_spot`, `Minput_toggle`, `Minput_reset`, `Minput_get_surrounding_text`, `Minput_delete_surrounding_text`。値は `::MInputCallbackFunc` 型の関数。

3.39 MInputGUIArgIC 構造体

関数 `minput_create_ic()` の引数の型宣言.

MInputGUIArgIC 連携図



フィールド

- `MFrame * frame`
- `MDrawWindow client`
- `MDrawWindow focus`

3.39.1 詳解

関数 `minput_create_ic()` の引数の型宣言.

`MInputGUIArgIC` は、関数 `minput_create_ic()` が内部入力メソッドの入力コンテキストを生成する際の、引数 `arg` 用の型である。

3.39.2 フィールド詳解

3.39.2.1 frame

`MFrame* MInputGUIArgIC::frame`

クライアントのフレーム

3.39.2.2 client

```
MDrawWindow MInputGUIArgIC::client
```

preedit テキストと status テキストを表示するウィンドウ

3.39.2.3 focus

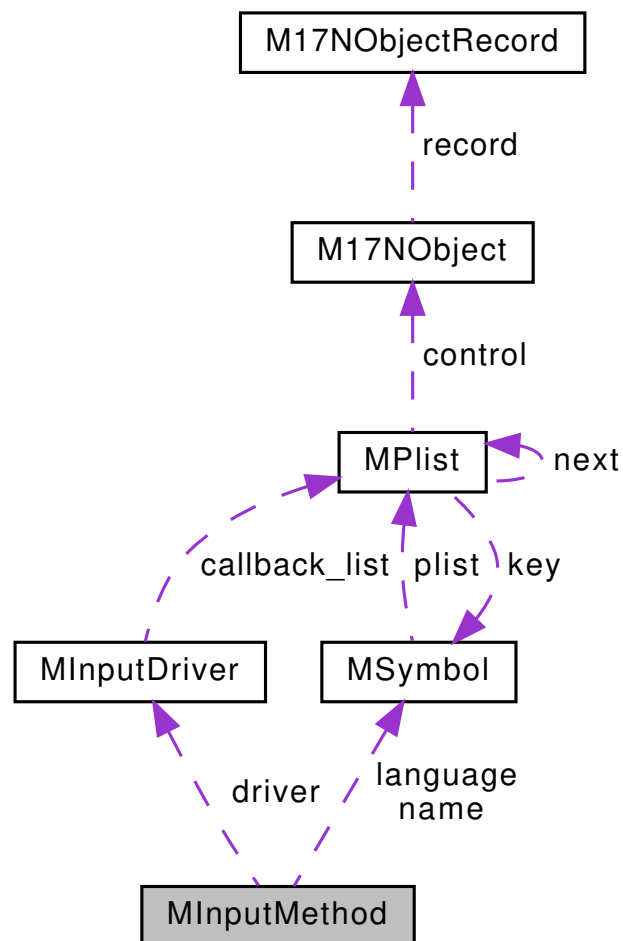
```
MDrawWindow MInputGUIArgIC::focus
```

入力コンテキストがフォーカスをおいているウィンドウ

3.40 MInputMethod 構造体

入力メソッドの構造体.

MInputMethod 連携図



フィールド

- [MSymbol language](#)
- [MSymbol name](#)
- [MInputDriver driver](#)
- void * [arg](#)
- void * [info](#)

3.40.1 詳解

入力メソッドの構造体.

See struct [MInputMethod](#)

[MInputMethod](#) は、入力メソッドオブジェクト用の構造体の型である。

3.40.2 フィールド詳解

3.40.2.1 language

[MSymbol](#) MInputMethod::language

どの言語用の入力メソッドか。入力メソッドが外部のものである場合の値は `Mnil`。

3.40.2.2 name

[MSymbol](#) MInputMethod::name

入力メソッドの名前。外部メソッドである場合には、`Minput_driver` をキーとするプロパティを持ち、その値は適切な入力メソッドドライバへのポインタでなくてはならない。

3.40.2.3 driver

[MInputDriver](#) MInputMethod::driver

その入力メソッド用の入力メソッドドライバ。

3.40.2.4 arg

void* MInputMethod::arg

[minput_open_jm\(\)](#) に渡される引数。

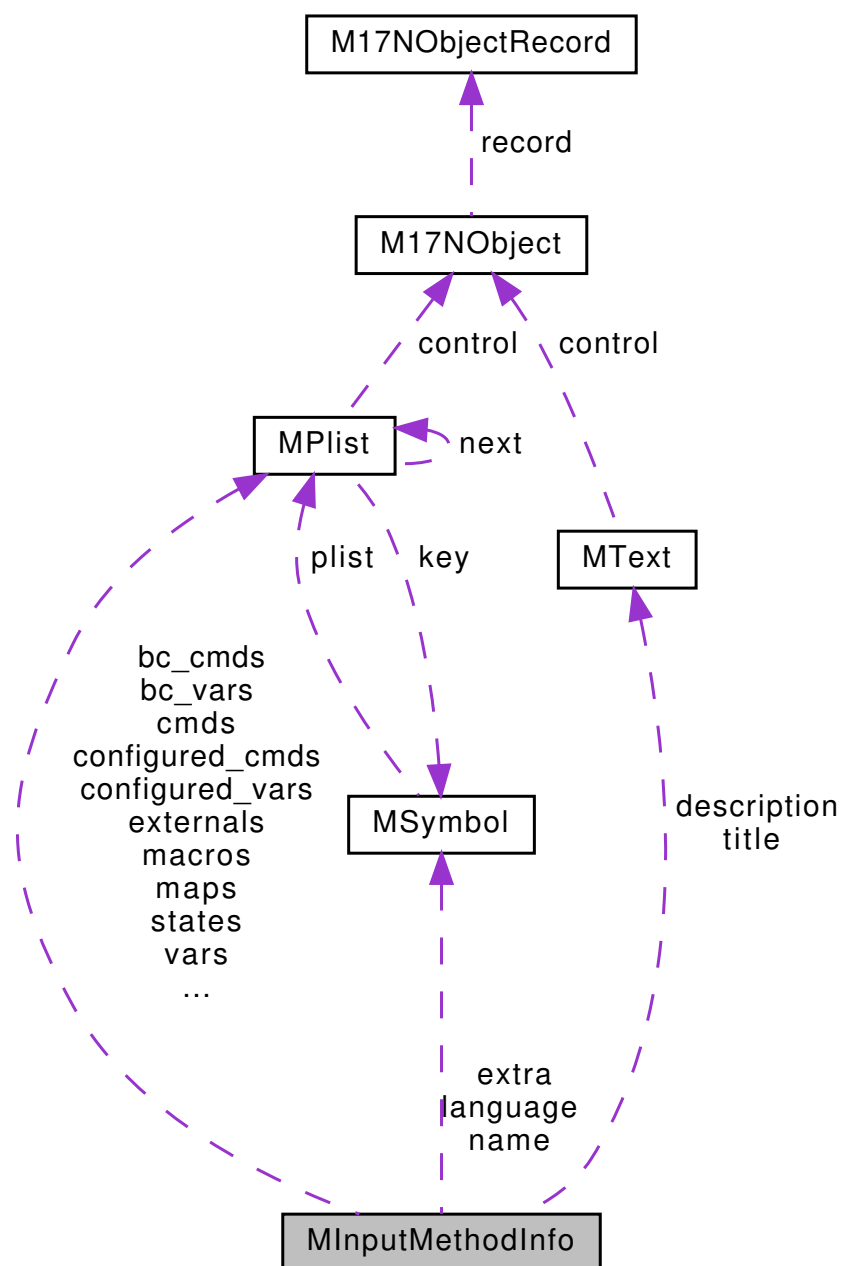
3.40.2.5 info

```
void* MInputMethod::info
```

<driver>.open_jm() が設定する追加情報へのポインタ。

3.41 MInputMethodInfo 構造体

MInputMethodInfo 連携図



フィールド

- [MDatabase](#) * [mdb](#)
- [MSymbol](#) [language](#)
- [MSymbol](#) [name](#)
- [MSymbol](#) [extra](#)
- [MPlist](#) * [cmds](#)
- [MPlist](#) * [configured_cmds](#)
- [MPlist](#) * [bc_cmds](#)
- [MPlist](#) * [vars](#)
- [MPlist](#) * [configured_vars](#)
- [MPlist](#) * [bc_vars](#)
- [MText](#) * [description](#)
- [MText](#) * [title](#)
- [MPlist](#) * [maps](#)
- [MPlist](#) * [states](#)
- [MPlist](#) * [macros](#)
- [MPlist](#) * [externals](#)
- unsigned long [tick](#)

3.41.1 フィールド詳解

3.41.1.1 mdb

[MDatabase](#)* [MInputMethodInfo::mdb](#)

3.41.1.2 language

[MSymbol](#) [MInputMethodInfo::language](#)

3.41.1.3 name

[MSymbol](#) [MInputMethodInfo::name](#)

3.41.1.4 extra

[MSymbol](#) [MInputMethodInfo::extra](#)

3.41.1.5 cmds

```
MPlist* MInputMethodInfo::cmds
```

3.41.1.6 configured_cmds

```
MPlist * MInputMethodInfo::configured_cmds
```

3.41.1.7 bc_cmds

```
MPlist * MInputMethodInfo::bc_cmds
```

3.41.1.8 vars

```
MPlist* MInputMethodInfo::vars
```

3.41.1.9 configured_vars

```
MPlist * MInputMethodInfo::configured_vars
```

3.41.1.10 bc_vars

```
MPlist * MInputMethodInfo::bc_vars
```

3.41.1.11 description

```
MText* MInputMethodInfo::description
```

3.41.1.12 title

```
MText* MInputMethodInfo::title
```

3.41.1.13 maps

```
MList* MInputMethodInfo::maps
```

3.41.1.14 states

```
MList* MInputMethodInfo::states
```

3.41.1.15 macros

```
MList* MInputMethodInfo::macros
```

3.41.1.16 externals

```
MList* MInputMethodInfo::externals
```

3.41.1.17 tick

```
unsigned long MInputMethodInfo::tick
```

3.42 MInputXIMArgIC 構造体

関数 `minput_create_ic()` の引数 `arg` によって指される構造体.

フィールド

- XIMStyle `input_style`
- Window `client_win`
- Window `focus_win`
- XVaNestedList `preedit_attrs`
- XVaNestedList `status_attrs`

3.42.1 詳解

関数 `minput_create_ic()` の引数 `arg` によって指される構造体.

`MInputXIMArgIC` 型は、関数 `minput_create_ic()` が名前 `Mxim` を持つ外部入力メソッド用に呼ばれる際に、引数 `arg` によって指される構造体である。

3.42.2 フィールド詳解

3.42.2.1 `input_style`

```
XIMStyle MInputXIMArgIC::input_style
```

`XCreateIC` の `XNInputStyle` に続く引数として用いられる。ゼロならば、(`XIMPreeditNothing` | `XIMStatusNothing`) が用いられ、`<preedit_attrs>` と `<status_attrs>` は `NULL` に設定される。

3.42.2.2 `client_win`

```
Window MInputXIMArgIC::client_win
```

`XCreateIC` の `XNClientWindow` に続く引数として用いられる。

3.42.2.3 `focus_win`

```
Window MInputXIMArgIC::focus_win
```

`XCreateIC` の `XNFocusWindow` に続く引数として用いられる。

3.42.2.4 `preedit_attrs`

```
XVaNestedList MInputXIMArgIC::preedit_attrs
```

`NULL` でなければ、`XCreateIC following` の `XNPreeditAttributes` に続く引数として用いられる。

3.42.2.5 `status_attrs`

```
XVaNestedList MInputXIMArgIC::status_attrs
```

`NULL` でなければ、`XCreateIC following` の `XNStatusAttributes` に続く引数として用いられる。

3.43 MInputXIMArgIM 構造体

関数 `minput_open_im()` の引数 `arg` によって指される構造体.

フィールド

- `Display *` `display`
- `XrmDatabase` `db`
- `char *` `res_class`
- `char *` `res_name`
- `char *` `locale`
- `char *` `modifier_list`

3.43.1 詳解

関数 `minput_open_im()` の引数 `arg` によって指される構造体.

`MInputXIMArgIM` 型は、関数 `minput_open_im()` が名前 `Mxim` を持つ外部入力メソッドを生成する際に引数 `arg` によって指される構造体である。

3.43.2 フィールド詳解

3.43.2.1 display

```
Display* MInputXIMArgIM::display
```

以下の4つのメンバの意味は、`XOpenIM()` の引数の意味と同じである.

クライアントのディスプレイ.

3.43.2.2 db

```
XrmDatabase MInputXIMArgIM::db
```

X リソース・データベースへのポインタ.

3.43.2.3 res_class

```
char* MInputXIMArgIM::res_class
```

アプリケーションの完全なクラス名.

3.43.2.4 res_name

```
char* MInputXIMArgIM::res_name
```

アプリケーションの完全なリソース名.

3.43.2.5 locale

```
char* MInputXIMArgIM::locale
```

XIM がオープンされたロケール名.

3.43.2.6 modifier_list

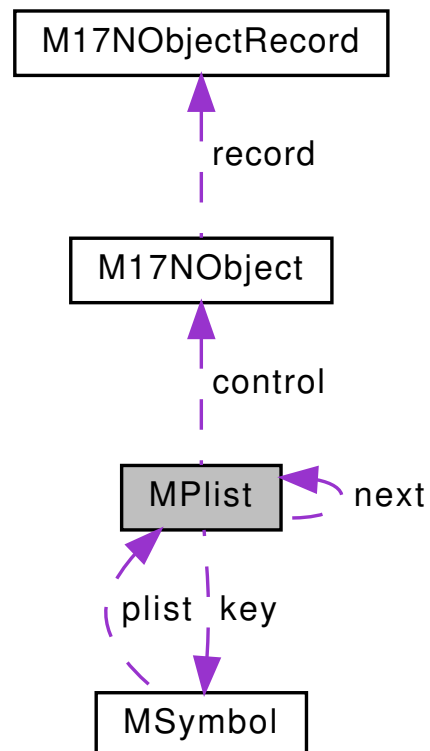
```
char* MInputXIMArgIM::modifier_list
```

XSetLocaleModifiers() の引数.

3.44 MPlist 構造体

プロパティリスト・オブジェクトの型宣言.

MPlist 連携図



フィールド

- [M17NObject control](#)
- [MSymbol key](#)
- `union {`
 - `void * pointer`
 - `M17NFunc func`
- `} val`
- `MPlist * next`

3.44.1 詳解

プロパティリスト・オブジェクトの型宣言.

[MPlist](#) はプロパティリスト (Property list) オブジェクトの型である。内部構造はアプリケーションプログラムからは見えない。

3.44.2 フィールド詳解

3.44.2.1 control

[M17NObject](#) `MPlist::control`

3.44.2.2 key

[MSymbol](#) `MPlist::key`

3.44.2.3 pointer

`void* MPlist::pointer`

3.44.2.4 func

[M17NFunc](#) `MPlist::func`

3.44.2.5

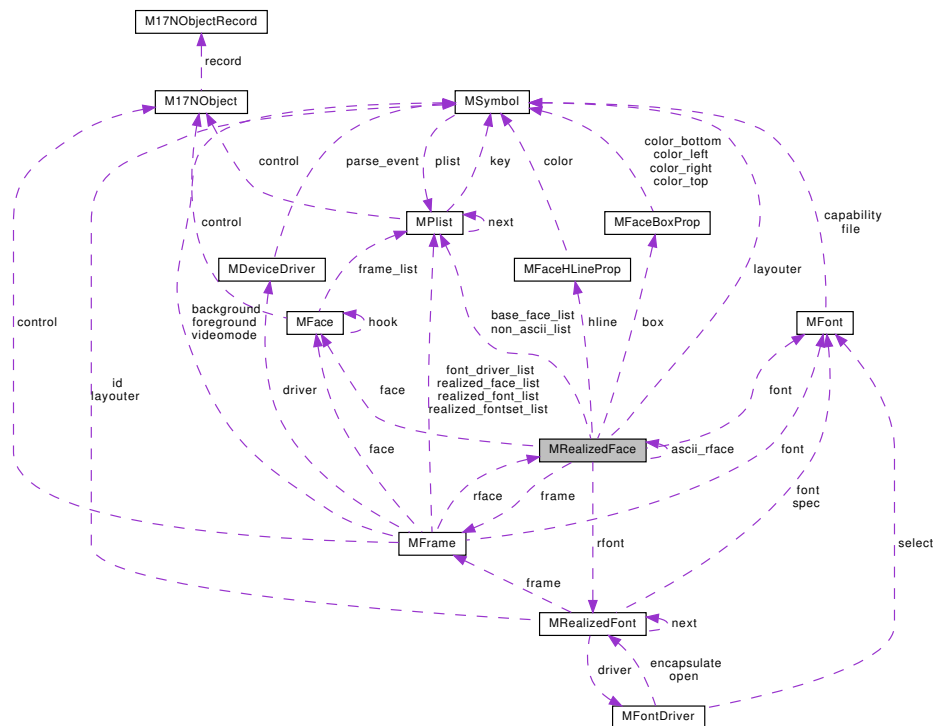
```
union { ... } MPlist::val
```

3.44.2.6 next

```
MPlist* MPlist::next
```

3.45 MRealizedFace 構造体

MRealizedFace 連携図



フィールド

- MFrame * frame
- MFace face
- MFont * font
- MPlist * base_face_list
- MRealizedFont * rfont
- MRealizedFontset * rfontset
- MSymbol layout
- MFaceHLineProp * hline

- `MFaceBoxProp` * `box`
- `MRealizedFace` * `ascii_rface`
- `MPlist` * `non_ascii_list`
- `int` `ascent`
- `int` `descent`
- `int` `space_width`
- `int` `average_width`
- `void` * `info`

3.45.1 フィールド詳解

3.45.1.1 frame

```
MFrame* MRealizedFace::frame
```

3.45.1.2 face

```
MFace MRealizedFace::face
```

3.45.1.3 font

```
MFont* MRealizedFace::font
```

3.45.1.4 base_face_list

```
MPlist* MRealizedFace::base_face_list
```

3.45.1.5 rfont

```
MRealizedFont* MRealizedFace::rfont
```


3.45.1.6 rfontset

```
MRealizedFontset* MRealizedFace::rfontset
```

3.45.1.7 layouter

```
MSymbol MRealizedFace::layouter
```

3.45.1.8 hline

```
MFaceHLineProp* MRealizedFace::hline
```

3.45.1.9 box

```
MFaceBoxProp* MRealizedFace::box
```

3.45.1.10 ascii_rface

```
MRealizedFace* MRealizedFace::ascii_rface
```

3.45.1.11 non_ascii_list

```
MPlist* MRealizedFace::non_ascii_list
```

3.45.1.12 ascent

```
int MRealizedFace::ascent
```

3.45.1.13 descent

```
int MRealizedFace::descent
```

3.45.1.14 space_width

```
int MRealizedFace::space_width
```

3.45.1.15 average_width

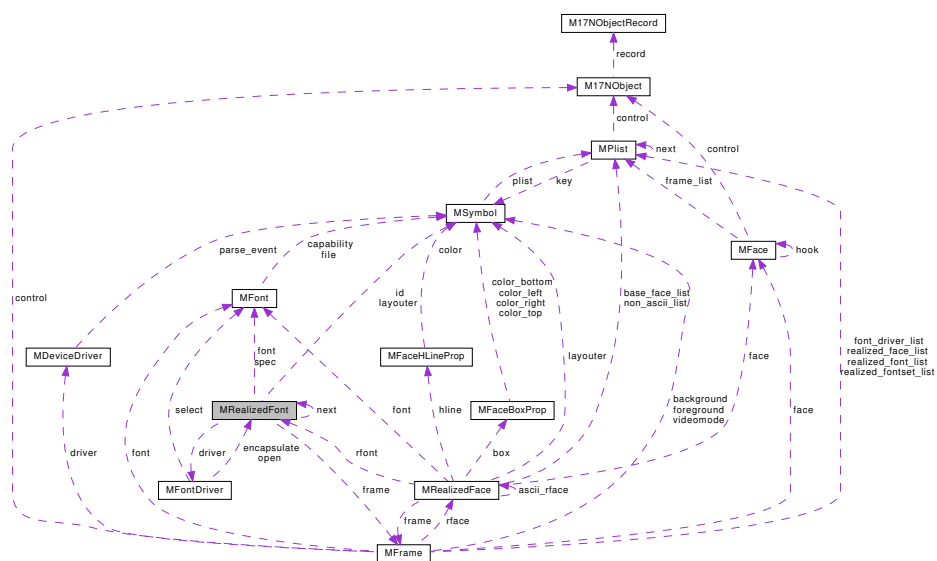
```
int MRealizedFace::average_width
```

3.45.1.16 info

```
void* MRealizedFace::info
```

3.46 MRealizedFont 構造体

MRealizedFont 連携図



フィールド

- [MFont spec](#)
- [MSymbol id](#)
- [MFrame * frame](#)
- [MFont * font](#)
- [MFontDriver * driver](#)
- [MSymbol layouter](#)
- [int encapsulating](#)
- [void * info](#)
- [int x_ppem](#)
- [int y_ppem](#)
- [int ascent](#)
- [int descent](#)
- [int max_advance](#)
- [int average_width](#)
- [int baseline_offset](#)
- [void * fontp](#)
- [MRealizedFont * next](#)

3.46.1 フィールド詳解

3.46.1.1 spec

[MFont](#) MRealizedFont::spec

3.46.1.2 id

[MSymbol](#) MRealizedFont::id

3.46.1.3 frame

[MFrame*](#) MRealizedFont::frame

3.46.1.4 font

[MFont*](#) MRealizedFont::font

3.46.1.5 driver

```
MFontDriver* MRealizedFont::driver
```

3.46.1.6 layouter

```
MSymbol MRealizedFont::layouter
```

3.46.1.7 encapsulating

```
int MRealizedFont::encapsulating
```

3.46.1.8 info

```
void* MRealizedFont::info
```

3.46.1.9 x_ppem

```
int MRealizedFont::x_ppem
```

3.46.1.10 y_ppem

```
int MRealizedFont::y_ppem
```

3.46.1.11 ascent

```
int MRealizedFont::ascent
```

3.46.1.12 descent

```
int MRealizedFont::descent
```

3.46.1.13 max_advance

```
int MRealizedFont::max_advance
```

3.46.1.14 average_width

```
int MRealizedFont::average_width
```

3.46.1.15 baseline_offset

```
int MRealizedFont::baseline_offset
```

3.46.1.16 fontp

```
void* MRealizedFont::fontp
```

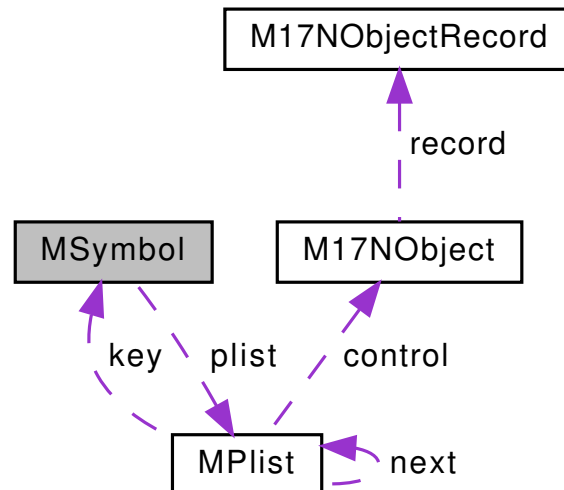
3.46.1.17 next

```
MRealizedFont* MRealizedFont::next
```

3.47 MSymbol 構造体

シンボルの型宣言.

MSymbol 連携図



フィールド

- unsigned **managing_key**: 1
- char * **name**
- int **length**
- **MPList** **plist**
- struct MSymbolStruct * **next**

3.47.1 詳解

シンボルの型宣言.

MSymbol はシンボル (symbol) オブジェクトの型である。内部構造はアプリケーションプログラムからは見えない。

3.47.2 フィールド詳解

3.47.2.1 managing_key

```
unsigned MSymbol::managing_key
```

3.47.2.2 name

```
char* MSymbol::name
```

3.47.2.3 length

```
int MSymbol::length
```

3.47.2.4 plist

```
MPList MSymbol::plist
```

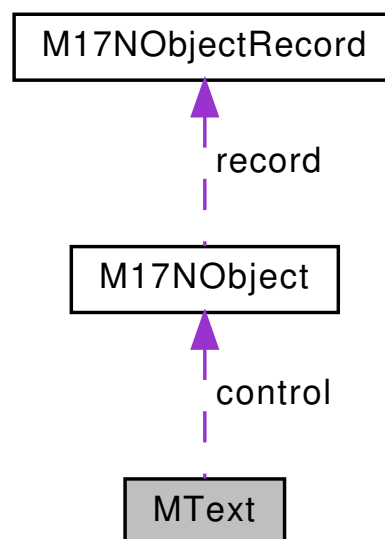
3.47.2.5 next

```
struct MSymbolStruct* MSymbol::next
```

3.48 MText 構造体

MText の型宣言.

MText 連携図



フィールド

- [M17NObject control](#)
- unsigned [format](#): 16
- unsigned [coverage](#): 16
- int [nchars](#)
- int [nbytes](#)
- unsigned char * [data](#)
- int [allocated](#)
- struct MTextPlist * [plist](#)
- int [cache_char_pos](#)
- int [cache_byte_pos](#)

3.48.1 詳解

[MText](#) の型宣言.

[Mtext](#) は M-text オブジェクトの型である。内部構造はアプリケーションプログラムからは見えない。

3.48.2 フィールド詳解

3.48.2.1 control

[M17NObject](#) MText::control

3.48.2.2 format

unsigned MText::format

3.48.2.3 coverage

unsigned MText::coverage

3.48.2.4 nchars

int MText::nchars

3.48.2.5 nbytes

```
int MText::nbytes
```

3.48.2.6 data

```
unsigned char* MText::data
```

3.48.2.7 allocated

```
int MText::allocated
```

3.48.2.8 plist

```
struct MTextPlist* MText::plist
```

3.48.2.9 cache_char_pos

```
int MText::cache_char_pos
```

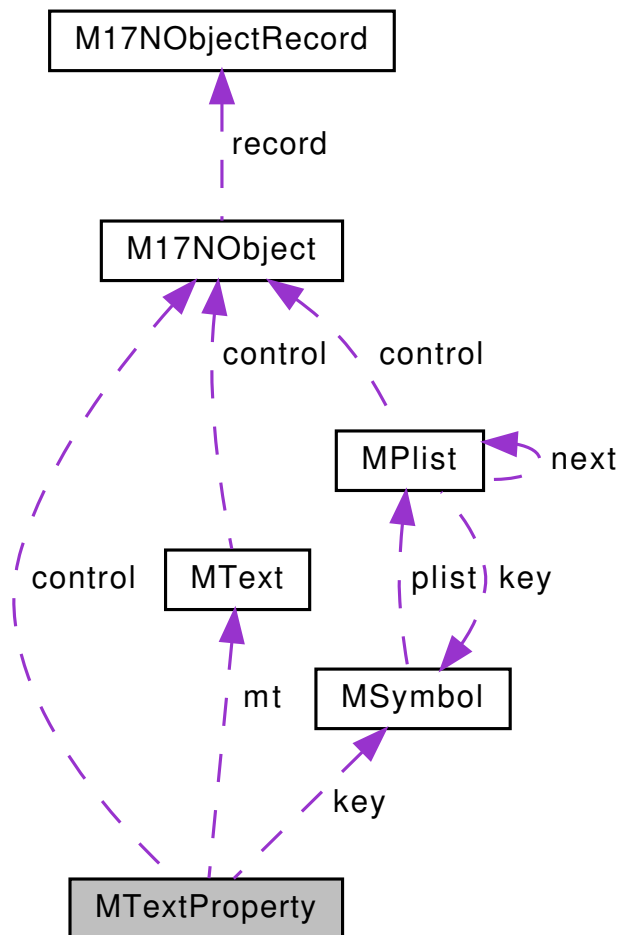
3.48.2.10 cache_byte_pos

```
int MText::cache_byte_pos
```

3.49 MTextProperty 構造体

テキストプロパティの型宣言.

MTextProperty 連携図



フィールド

- **M17NObject** control
- unsigned **attach_count**
- **MText** * **mt**
- int **start**
- int **end**
- **MSymbol** key
- void * **val**

3.49.1 詳解

テキストプロパティの型宣言.

MTextProperty は テキストプロパティ オブジェクトの型である。内部構造はアプリケーションプログラムからは見えない。

3.49.2 フィールド詳解

3.49.2.1 control

`M17NObject` MTextProperty::control

3.49.2.2 attach_count

`unsigned` MTextProperty::attach_count

3.49.2.3 mt

`MText*` MTextProperty::mt

3.49.2.4 start

`int` MTextProperty::start

3.49.2.5 end

`int` MTextProperty::end

3.49.2.6 key

`MSymbol` MTextProperty::key

3.49.2.7 val

`void*` MTextProperty::val

Chapter 4

ファイル詳解

4.1 character.c ファイル

マクロ定義

- `#define MCHAR_MAX`
文字コードの最大値.

関数

- `MSymbol mchar_define_property` (const char *name, `MSymbol` type)
文字プロパティを定義する.
- `void * mchar_get_prop` (int c, `MSymbol` key)
文字プロパティの値を得る.
- `int mchar_put_prop` (int c, `MSymbol` key, void *val)
文字プロパティの値を設定する.
- `MCharTable * mchar_get_prop_table` (`MSymbol` key, `MSymbol` *type)
文字プロパティの文字テーブルを得る.

変数

変数: 文字プロパティのキー

これらのシンボルは文字プロパティのキーとして使われる。

- `MSymbol Mscript`
スクリプトを表わすキー.
- `MSymbol Mname`
名前を表わすキー.
- `MSymbol Mcategory`
一般カテゴリを表わすキー.
- `MSymbol Mcombining_class`
標準結合クラスを表わすキー.
- `MSymbol Mbidi_category`

構築: Doxygen

- 双方向カテゴリを表わすキー.
- [MSymbol Msimple_case_folding](#)
対応する小文字一文字を表わすキー.
- [MSymbol Mcomplicated_case_folding](#)
対応する小文字の列を表わすキー.
- [MSymbol Mcased](#)
Case 処理に用いられる値のキー.
- [MSymbol Msoft_dotted](#)
Case 処理に用いられる値のキー.
- [MSymbol Mcase_mapping](#)
Case 処理に用いられる値のキー.
- [MSymbol Mblock](#)
スクリプトブロック名を表すキー.

4.2 character.h ファイル

マクロ定義

- [#define MAX_UTF8_CHAR_BYTES 6](#)
- [#define MAX_UNICODE_CHAR_BYTES 4](#)
- [#define USHORT_SIZE \(sizeof \(unsigned short\)\)](#)
- [#define UINT_SIZE \(sizeof \(unsigned int\)\)](#)
- [#define UNIT_BYTES\(format\)](#)
- [#define CHAR_UNITS_ASCII\(c\) \(\(c\) < 0x80\)](#)
- [#define CHAR_UNITS_UTF8\(c\)](#)
- [#define CHAR_UNITS_UTF16\(c\) \(\(c\) < 0x110000 ? \(2 - \(\(c\) < 0x10000\)\) : 0\)](#)
- [#define CHAR_UNITS_UTF32\(c\) 1](#)
- [#define CHAR_UNITS\(c, format\)](#)
- [#define CHAR_BYTES CHAR_UNITS_UTF8](#)
- [#define CHAR_UNITS_AT_UTF8\(p\)](#)
- [#define CHAR_UNITS_AT_UTF16\(p\)](#)
- [#define CHAR_UNITS_AT\(mt, p\)](#)
- [#define CHAR_BYTES_AT CHAR_UNITS_AT_UTF8](#)
- [#define CHAR_UNITS_BY_HEAD_UTF8\(c\)](#)
- [#define CHAR_UNITS_BY_HEAD_UTF16\(c\) \(2 - \(\(unsigned short\) \(c\) < 0xD800 || \(unsigned short\) \(c\) >= 0xDC00\)\)](#)
- [#define CHAR_UNITS_BY_HEAD\(c, format\)](#)
- [#define CHAR_BYTES_BY_HEAD CHAR_UNITS_BY_HEAD_UTF8](#)
- [#define STRING_CHAR_UTF8\(p\)](#)
- [#define STRING_CHAR_UTF16\(p\)](#)
- [#define STRING_CHAR STRING_CHAR_UTF8](#)
- [#define STRING_CHAR_ADVANCE_UTF8\(p\)](#)
- [#define STRING_CHAR_ADVANCE_UTF16\(p\)](#)
- [#define STRING_CHAR_ADVANCE STRING_CHAR_ADVANCE_UTF8](#)
- [#define STRING_CHAR_AND_UNITS_UTF8\(p, bytes\)](#)
- [#define STRING_CHAR_AND_UNITS_UTF16\(p, units\)](#)
- [#define STRING_CHAR_AND_UNITS\(p, units, format\)](#)
- [#define STRING_CHAR_AND_BYTES STRING_CHAR_AND_UNITS_UTF8](#)
- [#define CHAR_STRING_UTF8\(c, p\)](#)
- [#define CHAR_STRING_UTF16\(c, p\)](#)
- [#define CHAR_STRING CHAR_STRING_UTF8](#)

- #define `CHAR_HEAD_P_UTF8(p)` `((*(p) & 0xC0) != 0x80)`
- #define `CHAR_HEAD_P_UTF16(p)`
- #define `CHAR_HEAD_P` `CHAR_HEAD_P_UTF8`
- #define `TOLOWER(c)` `((c) >= 'A' && (c) <= 'Z') ? (c) + 32 : (c)`
- #define `TOUPPER(c)` `((c) >= 'a' && (c) <= 'z') ? (c) - 32 : (c)`
- #define `ISUPPER(c)` `((c) >= 'A' && (c) <= 'Z')`
- #define `ISALNUM(c)`

関数

- void `mchar_define_prop` (`MSymbol` key, `MSymbol` type, void *mdb)

4.2.1 マクロ定義詳解

4.2.1.1 MAX_UTF8_CHAR_BYTES

```
#define MAX_UTF8_CHAR_BYTES 6
```

4.2.1.2 MAX_UNICODE_CHAR_BYTES

```
#define MAX_UNICODE_CHAR_BYTES 4
```

4.2.1.3 USHORT_SIZE

```
#define USHORT_SIZE (sizeof (unsigned short))
```

4.2.1.4 UINT_SIZE

```
#define UINT_SIZE (sizeof (unsigned int))
```

4.2.1.5 UNIT_BYTES

```
#define UNIT_BYTES(  
    format )
```

値:

```
((format) <= MTEXT_FORMAT_UTF_8 ? 1      \  
 : ((format) <= MTEXT_FORMAT_UTF_16BE ? USHORT_SIZE  \  
 : UINT_SIZE)
```

4.2.1.6 CHAR_UNITS_ASCII

```
#define CHAR_UNITS_ASCII(  
    c ) ((c) < 0x80)
```

4.2.1.7 CHAR_UNITS_UTF8

```
#define CHAR_UNITS_UTF8(  
    c )
```

値:

```
((c) < 0x80 ? 1      \  
 : ((c) < 0x800 ? 2   \  
 : ((c) < 0x10000 ? 3  \  
 : ((c) < 0x200000 ? 4  \  
 : ((c) < 0x4000000 ? 5  \  
 : 6)
```

4.2.1.8 CHAR_UNITS_UTF16

```
#define CHAR_UNITS_UTF16(  
    c ) ((c) < 0x110000 ? (2 - ((c) < 0x10000)) : 0)
```

4.2.1.9 CHAR_UNITS_UTF32

```
#define CHAR_UNITS_UTF32(  
    c ) 1
```


4.2.1.10 CHAR_UNITS

```
#define CHAR_UNITS(
    c,
    format )
```

値:

```
((format) <= MTEXT_FORMAT_UTF_8 ? CHAR_UNITS_UTF8 (c) \
: (format) <= MTEXT_FORMAT_UTF_16BE ? CHAR_UNITS_UTF16 (c) \
: CHAR_UNITS_UTF32 (c))
```

4.2.1.11 CHAR_BYTES

```
#define CHAR_BYTES CHAR_UNITS_UTF8
```

4.2.1.12 CHAR_UNITS_AT_UTF8

```
#define CHAR_UNITS_AT_UTF8(
    p )
```

値:

```
(!(* (p) & 0x80) ? 1 \
: !(* (p) & 0x20) ? 2 \
: !(* (p) & 0x10) ? 3 \
: !(* (p) & 0x08) ? 4 \
: !(* (p) & 0x04) ? 5 \
: !(* (p) & 0x02) ? 6 \
: 0)
```

4.2.1.13 CHAR_UNITS_AT_UTF16

```
#define CHAR_UNITS_AT_UTF16(
    p )
```

値:

```
(2 - (*(unsigned short *) (p) < 0xD800 \
|| *(unsigned short *) (p) >= 0xDC00))
```

4.2.1.14 CHAR_UNITS_AT

```
#define CHAR_UNITS_AT(
    mt,
    p )
```

値:

```
((mt)->format <= MTEXT_FORMAT_UTF_8 ? CHAR_UNITS_AT_UTF8 (p) \
: (mt)->format <= MTEXT_FORMAT_UTF_16BE ? CHAR_UNITS_AT_UTF16 (p) \
: 1)
```

4.2.1.15 CHAR_BYTES_AT

```
#define CHAR_BYTES_AT CHAR_UNITS_AT_UTF8
```

4.2.1.16 CHAR_UNITS_BY_HEAD_UTF8

```
#define CHAR_UNITS_BY_HEAD_UTF8(  
    c )
```

値:

```
(!( (c) & 0x80) ? 1      \  
: !( (c) & 0x20) ? 2    \  
: !( (c) & 0x10) ? 3    \  
: !( (c) & 0x08) ? 4    \  
: !( (c) & 0x04) ? 5    \  
: !( (c) & 0x02) ? 6    \  
: 0)
```

4.2.1.17 CHAR_UNITS_BY_HEAD_UTF16

```
#define CHAR_UNITS_BY_HEAD_UTF16(  
    c )  (2 - ((unsigned short) (c) < 0xD800 || (unsigned short) (c) >= 0xDC00))
```

4.2.1.18 CHAR_UNITS_BY_HEAD

```
#define CHAR_UNITS_BY_HEAD(  
    c,  
    format )
```

値:

```
((format) <= MTEXT_FORMAT_UTF_8 ? CHAR_UNITS_BY_HEAD_UTF8 (c)  \  
: (format) <= MTEXT_FORMAT_UTF_16BE ? CHAR_UNITS_BY_HEAD_UTF16 (c)  \  
: 1)
```

4.2.1.19 CHAR_BYTES_BY_HEAD

```
#define CHAR_BYTES_BY_HEAD CHAR_UNITS_BY_HEAD_UTF8
```

4.2.1.20 STRING_CHAR_UTF8

```
#define STRING_CHAR_UTF8(
    p )
```

値:

```
(!(p)[0] & 0x80) ? (p)[0] \
: !((p)[0] & 0x20) ? (((p)[0] & 0x1F) << 6) \
| ((p)[1] & 0x3F)) \
: !((p)[0] & 0x10) ? (((p)[0] & 0x0F) << 12) \
| ((p)[1] & 0x3F) << 6) \
| ((p)[2] & 0x3F)) \
: !((p)[0] & 0x08) ? (((p)[0] & 0x07) << 18) \
| ((p)[1] & 0x3F) << 12) \
| ((p)[2] & 0x3F) << 6) \
| ((p)[3] & 0x3F)) \
: !((p)[0] & 0x04) ? (((p)[0] & 0x03) << 24) \
| ((p)[1] & 0x3F) << 18) \
| ((p)[2] & 0x3F) << 12) \
| ((p)[3] & 0x3F) << 6) \
| ((p)[4] & 0x3F)) \
: (((p)[0] & 0x01) << 30) \
| ((p)[1] & 0x3F) << 24) \
| ((p)[2] & 0x3F) << 18) \
| ((p)[3] & 0x3F) << 12) \
| ((p)[4] & 0x3F) << 6) \
| ((p)[5] & 0x3F))
```

4.2.1.21 STRING_CHAR_UTF16

```
#define STRING_CHAR_UTF16(
    p )
```

値:

```
((unsigned short) (p)[0] < 0xD800 || (unsigned short) (p)[0] >= 0xDC00) \
? (p)[0] \
: (((p)[0] - 0xD800) << 10) + ((p)[1] - 0xDC00) + 0x10000)
```

4.2.1.22 STRING_CHAR

```
#define STRING_CHAR STRING_CHAR_UTF8
```

4.2.1.23 STRING_CHAR_ADVANCE_UTF8

```
#define STRING_CHAR_ADVANCE_UTF8(
    p )
```

値:

```
(!(*p) & 0x80) ? (p)++, (p)[-1] \
: !(*p) & 0x20) ? (p) += 2, (((p)[-2] & 0x1F) << 6) \
| ((p)[-1] & 0x3F)) \
: !(*p) & 0x10) ? (p) += 3, (((p)[-3] & 0x0F) << 12) \
| ((p)[-2] & 0x3F) << 6) \
| ((p)[-1] & 0x3F)) \
: !(*p) & 0x08) ? (p) += 4, (((p)[-4] & 0x07) << 18) \
```

```

        | (((p)[-3] & 0x3F) << 12) \
        | (((p)[-2] & 0x3F) << 6) \
        | (((p)[-1] & 0x3F))) \
: !((p) & 0x04) ? ((p) += 5, (((p)[-5] & 0x03) << 24) \
        | (((p)[-4] & 0x3F) << 18) \
        | (((p)[-3] & 0x3F) << 12) \
        | (((p)[-2] & 0x3F) << 6) \
        | (((p)[-1] & 0x3F))) \
: ((p) += 6, (((p)[-6] & 0x01) << 30) \
        | (((p)[-5] & 0x3F) << 24) \
        | (((p)[-4] & 0x3F) << 18) \
        | (((p)[-3] & 0x3F) << 12) \
        | (((p)[-2] & 0x3F) << 6) \
        | (((p)[-1] & 0x3F))))

```

4.2.1.24 STRING_CHAR_ADVANCE_UTF16

```

#define STRING_CHAR_ADVANCE_UTF16(
    p )

```

値:

```

(((p)[0] < 0xD800 || (p)[0] >= 0xDC00) \
 ? ((p)++, (p)[-1]) \
 : ((p) += 2, (((p)[-2] - 0xD800) << 10) + ((p)[-1] - 0xDC00) + 0x10000)))

```

4.2.1.25 STRING_CHAR_ADVANCE

```

#define STRING_CHAR_ADVANCE STRING\_CHAR\_ADVANCE\_UTF8

```

4.2.1.26 STRING_CHAR_AND_UNITS_UTF8

```

#define STRING_CHAR_AND_UNITS_UTF8(
    p,
    bytes )

```

値:

```

(!((p)[0] & 0x80) ? ((bytes) = 1, (p)[0]) \
: !((p)[0] & 0x20) ? ((bytes) = 2, \
    (((p)[0] & 0x1F) << 6) \
    | ((p)[1] & 0x3F))) \
: !((p)[0] & 0x10) ? ((bytes) = 3, \
    (((p)[0] & 0x0F) << 12) \
    | (((p)[1] & 0x3F) << 6) \
    | ((p)[2] & 0x3F))) \
: !((p)[0] & 0x08) ? ((bytes) = 4, \
    (((p)[0] & 0x07) << 18) \
    | (((p)[1] & 0x3F) << 12) \
    | (((p)[2] & 0x3F) << 6) \
    | ((p)[3] & 0x3F))) \
: !((p)[0] & 0x04) ? ((bytes) = 5, \
    (((p)[0] & 0x03) << 24) \
    | (((p)[1] & 0x3F) << 18) \
    | (((p)[2] & 0x3F) << 12) \
    | (((p)[3] & 0x3F) << 6) \
    | ((p)[4] & 0x3F))) \
: ((bytes) = 6, \
    (((p)[0] & 0x01) << 30) \
    | (((p)[1] & 0x3F) << 24) \
    | (((p)[2] & 0x3F) << 18) \
    | (((p)[3] & 0x3F) << 12) \
    | (((p)[4] & 0x3F) << 6) \
    | (((p)[5] & 0x3F))))

```

4.2.1.27 STRING_CHAR_AND_UNITS_UTF16

```
#define STRING_CHAR_AND_UNITS_UTF16(
    p,
    units )
```

値:

```
((unsigned short) (p)[0] < 0xD800 || (unsigned short) (p)[0] >= 0xDC00) \
? ((units) = 1, (p)[0]) \
: ((units) = 2, \
   ((p)[0] - 0xD800) << 10) + ((p)[1] - 0xDC00) + 0x10000))
```

4.2.1.28 STRING_CHAR_AND_UNITS

```
#define STRING_CHAR_AND_UNITS(
    p,
    units,
    format )
```

値:

```
((format) <= MTEXT_FORMAT_UTF_8 \
? STRING_CHAR_AND_UNITS_UTF8 (p, units) \
: (format) <= MTEXT_FORMAT_UTF16BE \
? STRING_CHAR_AND_UNITS_UTF16 (p, units) \
: ((units) = 1, ((unsigned) (p))[0]))
```

4.2.1.29 STRING_CHAR_AND_BYTES

```
#define STRING_CHAR_AND_BYTES STRING\_CHAR\_AND\_UNITS\_UTF8
```

4.2.1.30 CHAR_STRING_UTF8

```
#define CHAR_STRING_UTF8(
    c,
    p )
```

値:

```
((c) < 0x80 \
? ((p)[0] = (c), 1) \
: (c) < 0x800 ? ((p)[0] = (0xC0 | ((c) >> 6)), \
                (p)[1] = (0x80 | ((c) & 0x3F)), \
                2) \
: (c) < 0x10000 ? ((p)[0] = (0xE0 | ((c) >> 12)), \
                  (p)[1] = (0x80 | ((c) >> 6) & 0x3F), \
                  (p)[2] = (0x80 | ((c) & 0x3F)), \
                  3) \
: (c) < 0x200000 ? ((p)[0] = (0xF0 | ((c) >> 18)), \
                   (p)[1] = (0x80 | ((c) >> 12) & 0x3F), \
                   (p)[2] = (0x80 | ((c) >> 6) & 0x3F), \
                   (p)[3] = (0x80 | ((c) & 0x3F)), \
                   4) \
: (c) < 0x4000000 ? ((p)[0] = 0xF8, \
                    (p)[1] = (0x80 | ((c) >> 18)), \
                    (p)[2] = (0x80 | ((c) >> 12) & 0x3F), \
                    (p)[3] = (0x80 | ((c) >> 6) & 0x3F), \
                    (p)[4] = (0x80 | ((c) & 0x3F)), \
                    5) \
: ((p)[0] = (0xFC | ((c) >> 30)), \
   (p)[1] = (0x80 | ((c) >> 24) & 0x3F), \
   (p)[2] = (0x80 | ((c) >> 18) & 0x3F), \
   (p)[3] = (0x80 | ((c) >> 12) & 0x3F), \
   (p)[4] = (0x80 | ((c) >> 6) & 0x3F), \
   (p)[5] = (0x80 | ((c) & 0x3F)), \
   6))
```

4.2.1.31 CHAR_STRING_UTF16

```
#define CHAR_STRING_UTF16(
    c,
    p )
```

値:

```
((c) < 0x10000 ? (p)[0] = (c), 1 \
: (p[0] = ((c) - 0x10000) >> 10) + 0xD800, \
  p[1] = ((c) - 0x10000) & 0x3FFF) + 0xDC00, \
  2))
```

4.2.1.32 CHAR_STRING

```
#define CHAR_STRING CHAR_STRING_UTF8
```

4.2.1.33 CHAR_HEAD_P_UTF8

```
#define CHAR_HEAD_P_UTF8(
    p ) ((*(p) & 0xC0) != 0x80)
```

4.2.1.34 CHAR_HEAD_P_UTF16

```
#define CHAR_HEAD_P_UTF16(
    p )
```

値:

```
((*(unsigned short *) (p) < 0xDC00 \
|| *(unsigned short *) (p) >= 0xE000)
```

4.2.1.35 CHAR_HEAD_P

```
#define CHAR_HEAD_P CHAR_HEAD_P_UTF8
```

4.2.1.36 TOLOWER

```
#define TOLOWER(
    c ) (((c) >= 'A' && (c) <= 'Z') ? (c) + 32 : (c))
```

4.2.1.37 TOUPPER

```
#define TOUPPER(
    c ) ((c) >= 'a' && (c) <= 'z') ? (c) - 32 : (c))
```

4.2.1.38 ISUPPER

```
#define ISUPPER(
    c ) ((c) >= 'A' && (c) <= 'Z')
```

4.2.1.39 ISALNUM

```
#define ISALNUM(
    c )
```

値:

```
((c) >= 'A' && (c) <= 'Z') \
|| ((c) >= 'a' && (c) <= 'z') \
|| ((c) >= '0' && (c) <= '9'))
```

4.2.2 関数詳解

4.2.2.1 mchar_define_prop()

```
void mchar_define_prop (
    MSymbol key,
    MSymbol type,
    void * mdb )
```

4.3 charset.c ファイル

マクロ定義

- `#define MCHAR_INVALID_CODE`
無効なコードポイント.

関数

- [MSymbol mchar_define_charset](#) (const char *name, [MPlist](#) *plist)
- [MSymbol mchar_resolve_charset](#) ([MSymbol](#) symbol)
文字セット名を解決する。
- int [mchar_list_charset](#) ([MSymbol](#) **symbols)
文字セットを表わすシンボルを列挙する。
- int [mchar_decode](#) ([MSymbol](#) charset_name, unsigned code)
コードポイントをデコードする。
- unsigned [mchar_encode](#) ([MSymbol](#) charset_name, int c)
文字コードをエンコードする。
- int [mchar_map_charset](#) ([MSymbol](#) charset_name, void(*func)(int from, int to, void *arg), void *func_arg)
指定した文字セットのすべての文字に対して関数を呼ぶ。

変数

変数: 文字セットを表現する定義済みシンボル。

以下の各シンボルは、定義済み文字セットを表現する。

- [MSymbol Mcharset_ascii](#)
ASCII 文字セットを表現するシンボル。
- [MSymbol Mcharset_iso_8859_1](#)
ISO/IEC 8859-1:1998 文字セットを表現するシンボル。
- [MSymbol Mcharset_unicode](#)
Unicode 文字セットを表現するシンボル。
- [MSymbol Mcharset_m17n](#)
全文字を含む文字セットを表現するシンボル。
- [MSymbol Mcharset_binary](#)
正しくデコードできない文字の文字セットを表現するシンボル。

変数: [mchar_define_charset](#) 用のパラメータ・キー

これらは、関数 [mchar_define_charset\(\)](#) 用のパラメータ・キーとして使われるシンボルである。詳しくはこの関数の解説を参照のこと。

- [MSymbol Mmethod](#)
- [MSymbol Mdimension](#)
- [MSymbol Mmin_range](#)
- [MSymbol Mmax_range](#)
- [MSymbol Mmin_code](#)
- [MSymbol Mmax_code](#)
- [MSymbol Mascii_compatible](#)
- [MSymbol Mfinal_byte](#)
- [MSymbol Mrevision](#)
- [MSymbol Mmin_char](#)
- [MSymbol Mmapfile](#)
- [MSymbol Mparents](#)
- [MSymbol Msubset_offset](#)
- [MSymbol Mdefine_coding](#)
- [MSymbol Malias](#)

変数: 文字セットのメソッド指定に使われるシンボル

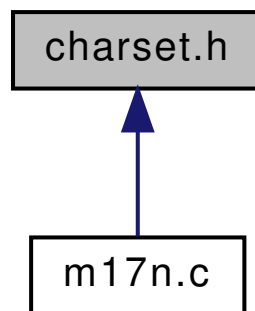
これらは、文字セットのメソッドを指定するための定義済みシンボルであり、文字セットの [Mmethod](#) パラメータの値となることができる。この値は関数 [mchar_define_charset\(\)](#) の引数として使われる。

メソッドとは、コードポイントと文字コードを相互変換する際の方式のことである。詳しくは関数 [mchar_define_charset\(\)](#) の解説を参照のこと。

- [MSymbol Moffset](#)
- [MSymbol Mmap](#)
マップ型のメソッドを示すシンボル.
- [MSymbol Munify](#)
ユニファイ型のメソッドを示すシンボル.
- [MSymbol Msubset](#)
サブセット型のメソッドを示すシンボル.
- [MSymbol Msuperset](#)
スーパーセット型のメソッドを示すシンボル.

4.4 charset.h ファイル

被依存関係図:



データ構造

- struct [MCharset](#)
- struct [MCharsetISO2022Table](#)

マクロ定義

- `#define` [MCHARSET](#)(charset_sym)
- `#define` [CODE_POINT_TO_INDEX](#)(charset, code)
- `#define` [INDEX_TO_CODE_POINT](#)(charset, idx)
- `#define` [DECODE_CHAR](#)(charset, code)
- `#define` [ENCODE_CHAR](#)(charset, c)
- `#define` [ISO_MAX_DIMENSION](#) 3
- `#define` [ISO_MAX_CHARS](#) 2
- `#define` [ISO_MAX_FINAL](#) 0x80
- `#define` [MCHARSET_ISO_2022](#)(dim, chars, final) mcharset_iso_2022_table.classified[(dim) - 1][(chars) == 96][(final)]

列挙型

- `enum mcharset_method {`
`MCHARSET_METHOD_OFFSET ,`
`MCHARSET_METHOD_MAP ,`
`MCHARSET_METHOD_DEFERRED ,`
`MCHARSET_METHOD_SUBSET ,`
`MCHARSET_METHOD_SUPERSET ,`
`MCHARSET_METHOD_MAX }`

関数

- `MCharset * mcharset_find (MSymbol name)`
- `int mcharset_decode_char (MCharset *charset, unsigned code)`
- `unsigned mcharset_encode_char (MCharset *charset, int c)`
- `int mcharset_load_from_database ()`

変数

- `MPlist * mcharset_cache`
- `MCharset * mcharset_ascii`
- `MCharset * mcharset_binary`
- `MCharset * mcharset_m17n`
- `MCharset * mcharset_unicode`
- `MCharsetISO2022Table mcharset_iso_2022_table`

4.4.1 マクロ定義詳解

4.4.1.1 MCHARSET

```
#define MCHARSET(
    charset_sym )
```

値:

```
(( (charset_sym) == MPLIST_KEY (mcharset_cache)
  || (MPLIST_KEY (mcharset_cache) = (charset_sym),
      MPLIST_VAL (mcharset_cache)
      = (MCharset *) msymbol_get ((charset_sym), Mcharset)))
  ? MPLIST_VAL (mcharset_cache)
  : mcharset_find (charset_sym))
```

4.4.1.2 CODE_POINT_TO_INDEX

```
#define CODE_POINT_TO_INDEX(  
    charset,  
    code )
```

値:

```
((charset)->no_code_gap  
? (code) - (charset)->min_code  
: (((charset)->code_range_mask[(code) >> 24] & 0x8)  
  && ((charset)->code_range_mask[(code) >> 16] & 0xFF) & 0x4)  
  && ((charset)->code_range_mask[(code) >> 8] & 0xFF) & 0x2)  
  && ((charset)->code_range_mask[(code) & 0xFF] & 0x1))  
? (((code) >> 24) - (charset)->code_range[12])  
  * (charset)->code_range[11])  
  + (((code) >> 16) & 0xFF) - (charset)->code_range[8])  
  * (charset)->code_range[7])  
  + (((code) >> 8) & 0xFF) - (charset)->code_range[4])  
  * (charset)->code_range[3])  
  + (((code) & 0xFF) - (charset)->code_range[0])  
  - ((charset)->min_code - (charset)->code_range_min_code))  
: -1)
```

4.4.1.3 INDEX_TO_CODE_POINT

```
#define INDEX_TO_CODE_POINT(  
    charset,  
    idx )
```

値:

```
((charset)->no_code_gap  
? (idx) + (charset)->min_code  
: (idx += (charset)->min_code - (charset)->code_range_min_code,  
  (((charset)->code_range[0] + (idx) % (charset)->code_range[2])  
   | (((charset)->code_range[4]  
     + ((idx) / (charset)->code_range[3] % (charset)->code_range[6]))  
     << 8)  
   | (((charset)->code_range[8]  
     + ((idx) / (charset)->code_range[7] % (charset)->code_range[10]))  
     << 16)  
   | (((charset)->code_range[12] + ((idx) / (charset)->code_range[11]))  
     << 24))))
```

4.4.1.4 DECODE_CHAR

```
#define DECODE_CHAR(  
    charset,  
    code )
```

値:

```
((code) < 128 && (charset)->ascii_compatible)  
? (int) (code)  
: ((code) < (charset)->min_code || (code) > (charset)->max_code)  
? -1  
: ! (charset)->simple  
? mcharset_decode_char ((charset), (code))  
: (charset)->method == Moffset  
? (code) - (charset)->min_code + (charset)->min_char  
: (charset)->decoder[(code) - (charset)->min_code])
```

4.4.1.5 ENCODE_CHAR

```
#define ENCODE_CHAR(  
    charset,  
    c )
```

値:

```
(! (charset)->simple  
? mcharset_encode_char ((charset), (c)) \  
: ((c) < (charset)->min_char || (c) > (charset)->max_char) \  
? MCHAR_INVALID_CODE \  
: (charset)->method == Moffset \  
? (c) - (charset)->min_char + (charset)->min_code \  
: (unsigned) mchartable_lookup ((charset)->encoder, (c)))
```

4.4.1.6 ISO_MAX_DIMENSION

```
#define ISO_MAX_DIMENSION 3
```

4.4.1.7 ISO_MAX_CHARS

```
#define ISO_MAX_CHARS 2
```

4.4.1.8 ISO_MAX_FINAL

```
#define ISO_MAX_FINAL 0x80
```

4.4.1.9 MCHARSET_ISO_2022

```
#define MCHARSET_ISO_2022(  
    dim,  
    chars,  
    final ) mcharset_iso_2022_table.classified[(dim) - 1][(chars) == 96][(final)]
```

4.4.2 列挙型詳解

4.4.2.1 mcharset_method

```
enum mcharset_method
```

列挙値

MCHARSET_METHOD_OFFSET	
MCHARSET_METHOD_MAP	
MCHARSET_METHOD_DEFERRED	
MCHARSET_METHOD_SUBSET	
MCHARSET_METHOD_SUPERSET	
MCHARSET_METHOD_MAX	

4.4.3 関数詳解

4.4.3.1 mcharset_find()

```
MCharset* mcharset_find (
    MSymbol name )
```

4.4.3.2 mcharset_decode_char()

```
int mcharset_decode_char (
    MCharset * charset,
    unsigned code )
```

4.4.3.3 mcharset_encode_char()

```
unsigned mcharset_encode_char (
    MCharset * charset,
    int c )
```

4.4.3.4 mcharset_load_from_database()

```
int mcharset_load_from_database ( )
```

4.4.4 変数詳解

4.4.4.1 mcharset_cache

`MPlist*` mcharset_cache [extern]

4.4.4.2 mcharset_ascii

`MCharset*` mcharset_ascii [extern]

4.4.4.3 mcharset_binary

`MCharset*` mcharset_binary [extern]

4.4.4.4 mcharset_m17n

`MCharset*` mcharset_m17n [extern]

4.4.4.5 mcharset_unicode

`MCharset*` mcharset_unicode [extern]

4.4.4.6 mcharset_iso_2022_table

`MCharsetISO2022Table` mcharset_iso_2022_table [extern]

4.5 chartab.c ファイル

関数

- `MCharTable * mchartable (MSymbol key, void *default_value)`
新しい文字テーブルを作る。
 - `int mchartable_min_char (MCharTable *table)`
 - `int mchartable_max_char (MCharTable *table)`
 - `void * mchartable_lookup (MCharTable *table, int c)`
文字テーブル中で文字に割り当てられた値を返す。
 - `int mchartable_set (MCharTable *table, int c, void *val)`
文字テーブル中での文字の値を設定する。
 - `int mchartable_set_range (MCharTable *table, int from, int to, void *val)`
指定範囲の文字に値を設定する。
 - `void mchartable_range (MCharTable *table, int *from, int *to)`
値がデフォルトと異なる文字を探す。
 - `int mchartable_map (MCharTable *table, void *ignore, void(*func)(int, int, void *, void *), void *func_arg)`
文字テーブル中の文字に対して指定の関数を呼ぶ。
-
- `MCharTable * mdebug_dump_chartab (MCharTable *table, int indent)`
文字テーブルをダンプする。

変数

- `MSymbol Mchar_table`

4.5.1 関数詳解

4.5.1.1 mdebug_dump_chartab()

```
MCharTable* mdebug_dump_chartab (
    MCharTable * table,
    int indent )
```

文字テーブルをダンプする。

関数 `mdebug_dump_chartab()` は文字テーブル `table` を標準エラー出力 もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読 な形で印刷する。`indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `table` を返す。

4.6 chartab.h ファイル

関数

- void * [mchartable_lookup](#) ([MCharTable](#) *table, int c, int *next_c, int default_p)

4.6.1 関数詳解

4.6.1.1 mchartable_lookup()

```
void* mchartable_lookup (
    MCharTable * table,
    int c,
    int * next_c,
    int default_p )
```

4.7 coding.c ファイル

関数

- [MSymbol mconv_define_coding](#) (const char *name, [MPlist](#) *plist, int(*resetter)([MConverter](#) *), int(*decoder)(const unsigned char *, int, [MText](#) *, [MConverter](#) *), int(*encoder)([MText](#) *, int, int, unsigned char *, int, [MConverter](#) *), void *extra_info)
- [MSymbol mconv_resolve_coding](#) ([MSymbol](#) symbol)
コード系の名前を解決する。
- int [mconv_list_codings](#) ([MSymbol](#) **symbols)
コード系を表わすシンボルを列挙する。
- [MConverter](#) * [mconv_buffer_converter](#) ([MSymbol](#) name, const unsigned char *buf, int n)
バッファに結び付けられたコードコンバータを作る。
- [MConverter](#) * [mconv_stream_converter](#) ([MSymbol](#) name, FILE *fp)
ストリームに結び付けられたコードコンバータを作る。
- int [mconv_reset_converter](#) ([MConverter](#) *converter)
コードコンバータをリセットする。
- void [mconv_free_converter](#) ([MConverter](#) *converter)
コードコンバータを解放する。
- [MConverter](#) * [mconv_rebind_buffer](#) ([MConverter](#) *converter, const unsigned char *buf, int n)
コードコンバータにバッファ領域を結び付ける。
- [MConverter](#) * [mconv_rebind_stream](#) ([MConverter](#) *converter, FILE *fp)
コードコンバータにストリームを結び付ける。
- [MText](#) * [mconv_decode](#) ([MConverter](#) *converter, [MText](#) *mt)
バイト列を M-text にデコードする。
- [MText](#) * [mconv_decode_buffer](#) ([MSymbol](#) name, const unsigned char *buf, int n)
コード系に基づいてバッファ領域をデコードする。
- [MText](#) * [mconv_decode_stream](#) ([MSymbol](#) name, FILE *fp)

コード系に基づいてストリーム入力をデコードする。

- `int mconv_encode (MConverter *converter, MText *mt)`
M-text をバイト列にエンコードする。
- `int mconv_encode_range (MConverter *converter, MText *mt, int from, int to)`
M-text の一部をバイト列にエンコードする。
- `int mconv_encode_buffer (MSymbol name, MText *mt, unsigned char *buf, int n)`
M-text をエンコードしてバッファ領域に書き込む。
- `int mconv_encode_stream (MSymbol name, MText *mt, FILE *fp)`
M-text をエンコードしてストリームに書き込む。
- `int mconv_getc (MConverter *converter)`
コードコンバータ経由で一文字を読みこむ。
- `int mconv_ungetc (MConverter *converter, int c)`
コードコンバータに一文字戻す。
- `int mconv_putc (MConverter *converter, int c)`
コードコンバータを経由して一文字書き出す。
- `MText * mconv_gets (MConverter *converter, MText *mt)`
コードコンバータを使って一行読み込む。

変数

変数: 定義済みコード系を指定するためのシンボル

- `MSymbol Mcoding_us_ascii`
US-ASCII コード系のシンボル。
- `MSymbol Mcoding_iso_8859_1`
ISO-8859-1 コード系のシンボル。
- `MSymbol Mcoding_utf_8`
UTF-8 コード系のシンボル。
- `MSymbol Mcoding_utf_8_full`
UTF-8-FULL コード系のシンボル。
- `MSymbol Mcoding_utf_16`
UTF-16 コード系のシンボル。
- `MSymbol Mcoding_utf_16be`
UTF-16BE コード系のシンボル。
- `MSymbol Mcoding_utf_16le`
UTF-16LE コード系のシンボル。
- `MSymbol Mcoding_utf_32`
UTF-32 コード系のシンボル。
- `MSymbol Mcoding_utf_32be`
UTF-32BE コード系のシンボル。
- `MSymbol Mcoding_utf_32le`
UTF-32LE コード系のシンボル。
- `MSymbol Mcoding_sjis`
SJIS コード系のシンボル。

変数: `mconv_define_coding()` 用パラメータキー

- `MSymbol Mtype`
- `MSymbol Mcharsets`
- `MSymbol Mflags`
- `MSymbol Mdesignation`
- `MSymbol Minvocation`

- [MSymbol Mcode_unit](#)
- [MSymbol Mbom](#)
- [MSymbol Mlittle_endian](#)

変数：コード系のタイプを示すシンボル。

- [MSymbol Mutf](#)
- [MSymbol Miso_2022](#)

変数：パラメータ **Mflags** の値となり得るシンボル。

関数 [mconv_define_coding\(\)](#) の引数として用いられるコード系のパラメータ **Mflags** の値となり得るシンボル。(詳細は [mconv_define_coding\(\)](#) 参照)。

- [MSymbol Mreset_at_eol](#)
- [MSymbol Mreset_at_cnl](#)
- [MSymbol Meight_bit](#)
- [MSymbol Mlong_form](#)
- [MSymbol Mdesignation_g0](#)
- [MSymbol Mdesignation_g1](#)
- [MSymbol Mdesignation_ctxt](#)
- [MSymbol Mdesignation_ctxt_ext](#)
- [MSymbol Mlocking_shift](#)
- [MSymbol Msingle_shift](#)
- [MSymbol Msingle_shift_7](#)
- [MSymbol Meuc_tw_shift](#)
- [MSymbol Miso_6429](#)
- [MSymbol Mrevision_number](#)
- [MSymbol Mfull_support](#)

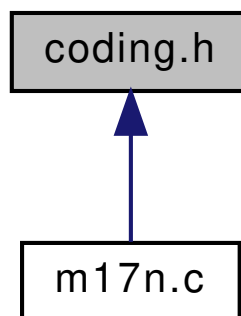
変数: その他

ほかの変数。

- [MSymbol Mmaybe](#)
"maybe"という名前を持つシンボル。
- [MSymbol Mcoding](#)
シンボル `Mcoding`。

4.8 coding.h ファイル

被依存関係図:



関数

- void `mconv__register_charset_coding` (MSymbol name)
- int `mcoding_load_from_database` ()

4.8.1 関数詳解

4.8.1.1 `mconv__register_charset_coding()`

```
void mconv__register_charset_coding (
    MSymbol name )
```

4.8.1.2 `mcoding_load_from_database()`

```
int mcoding_load_from_database ( )
```

4.9 database.c ファイル

関数

- MDatabase * `mdatabase_find` (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3)
データベース中のデータを探す.
- MPlist * `mdatabase_list` (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3)
m17n データベースのデータリストを返す.
- MDatabase * `mdatabase_define` (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3, void (*loader)(MSymbol *, void *), void *extra_info)
m17n データベースのデータを定義する.
- void * `mdatabase_load` (MDatabase *mdb)
データベースからデータをロードする.
- MSymbol * `mdatabase_tag` (MDatabase *mdb)
データのタグを得る.

変数

- MSymbol `Mcharset`
- char * `mdatabase_dir`

4.10 database.h ファイル

データ構造

- struct [MDatabaseInfo](#)

マクロ定義

- #define [M17NDIR](#) "/usr/local/share/m17n"
- #define [PATH_MAX](#) 1024
- #define [PATH_SEPARATOR](#) '/'

列挙型

- enum [MDatabaseStatus](#) {
 [MDB_STATUS_AUTO](#) ,
 [MDB_STATUS_AUTO_WILDCARD](#) ,
 [MDB_STATUS_EXPLICIT](#) ,
 [MDB_STATUS_DISABLED](#) ,
 [MDB_STATUS_UPDATED](#) ,
 [MDB_STATUS_OUTDATED](#) }

関数

- void [mdatabase_update](#) (void)
- [MPlist](#) * [mdatabase_load_for_keys](#) ([MDatabase](#) *mdb, [MPlist](#) *keys)
- int [mdatabase_check](#) ([MDatabase](#) *mdb)
- char * [mdatabase_find_file](#) (char *filename)
- char * [mdatabase_file](#) ([MDatabase](#) *mdb)
- int [mdatabase_lock](#) ([MDatabase](#) *mdb)
- int [mdatabase_save](#) ([MDatabase](#) *mdb, [MPlist](#) *data)
- int [mdatabase_unlock](#) ([MDatabase](#) *mdb)
- [MPlist](#) * [mdatabase_props](#) ([MDatabase](#) *mdb)

変数

- [MPlist](#) * [mdatabase_dir_list](#)
- void (*)([mdatabase_load_charset_func](#))(FILE *fp, [MSymbol](#) charset_name)

4.10.1 マクロ定義詳解

4.10.1.1 M17NDIR

```
#define M17NDIR "/usr/local/share/m17n"
```

4.10.1.2 PATH_MAX

```
#define PATH_MAX 1024
```

4.10.1.3 PATH_SEPARATOR

```
#define PATH_SEPARATOR '/'
```

4.10.2 列挙型詳解

4.10.2.1 MDatabaseStatus

```
enum MDatabaseStatus
```

列挙値

MDB_STATUS_AUTO	
MDB_STATUS_AUTO_WILDCARD	
MDB_STATUS_EXPLICIT	
MDB_STATUS_DISABLED	
MDB_STATUS_UPDATED	
MDB_STATUS_OUTDATED	

4.10.3 関数詳解

4.10.3.1 mdatabase_update()

```
void mdatabase_update (  
    void )
```

4.10.3.2 mdatabase_load_for_keys()

```
MPlist* mdatabase_load_for_keys (
    MDatabase * mdb,
    MPlist * keys )
```

4.10.3.3 mdatabase_check()

```
int mdatabase_check (
    MDatabase * mdb )
```

4.10.3.4 mdatabase_find_file()

```
char* mdatabase_find_file (
    char * filename )
```

4.10.3.5 mdatabase_file()

```
char* mdatabase_file (
    MDatabase * mdb )
```

4.10.3.6 mdatabase_lock()

```
int mdatabase_lock (
    MDatabase * mdb )
```

4.10.3.7 mdatabase_save()

```
int mdatabase_save (
    MDatabase * mdb,
    MPlist * data )
```

4.10.3.8 mdatabase_unlock()

```
int mdatabase_unlock (
    MDatabase * mdb )
```

4.10.3.9 mdatabase_props()

```
MList* mdatabase_props (
    MDatabase * mdb )
```

4.10.4 変数詳解

4.10.4.1 mdatabase_dir_list

```
MList* mdatabase_dir_list [extern]
```

4.10.4.2 mdatabase_load_charset_func

```
void*(* mdatabase_load_charset_func) (FILE *fp, MSymbol charset_name) (
    FILE * fp,
    MSymbol charset_name ) [extern]
```

4.11 dbdata.txt ファイル

4.12 dbformat.txt ファイル

4.13 dbtutorial.txt ファイル

4.14 draw.c ファイル

関数

- int [mdraw_text](#) (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to)
ウィンドウに M-text を描画する.
- int [mdraw_image_text](#) (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to)

ディスプレイにM-text を画像として描く。

- int `mdraw_text_with_control` (MFrame *frame, MDrawWindow win, int x, int y, MText *mt, int from, int to, MDrawControl *control)

ディスプレイにM-text を詳細な制御つきで描く。

- int `mdraw_text_extents` (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawMetric *overallink_return, MDrawMetric *overallLogical_return, MDrawMetric *overallLine_return)

テキストの幅（ピクセル単位）を計算する。

- int `mdraw_text_per_char_extents` (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawMetric *ink_array_return, MDrawMetric *logical_array_return, int array_size, int *num_chars_return, MDrawMetric *overallink_return, MDrawMetric *overallLogical_return)

M-text の各文字の表示範囲を計算する。

- int `mdraw_coordinates_position` (MFrame *frame, MText *mt, int from, int to, int x_offset, int y_offset, MDrawControl *control)

指定した座標に最も近い文字の文字位置を得る。

- int `mdraw_glyph_info` (MFrame *frame, MText *mt, int from, int pos, MDrawControl *control, MDrawGlyphInfo *info)

グリフに関する情報を計算する。

- int `mdraw_glyph_list` (MFrame *frame, MText *mt, int from, int to, MDrawControl *control, MDrawGlyph *glyphs, int array_size, int *num_glyphs_return)

グリフ列に関する情報を計算する。

- void `mdraw_text_items` (MFrame *frame, MDrawWindow win, int x, int y, MDrawTextItem *items, int nitems)

textitem を表示する。

- int `mdraw_default_line_break` (MText *mt, int pos, int from, int to, int line, int y)

改行位置を計算する。

- void `mdraw_per_char_extents` (MFrame *frame, MText *mt, MDrawMetric *array_return, MDrawMetric *overall_return)

M-text の文字毎の表示範囲情報を得る。

- void `mdraw_clear_cache` (MText *mt)

キャッシュ情報を消す。

変数

- int `mdraw_line_break_option`

4.15 exprog.txt ファイル

4.16 face.c ファイル

関数

- MFace * `mface` ()
新しいフェースをつくる。
- MFace * `mface_copy` (MFace *face)
フェースのコピーを作る。
- int `mface_equal` (MFace *face1, MFace *face2)
- MFace * `mface_merge` (MFace *dst, MFace *src)

フェースを統合する.

- `MFace * mface_from_font (MFont *font)`
フォントからフェースを作る.
- `void * mface_get_prop (MFace *face, MSymbol key)`
フェースのプロパティの値を得る.
- `MFaceHookFunc mface_get_hook (MFace *face)`
フェースのフック関数を得る.
- `int mface_put_prop (MFace *face, MSymbol key, void *val)`
フェースプロパティの値を設定する.
- `int mface_put_hook (MFace *face, MFaceHookFunc func)`
フェースのフック関数を設定する.
- `void mface_update (MFrame *frame, MFace *face)`
フェースを更新する.
- `MFace * mdebug_dump_face (MFace *face, int indent)`
フェースをダンプする.

変数

変数: フェースプロパティのキー

- `MSymbol Mforeground`
前景色を指定するフェースプロパティのキー.
- `MSymbol Mbackground`
背景色を指定するためのフェースプロパティのキー.
- `MSymbol Mvideomode`
ビデオモードを指定するためのフェースプロパティのキー.
- `MSymbol Mratio`
フォントのサイズの比率を指定するためのフェースプロパティのキー.
- `MSymbol Mhline`
水平線を指定するためのフェースプロパティのキー.
- `MSymbol Mbox`
囲み枠を指定するためのフェースプロパティのキー.
- `MSymbol Mfontset`
フォントセットを指定するためのフェースプロパティのキー.
- `MSymbol Mhook_func`
フックを指定するためのフェースプロパティのキー.
- `MSymbol Mhook_arg`
フックの引数を指定するためのフェースプロパティのキー.

変数: フェースの `#Mvideomode` プロパティの可能な値

変数 `Mvideomode` の説明を参照のこと。

- `MSymbol Mnormal`
- `MSymbol Mreverse`

変数: 定義済みフェース

- `MFace * mface_normal_video`
標準ビデオフェース.
- `MFace * mface_reverse_video`
リバーズビデオフェース.

- `MFace * mface_underline`
- `MFace * mface_medium`
ミディアムフェース.
- `MFace * mface_bold`
ボールドフェース.
- `MFace * mface_italic`
イタリックフェース.
- `MFace * mface_bold_italic`
ボールドイタリックフェース.
- `MFace * mface_xx_small`
最小のフェース.
- `MFace * mface_x_small`
より小さいフェース.
- `MFace * mface_small`
小さいフェース.
- `MFace * mface_normalsize`
標準の大きさのフェース.
- `MFace * mface_large`
大きいフェース.
- `MFace * mface_x_large`
もっと大きいフェース.
- `MFace * mface_xx_large`
最大のフェース.
- `MFace * mface_black`
黒フェース.
- `MFace * mface_white`
白フェース.
- `MFace * mface_red`
赤フェース.
- `MFace * mface_green`
緑フェース.
- `MFace * mface_blue`
青フェース.
- `MFace * mface_cyan`
シアンフェース.
- `MFace * mface_yellow`
黄フェース.
- `MFace * mface_magenta`
マゼンタフェース.

変数: フェースを取り扱うためのその他のシンボル

- `MSymbol Mface`
フェースを指定するテキストプロパティのキー.

4.17 face.h ファイル

データ構造

- `struct MFace`
フェースの型宣言.
- `struct MRealizedFace`

列挙型

- enum MFaceProperty {
MFACE_FOUNDRY ,
MFACE_FAMILY ,
MFACE_WEIGHT ,
MFACE_STYLE ,
MFACE_STRETCH ,
MFACE_ADSTYLE ,
MFACE_SIZE ,
MFACE_FONTSET ,
MFACE_FOREGROUND ,
MFACE_BACKGROUND ,
MFACE_HLINE ,
MFACE_BOX ,
MFACE_VIDEOMODE ,
MFACE_RATIO ,
MFACE_HOOK_ARG ,
MFACE_PROPERTY_MAX }

関数

- MRealizedFace * mface_realize (MFrame *frame, MFace **faces, int num, int limited_size, MFont *font)
- MGlyph * mface_for_chars (MSymbol script, MSymbol language, MSymbol charset, MGlyph *from_g, MGlyph *to_g, int size)
- void mface_free_realized (MRealizedFace *rface)
- void mface_update_frame_face (MFrame *frame)

変数

- MFace * mface_default

4.17.1 列挙型詳解

4.17.1.1 MFaceProperty

enum MFaceProperty

列挙値

MFACE_FOUNDRY	
MFACE_FAMILY	
MFACE_WEIGHT	
MFACE_STYLE	
MFACE_STRETCH	
MFACE_ADSTYLE	

列挙値

MFACE_SIZE	
MFACE_FONTSET	
MFACE_FOREGROUND	
MFACE_BACKGROUND	
MFACE_HLINE	
MFACE_BOX	
MFACE_VIDEOMODE	
MFACE_RATIO	
MFACE_HOOK_ARG	
MFACE_PROPERTY_MAX	

4.17.2 関数詳解

4.17.2.1 mface_realize()

```
MRealizedFace* mface_realize (
    MFrame * frame,
    MFace ** faces,
    int num,
    int limited_size,
    MFont * font )
```

4.17.2.2 mface_for_chars()

```
MGlyph* mface_for_chars (
    MSymbol script,
    MSymbol language,
    MSymbol charset,
    MGlyph * from_g,
    MGlyph * to_g,
    int size )
```

4.17.2.3 mface_free_realized()

```
void mface_free_realized (
    MRealizedFace * rface )
```

4.17.2.4 mface_update_frame_face()

```
void mface_update_frame_face (
    MFrame * frame )
```

4.17.3 変数詳解

4.17.3.1 mface_default

```
MFace* mface_default [extern]
```

4.18 fdl.txt ファイル

4.19 font.c ファイル

関数

- **MFont * mfont ()**
新しいフォントを作る.
- **MFont * mfont_parse_name (const char *name, MSymbol format)**
フォント名からフォントを作る.
- **char * mfont_unparse_name (MFont *font, MSymbol format)**
フォントからフォント名を作る.
- **MFont * mfont_copy (MFont *font)**
フォントのコピーを作る.
- **void * mfont_get_prop (MFont *font, MSymbol key)**
フォントのプロパティの値を得る.
- **int mfont_put_prop (MFont *font, MSymbol key, void *val)**
フォントのプロパティに値を設定する.
- **MSymbol * mfont_selection_priority ()**
フォント選択の優先度を返す.
- **int mfont_set_selection_priority (MSymbol *keys)**
フォント選択優先度を設定する.
- **MFont * mfont_find (MFrame *frame, MFont *spec, int *score, int max_size)**
フォントを探す.
- **int mfont_set_encoding (MFont *font, MSymbol encoding_name, MSymbol repertory_name)**
フォントのエンコーディングを設定する.
- **char * mfont_name (MFont *font)**
フォント名からフォントを作る.
- **MFont * mfont_from_name (const char *name)**
フォントからフォント名を作る.
- **int mfont_resize_ratio (MFont *font)**

フォントのリサイズ情報を得る

- `MPlist * mfont_list (MFrame *frame, MFont *font, MSymbol language, int maxnum)`

フォントのリストを得る

- `MPlist * mfont_list_family_names (MFrame *frame)`
- `int mfont_check (MFrame *frame, MFontset *fontset, MSymbol script, MSymbol language, MFont *font)`
- `int mfont_match_p (MFont *font, MFont *spec)`
- `MFont * mfont_open (MFrame *frame, MFont *font)`
- `MFont * mfont_encapsulate (MFrame *frame, MSymbol data_type, void *data)`
- `int mfont_close (MFont *font)`

- `MFont * mdebug_dump_font (MFont *font)`

フォントをダンプする.

変数

- `MPlist * mfont_freetype_path`

フォントファイルとフォントファイルを含むディレクトリのリスト.

変数: フォントプロパティを指定する定義済みシンボル

- `MSymbol Mfoundry`
開発元を指定するフォントプロパティのキー.
- `MSymbol Mfamily`
ファミリを指定するフォントプロパティのキー.
- `MSymbol Mweight`
太さを指定するフォントプロパティのキー.
- `MSymbol Mstyle`
スタイルを指定するフォントプロパティのキー.
- `MSymbol Mstretch`
幅を指定するフォントプロパティのキー.
- `MSymbol Madstyle`
adstyle を指定するフォントプロパティのキー.
- `MSymbol Mspacing`
spacing を指定するフォントプロパティのキー.
- `MSymbol Mregistry`
レジストリを指定するフォントプロパティのキー.
- `MSymbol Msize`
サイズを指定するフォントプロパティのキー.
- `MSymbol Mottf`
- `MSymbol Mfontfile`
フォントファイルを指定するフォントプロパティのキー.
- `MSymbol Mresolution`
解像度を指定するフォントプロパティのキー.
- `MSymbol Mmax_advance`
- `MSymbol Mfontconfig`
"fontconfig" という名前を持つシンボル.
- `MSymbol Mx`
"x" という名前を持つシンボル.
- `MSymbol Mfreetype`
"freetype" という名前を持つシンボル.
- `MSymbol Mxft`
"xft" という名前を持つシンボル.

4.19.1 関数詳解

4.19.1.1 mdebug_dump_font()

```
MFont* mdebug_dump_font (
    MFont * font )
```

フォントをダンプする。

関数 `mdebug_dump_font()` はフォント `font` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で出力する。

戻り値:

この関数は `font` を返す。

4.20 font.h ファイル

データ構造

- struct `MFont`
フォントの型宣言.
- struct `MFontPropertyTable`
- struct `MRealizedFont`
- struct `MFLTFontForRealized`
- struct `MFontScore`
- struct `MFontList`
- struct `MFontDriver`
- struct `MFontCapability`

マクロ定義

- `#define FONT_PROPERTY(font, n) (mfont_property_table[(n)].names[(font)->property[(n)])]`
- `#define MFONT_INIT(font) memset ((font), 0, sizeof (MFont))`

型定義

- `typedef struct MFontEncoding MFontEncoding`
- `typedef unsigned OTF_Tag`

列挙型

- enum MFontProperty {
MFONT_FOUNDRY ,
MFONT_FAMILY ,
MFONT_WEIGHT ,
MFONT_STYLE ,
MFONT_STRETCH ,
MFONT_ADSTYLE ,
MFONT_REGISTRY ,
MFONT_RESY ,
MFONT_SIZE ,
MFONT_SPACING ,
MFONT_PROPERTY_MAX = MFONT_SIZE }
- enum MFontType {
MFONT_TYPE_SPEC ,
MFONT_TYPE_OBJECT ,
MFONT_TYPE_REALIZED ,
MFONT_TYPE_FAILURE }
- enum MFontSource {
MFONT_SOURCE_UNDECIDED = 0 ,
MFONT_SOURCE_X = 1 ,
MFONT_SOURCE_FT = 2 }
- enum MFontSpacing {
MFONT_SPACING_UNDECIDED ,
MFONT_SPACING_PROPORTIONAL ,
MFONT_SPACING_MONO ,
MFONT_SPACING_CHARCELL }
- enum MFontOpenTypeTable {
MFONT_OTT_GSUB ,
MFONT_OTT_GPOS ,
MFONT_OTT_MAX }

関数

- int mfont__flt_init ()
- void mfont__flt_fini ()
- void mfont__free_realized (MRealizedFont *rfont)
- int mfont__match_p (MFont *font, MFont *spec, int prop)
- int mfont__merge (MFont *dst, MFont *src, int error_on_conflict)
- void mfont__set_spec_from_face (MFont *spec, MFace *face)
- MSymbol mfont__set_spec_from_plist (MFont *spec, MPlist *plist)
- int mfont__has_char (MFrame *frame, MFont *font, MFont *spec, int c)
- unsigned mfont__encode_char (MFrame *frame, MFont *font, MFont *spec, int c)
- int mfont__get_glyph_id (MFLTFont *font, MFLTGlyphString *gstring, int from, int to)
- MFont * mfont__select (MFrame *frame, MFont *font, int max_size)
- MFontList * mfont__list (MFrame *frame, MFont *spec, MFont *request, int limited_size)
- MRealizedFont * mfont__open (MFrame *frame, MFont *font, MFont *spec)
- void mfont__get_metric (MGlyphString *gstring, int from, int to)
- int mfont__get_metrics (MFLTFont *font, MFLTGlyphString *gstring, int from, int to)
- void mfont__set_property (MFont *font, enum MFontProperty key, MSymbol val)
- int mfont__split_name (char *name, int *property_idx, unsigned short *point, unsigned short *resy)
- int mfont__parse_name_into_font (const char *name, MSymbol format, MFont *font)

- `MPlist * mfont_encoding_list` (void)
- `MFontCapability * mfont_get_capability` (MSymbol sym)
- `int mfont_check_capability` (MRealizedFont *rfont, MSymbol capability)
- `unsigned mfont_ftl_encode_char` (MSymbol layouter_name, int c)
- `int mfont_ftl_run` (MGlyphString *gstring, int from, int to, MRealizedFace *rface)

変数

- `MFontPropertyTable mfont_property_table` [MFONT_REGISTRY+1]
- `MSymbol Mlayouter`
- `MSymbol Miso8859_1`
- `MSymbol Miso10646_1`
- `MSymbol Municode_bmp`
- `MSymbol Municode_full`
- `MSymbol Mapple_roman`

4.20.1 マクロ定義詳解

4.20.1.1 FONT_PROPERTY

```
#define FONT_PROPERTY(  
    font,  
    n )  (mfont_property_table[ (n) ].names[ (font)->property[ (n) ] ])
```

4.20.1.2 MFONT_INIT

```
#define MFONT_INIT(  
    font )  memset ((font), 0, sizeof (MFont))
```

4.20.2 型定義詳解

4.20.2.1 MFontEncoding

```
typedef struct MFontEncoding MFontEncoding
```

4.20.2.2 OTF_Tag

```
typedef unsigned OTF_Tag
```

4.20.3 列挙型詳解

4.20.3.1 MFontProperty

```
enum MFontProperty
```

列挙値

MFONT_FOUNDRY	
MFONT_FAMILY	
MFONT_WEIGHT	
MFONT_STYLE	
MFONT_STRETCH	
MFONT_ADSTYLE	
MFONT_REGISTRY	
MFONT_RESY	
MFONT_SIZE	
MFONT_SPACING	
MFONT_PROPERTY_MAX	

4.20.3.2 MFontType

```
enum MFontType
```

列挙値

MFONT_TYPE_SPEC	
MFONT_TYPE_OBJECT	
MFONT_TYPE_REALIZED	
MFONT_TYPE_FAILURE	

4.20.3.3 MFontSource

```
enum MFontSource
```

列挙値

MFONT_SOURCE_UNDECIDED	
MFONT_SOURCE_X	
MFONT_SOURCE_FT	

4.20.3.4 MFontSpacing

enum `MFontSpacing`

列挙値

MFONT_SPACING_UNDECIDED	
MFONT_SPACING_PROPORTIONAL	
MFONT_SPACING_MONO	
MFONT_SPACING_CHARCELL	

4.20.3.5 MFontOpenTypeTable

enum `MFontOpenTypeTable`

列挙値

MFONT_OTT_GSUB	
MFONT_OTT_GPOS	
MFONT_OTT_MAX	

4.20.4 関数詳解

4.20.4.1 mfont_flt_init()

```
int mfont_flt_init ( )
```

4.20.4.2 mfont__flt_fini()

```
void mfont__flt_fini ( )
```

4.20.4.3 mfont__free_realized()

```
void mfont__free_realized (
    MRealizedFont * rfont )
```

4.20.4.4 mfont__match_p()

```
int mfont__match_p (
    MFont * font,
    MFont * spec,
    int prop )
```

4.20.4.5 mfont__merge()

```
int mfont__merge (
    MFont * dst,
    MFont * src,
    int error_on_conflict )
```

4.20.4.6 mfont__set_spec_from_face()

```
void mfont__set_spec_from_face (
    MFont * spec,
    MFace * face )
```

4.20.4.7 mfont__set_spec_from_plist()

```
MSymbol mfont__set_spec_from_plist (
    MFont * spec,
    MPlist * plist )
```

4.20.4.8 mfont_has_char()

```
int mfont_has_char (
    MFrame * frame,
    MFont * font,
    MFont * spec,
    int c )
```

4.20.4.9 mfont_encode_char()

```
unsigned mfont_encode_char (
    MFrame * frame,
    MFont * font,
    MFont * spec,
    int c )
```

4.20.4.10 mfont_get_glyph_id()

```
int mfont_get_glyph_id (
    MFLTFont * font,
    MFLTGlyphString * gstring,
    int from,
    int to )
```

4.20.4.11 mfont_select()

```
MFont* mfont_select (
    MFrame * frame,
    MFont * font,
    int max_size )
```

4.20.4.12 mfont_list()

```
MFontList* mfont_list (
    MFrame * frame,
    MFont * spec,
    MFont * request,
    int limited_size )
```

4.20.4.13 mfont_open()

```
MRealizedFont* mfont_open (
    MFrame * frame,
    MFont * font,
    MFont * spec )
```

4.20.4.14 mfont_get_metric()

```
void mfont_get_metric (
    MGlyphString * gstring,
    int from,
    int to )
```

4.20.4.15 mfont_get_metrics()

```
int mfont_get_metrics (
    MFLTFont * font,
    MFLTGlyphString * gstring,
    int from,
    int to )
```

4.20.4.16 mfont_set_property()

```
void mfont_set_property (
    MFont * font,
    enum MFontProperty key,
    MSymbol val )
```

4.20.4.17 mfont_split_name()

```
int mfont_split_name (
    char * name,
    int * property_idx,
    unsigned short * point,
    unsigned short * resy )
```

4.20.4.18 mfont_parse_name_into_font()

```
int mfont_parse_name_into_font (
    const char * name,
    MSymbol format,
    MFont * font )
```

4.20.4.19 mfont_encoding_list()

```
MPlist* mfont_encoding_list (
    void )
```

4.20.4.20 mfont_get_capability()

```
MFontCapability* mfont_get_capability (
    MSymbol sym )
```

4.20.4.21 mfont_check_capability()

```
int mfont_check_capability (
    MRealizedFont * rfont,
    MSymbol capability )
```

4.20.4.22 mfont_flt_encode_char()

```
unsigned mfont_flt_encode_char (
    MSymbol layouter_name,
    int c )
```

4.20.4.23 mfont_flt_run()

```
int mfont_flt_run (
    MGlyphString * gstring,
    int from,
    int to,
    MRealizedFace * rface )
```

4.20.5 変数詳解

4.20.5.1 mfont_property_table

`MFonPropertyTable` mfont_property_table[MFONT_REGISTRY+1] [extern]

4.20.5.2 Mlayouter

`MSymbol` Mlayouter [extern]

4.20.5.3 Miso8859_1

`MSymbol` Miso8859_1 [extern]

4.20.5.4 Miso10646_1

`MSymbol` Miso10646_1

4.20.5.5 Municode_bmp

`MSymbol` Municode_bmp

4.20.5.6 Municode_full

`MSymbol` Municode_full

4.20.5.7 Mapple_roman

`MSymbol` Mapple_roman [extern]

4.21 fontset.c ファイル

関数

- `MFontset * mfontset (char *name)`
フォントセットを返す.
- `MSymbol mfontset_name (MFontset *fontset)`
フォントセットの名前を返す.
- `MFontset * mfontset_copy (MFontset *fontset, char *name)`
フォントセットのコピーを作る.
- `int mfontset_modify_entry (MFontset *fontset, MSymbol script, MSymbol language, MSymbol charset, MFont *spec, MSymbol layouter_name, int how)`
フォントセットの内容を変更する.
- `MPlist * mfontset_lookup (MFontset *fontset, MSymbol script, MSymbol language, MSymbol charset)`
フォントセットを検索する.
- `MFontset * mdebug_dump_fontset (MFontset *fontset, int indent)`
フォントセットをダンプする.

4.21.1 関数詳解

4.21.1.1 mdebug_dump_fontset()

```
MFontset* mdebug_dump_fontset (
    MFontset * fontset,
    int indent )
```

フォントセットをダンプする.

関数 `mdebug_dump_fontset()` はフォントセット `fontset` を標準エラー出力 もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読 な形で出力する。 `indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `fontset` を返す。

4.22 fontset.h ファイル

関数

- `MRealizedFontset * mfont_realize_fontset (MFrame *frame, MFontset *fontset, MFace *face, MFont *spec)`
- `void mfont_free_realized_fontset (MRealizedFontset *realized)`
- `MRealizedFont * mfont_lookup_fontset (MRealizedFontset *realized, MGlyph *g, int *num, MSymbol script, MSymbol language, MSymbol charset, int size, int ignore_fallback)`
- `MRealizedFont * mfont_get_font (MFrame *frame, MFontset *fontset, MSymbol script, MSymbol language, MFont *font, int *best)`

4.22.1 関数詳解

4.22.1.1 mfont_realize_fontset()

```
MRealizedFontset* mfont_realize_fontset (
    MFrame * frame,
    MFontset * fontset,
    MFace * face,
    MFont * spec )
```

4.22.1.2 mfont_free_realized_fontset()

```
void mfont_free_realized_fontset (
    MRealizedFontset * realized )
```

4.22.1.3 mfont_lookup_fontset()

```
MRealizedFont* mfont_lookup_fontset (
    MRealizedFontset * realized,
    MGlyph * g,
    int * num,
    MSymbol script,
    MSymbol language,
    MSymbol charset,
    int size,
    int ignore_fallback )
```

4.22.1.4 mfontset_get_font()

```
MRealizedFont* mfontset_get_font (
    MFrame * frame,
    MFontset * fontset,
    MSymbol script,
    MSymbol language,
    MFont * font,
    int * best )
```

4.23 input-gui.c ファイル

関数

- **MSymbol minput_event_to_key** (MFrame *frame, void *event)
イベントを入力キーに変換する。

変数

- [MInputDriver minput_gui_driver](#)
ウィンドウシステムの内部入力メソッド用入力ドライバ.
- [MSymbol Mxim](#)
"xim"を名前として持つシンボル.

4.24 input.c ファイル

関数

- [MInputMethod * mdebug_dump_im \(MInputMethod *im, int indent\)](#)
入力メソッドをダンプする.

関数

- [MInputMethod * minput_open_im \(MSymbol language, MSymbol name, void *arg\)](#)
入力メソッドをオープンする.
- [void minput_close_im \(MInputMethod *im\)](#)
入力メソッドをクローズする.
- [MInputContext * minput_create_ic \(MInputMethod *im, void *arg\)](#)
入力コンテキストを生成する.
- [void minput_destroy_ic \(MInputContext *ic\)](#)
入力コンテキストを破壊する.
- [int minput_filter \(MInputContext *ic, MSymbol key, void *arg\)](#)
入力キーをフィルタする.
- [int minput_lookup \(MInputContext *ic, MSymbol key, void *arg, MText *mt\)](#)
入力コンテキスト中のテキストを探す.
- [void minput_set_spot \(MInputContext *ic, int x, int y, int ascent, int descent, int fontsize, MText *mt, int pos\)](#)
入力コンテキストのスポットを設定する.
- [void minput_toggle \(MInputContext *ic\)](#)
入力メソッドを切替える.
- [void minput_reset_ic \(MInputContext *ic\)](#)
入力コンテキストをリセットする.
- [MPList * minput_get_title_icon \(MSymbol language, MSymbol name\)](#)
入力メソッドのタイトルとアイコン用ファイル名を得る.
- [MText * minput_get_description \(MSymbol language, MSymbol name\)](#)
入力メソッドの説明テキストを得る.
- [MPList * minput_get_command \(MSymbol language, MSymbol name, MSymbol command\)](#)
- [int minput_config_command \(MSymbol language, MSymbol name, MSymbol command, MPList *keyseqlist\)](#)
- [MPList * minput_get_variable \(MSymbol language, MSymbol name, MSymbol variable\)](#)
- [int minput_config_variable \(MSymbol language, MSymbol name, MSymbol variable, MPList *value\)](#)
入力メソッドの変数の値を設定する.
- [char * minput_config_file \(\)](#)
ユーザ毎のカスタマイズファイルの名前を得る.
- [int minput_save_config \(void\)](#)
設定をユーザ毎のカスタマイズファイルに保存する.
- [MPList * minput_list \(MSymbol language\)](#)

Obsolete な関数

- `MPlist * minput_get_variables` (`MSymbol language`, `MSymbol name`)
- `int minput_set_variable` (`MSymbol language`, `MSymbol name`, `MSymbol variable`, `void *value`)
入力メソッド変数の初期値を設定する.
- `MPlist * minput_get_commands` (`MSymbol language`, `MSymbol name`)
入力メソッドのコマンドに関する情報を得る.
- `int minput_assign_command_keys` (`MSymbol language`, `MSymbol name`, `MSymbol command`, `MPlist *keyseq`)
入力メソッドコマンドにキーシーケンスを割り当てる.
- `MPlist * minput_parse_lm_names` (`MText *mt`)
- `int minput_callback` (`MInputContext *ic`, `MSymbol command`)

変数

- `MSymbol Minput_method`
"input-method" を名前として持つシンボル.
- `MInputDriver minput_default_driver`
内部入力メソッド用デフォルトドライバ.
- `MInputDriver * minput_driver`
内部入力メソッド用ドライバ.
- `MSymbol Minput_driver`

変数：コールバックコマンド用定義済みシンボル.

入力メソッドドライバのコールバック関数において `COMMAND` 引数として用いられる定義済みシンボル (`MInputDriver::callback_list` 参照).

ほとんどは追加の引数を必要としないし値を返さないが、以下は例外である。

`Minput_get_surrounding_text`: このコマンドに割り当てられたコールバック関数が呼ばれた際には、`MInputContext::plist` の第一要素はキーとして `::MInteger` をとり、その値はサラウンディングテキストのうちどの部分を取って来るかを指定する。値が正であれば、現在のカーソル位置に続く 値の個数分の文字を取る。負であれば、カーソル位置に先行する値の絶対 値分の文字を取る。現在サラウンドテキストがサポートされているかどうかを知りたいだけであれば、この値はゼロでも良い。

サラウンディングテキストがサポートされていれば、コールバック関数は この要素のキーを `Mtext` に、値を取り込んだ `M-text` に設定しなくてはならない。もしテキストの長さが充分でなければ、この `M-text` の長さは要求されている文字数より短くて良い。最悪の場合 0 でもよいし、アプリケーション側で必要で効率的だと思えば長くても良い。

サラウンディングテキストがサポートされていなければ、コールバック関数は `MInputContext::plist` の第一要素を変更してはならない。

`Minput_delete_surrounding_text`: このコマンドに割り当てられたコールバック関数が呼ばれた際には、`::MInputContext::plist` の第一要素は、キーとして `::MInteger` をとり、値は削除すべきサラウンディングテキストを `Minput_get_surrounding_text` と同様のやり方で指定する。コールバック関数は指定されたテキストを削除しなければならない。また `MInputContext::plist` を変えてはならない。

- `MSymbol Minput_preedit_start`
- `MSymbol Minput_preedit_done`
- `MSymbol Minput_preedit_draw`
- `MSymbol Minput_status_start`
- `MSymbol Minput_status_done`
- `MSymbol Minput_status_draw`
- `MSymbol Minput_candidates_start`
- `MSymbol Minput_candidates_done`
- `MSymbol Minput_candidates_draw`
- `MSymbol Minput_set_spot`
- `MSymbol Minput_toggle`

- [MSymbol Minput_reset](#)
- [MSymbol Minput_get_surrounding_text](#)
- [MSymbol Minput_delete_surrounding_text](#)

変数: 特別な入力イベント用定義済みシンボル.

[minput.filter\(\)](#) の KEY 引数として用いられる定義済みシンボル。

- [MSymbol Minput_focus_out](#)
- [MSymbol Minput_focus_in](#)
- [MSymbol Minput_focus_move](#)

変数: 入力メソッド情報用定義済みシンボル.

- [MSymbol Minherited](#)
- [MSymbol Mcustomized](#)
- [MSymbol Mconfigured](#)

4.25 input.h ファイル

データ構造

- struct [MInputMethodInfo](#)
- struct [MInputContextInfo](#)

マクロ定義

- [#define MINPUT_KEY_SHIFT_MODIFIER](#) (1 << 0)
- [#define MINPUT_KEY_CONTROL_MODIFIER](#) (1 << 1)
- [#define MINPUT_KEY_META_MODIFIER](#) (1 << 2)
- [#define MINPUT_KEY_ALT_MODIFIER](#) (1 << 3)
- [#define MINPUT_KEY_SUPER_MODIFIER](#) (1 << 4)
- [#define MINPUT_KEY_HYPER_MODIFIER](#) (1 << 5)
- [#define MINPUT_KEY_ALTGR_MODIFIER](#) (1 << 6)

型定義

- typedef struct [MIMState](#) [MIMState](#)
- typedef struct [MIMMap](#) [MIMMap](#)
- typedef struct [MIMInputStack](#) [MIMInputStack](#)

関数

- [MSymbol minput_char_to_key](#) (int c)

4.25.1 マクロ定義詳解

4.25.1.1 MINPUT_KEY_SHIFT_MODIFIER

```
#define MINPUT_KEY_SHIFT_MODIFIER (1 << 0)
```

4.25.1.2 MINPUT_KEY_CONTROL_MODIFIER

```
#define MINPUT_KEY_CONTROL_MODIFIER (1 << 1)
```

4.25.1.3 MINPUT_KEY_META_MODIFIER

```
#define MINPUT_KEY_META_MODIFIER (1 << 2)
```

4.25.1.4 MINPUT_KEY_ALT_MODIFIER

```
#define MINPUT_KEY_ALT_MODIFIER (1 << 3)
```

4.25.1.5 MINPUT_KEY_SUPER_MODIFIER

```
#define MINPUT_KEY_SUPER_MODIFIER (1 << 4)
```

4.25.1.6 MINPUT_KEY_HYPER_MODIFIER

```
#define MINPUT_KEY_HYPER_MODIFIER (1 << 5)
```

4.25.1.7 MINPUT_KEY_ALTGR_MODIFIER

```
#define MINPUT_KEY_ALTGR_MODIFIER (1 << 6)
```

4.25.2 型定義詳解

4.25.2.1 MIMState

```
typedef struct MIMState MIMState
```

4.25.2.2 MIMMap

```
typedef struct MIMMap MIMMap
```

4.25.2.3 MIMInputStack

```
typedef struct MIMInputStack MIMInputStack
```

4.25.3 関数詳解

4.25.3.1 minput_char_to_key()

```
MSymbol minput_char_to_key (  
    int c )
```

4.26 internal-flt.h ファイル

マクロ定義

- #define MAKE_COMBINING_CODE(base_y, base_x, add_y, add_x, off_y, off_x)
- #define COMBINING_CODE_OFF_Y(code) (((code) >> 16) & 0xFF) - 128)
- #define COMBINING_CODE_OFF_X(code) (((code) >> 8) & 0xFF) - 128)
- #define COMBINING_CODE_BASE_X(code) (((code) >> 6) & 0x3)
- #define COMBINING_CODE_BASE_Y(code) (((code) >> 4) & 0x3)
- #define COMBINING_CODE_ADD_X(code) (((code) >> 2) & 0x3)
- #define COMBINING_CODE_ADD_Y(code) ((code) & 0x3)
- #define PACK_OTF_TAG(TAG)

変数

- MSymbol Mcombining

4.26.1 マクロ定義詳解

4.26.1.1 MAKE_COMBINING_CODE

```
#define MAKE_COMBINING_CODE(
    base_y,
    base_x,
    add_y,
    add_x,
    off_y,
    off_x )
```

値:

```
((off_y) << 16)
| ((off_x) << 8)
| ((base_x) << 6)
| ((base_y) << 4)
| ((add_x) << 2)
| (add_y))
```

4.26.1.2 COMBINING_CODE_OFF_Y

```
#define COMBINING_CODE_OFF_Y(
    code ) (((code) >> 16) & 0xFF) - 128)
```

4.26.1.3 COMBINING_CODE_OFF_X

```
#define COMBINING_CODE_OFF_X(
    code ) (((code) >> 8) & 0xFF) - 128)
```

4.26.1.4 COMBINING_CODE_BASE_X

```
#define COMBINING_CODE_BASE_X(
    code ) (((code) >> 6) & 0x3)
```

4.26.1.5 COMBINING_CODE_BASE_Y

```
#define COMBINING_CODE_BASE_Y(
    code ) (((code) >> 4) & 0x3)
```


4.26.1.6 COMBINING_CODE_ADD_X

```
#define COMBINING_CODE_ADD_X(  
    code ) (((code) >> 2) & 0x3)
```

4.26.1.7 COMBINING_CODE_ADD_Y

```
#define COMBINING_CODE_ADD_Y(  
    code ) ((code) & 0x3)
```

4.26.1.8 PACK_OTF_TAG

```
#define PACK_OTF_TAG(  
    TAG )
```

値:

```
(( ( (TAG) & 0x7F000000) >> 3) \  
 | ((TAG) & 0x7F0000) >> 2) \  
 | ((TAG) & 0x7F00) >> 1) \  
 | ((TAG) & 0x7F))
```

4.26.2 変数詳解

4.26.2.1 Mcombining

[MSymbol](#) Mcombining [extern]

4.27 internal-gui.h ファイル

データ構造

- struct [MFrame](#)
フレームの型宣言.
- struct [MGlyph](#)
- struct [MGlyphString](#)
- struct [MDrawPoint](#)
- struct [MDeviceDriver](#)

マクロ定義

- #define [M_CHECK_WRITABLE](#)(frame, err, ret)
- #define [M_CHECK_READABLE](#)(frame, err, ret)
- #define [MGLYPH](#)(idx) (gstring->glyphs + ((idx) >= 0 ? (idx) : (gstring->used + (idx))))
- #define [GLYPH_INDEX](#)(g) ((g) - gstring->glyphs)
- #define [INIT_GLYPH](#)(g) (memset (&(g), 0, sizeof (g)))
- #define [APPEND_GLYPH](#)(gstring, g) [MLIST_APPEND1](#) ((gstring), glyphs, (g), [MERROR_DRAW](#))
- #define [INSERT_GLYPH](#)(gstring, at, g)
- #define [DELETE_GLYPH](#)(gstring, at)
- #define [REPLACE_GLYPHS](#)(gstring, from, to, len)

型定義

- typedef struct [MRealizedFontset](#) [MRealizedFontset](#)

列挙型

- enum [MDeviceType](#) {
 [MDEVICE_SUPPORT_OUTPUT](#) = 1 ,
 [MDEVICE_SUPPORT_INPUT](#) = 2 }
- enum [glyph_type](#) {
 [GLYPH_CHAR](#) ,
 [GLYPH_SPACE](#) ,
 [GLYPH_PAD](#) ,
 [GLYPH_BOX](#) ,
 [GLYPH_ANCHOR](#) ,
 [GLYPH_TYPE_MAX](#) }
- enum [glyph_category](#) {
 [GLYPH_CATEGORY_NORMAL](#) ,
 [GLYPH_CATEGORY_MODIFIER](#) ,
 [GLYPH_CATEGORY_FORMATTER](#) }

関数

- int [mfont__init](#) ()
- void [mfont__fini](#) ()
- int [mface__init](#) ()
- void [mface__fini](#) ()
- int [mdraw__init](#) ()
- void [mdraw__fini](#) ()
- int [mfont__fontset_init](#) ()
- void [mfont__fontset_fini](#) ()
- int [minput__win_init](#) ()
- void [minput__win_fini](#) ()

変数

- [MSymbol Mlatin](#)
- [MSymbol Mgd](#)

4.27.1 マクロ定義詳解

4.27.1.1 M_CHECK_WRITABLE

```
#define M_CHECK_WRITABLE(  
    frame,  
    err,  
    ret )
```

値:

```
do {  
    if (! ((frame)->device_type & MDEVICE_SUPPORT_OUTPUT)) \  
        MERROR ((err), (ret));  
} while (0)
```

4.27.1.2 M_CHECK_READABLE

```
#define M_CHECK_READABLE(  
    frame,  
    err,  
    ret )
```

値:

```
do {  
    if (! ((frame)->device_type & MDEVICE_SUPPORT_INPUT)) \  
        MERROR ((err), (ret));  
} while (0)
```

4.27.1.3 MGLYPH

```
#define MGLYPH(  
    idx )  (gstring->glyphs + ((idx) >= 0 ?  (idx) :  (gstring->used + (idx))))
```

4.27.1.4 GLYPH_INDEX

```
#define GLYPH_INDEX(  
    g )  ((g) - gstring->glyphs)
```

4.27.1.5 INIT_GLYPH

```
#define INIT_GLYPH(
    g )    (memset (&(g), 0, sizeof (g)))
```

4.27.1.6 APPEND_GLYPH

```
#define APPEND_GLYPH(
    gstring,
    g )    MLIST_APPEND1 ((gstring), glyphs, (g), MERROR_DRAW)
```

4.27.1.7 INSERT_GLYPH

```
#define INSERT_GLYPH(
    gstring,
    at,
    g )
```

値:

```
do {
    MLIST_INSERT1 ((gstring), glyphs, (at), 1, MERROR_DRAW); \
    (gstring)->glyphs[at] = g; \
} while (0)
```

4.27.1.8 DELETE_GLYPH

```
#define DELETE_GLYPH(
    gstring,
    at )
```

値:

```
do {
    MLIST_DELETE1 (gstring, glyphs, at, 1); \
} while (0)
```

4.27.1.9 REPLACE_GLYPHS

```
#define REPLACE_GLYPHS(
    gstring,
    from,
    to,
    len )
```

値:

```
do {
    int newlen = (gstring)->used - (from); \
    int diff = newlen - (len); \
    if (diff < 0) \
        MLIST_DELETE1 (gstring, glyphs, (to) + newlen, -diff); \
    else if (diff > 0) \
        MLIST_INSERT1 ((gstring), glyphs, (to) + (len), diff, MERROR_DRAW); \
    memmove ((gstring)->glyphs + to, (gstring)->glyphs + (from + diff), \
        (sizeof (MGlyph)) * newlen); \
    (gstring)->used -= newlen; \
} while (0)
```

4.27.2 型定義詳解

4.27.2.1 MRealizedFontset

```
typedef struct MRealizedFontset MRealizedFontset
```

4.27.3 列挙型詳解

4.27.3.1 MDeviceType

```
enum MDeviceType
```

列挙値

MDEVICE_SUPPORT_OUTPUT	
MDEVICE_SUPPORT_INPUT	

4.27.3.2 glyph_type

```
enum glyph_type
```

列挙値

GLYPH_CHAR	
GLYPH_SPACE	
GLYPH_PAD	
GLYPH_BOX	
GLYPH_ANCHOR	
GLYPH_TYPE_MAX	

4.27.3.3 glyph_category

```
enum glyph_category
```

列挙値

GLYPH_CATEGORY_NORMAL	
GLYPH_CATEGORY_MODIFIER	
GLYPH_CATEGORY_FORMATTER	

4.27.4 関数詳解

4.27.4.1 mfont_init()

```
int mfont_init ( )
```

4.27.4.2 mfont_fini()

```
void mfont_fini ( )
```

4.27.4.3 mface_init()

```
int mface_init ( )
```

4.27.4.4 mface_fini()

```
void mface_fini ( )
```

4.27.4.5 mdraw_init()

```
int mdraw_init ( )
```

4.27.4.6 mdraw_fini()

```
void mdraw_fini ( )
```

4.27.4.7 mfont_fontset_init()

```
int mfont_fontset_init ( )
```

4.27.4.8 mfont_fontset_fini()

```
void mfont_fontset_fini ( )
```

4.27.4.9 minput_win_init()

```
int minput_win_init ( )
```

4.27.4.10 minput_win_fini()

```
void minput_win_fini ( )
```

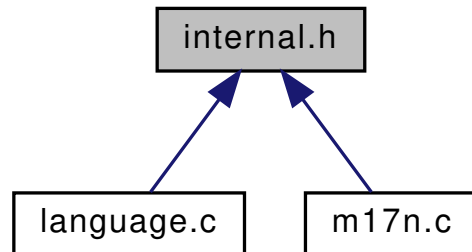
4.27.5 変数詳解

4.27.5.1 Mlatin

```
MSymbol Mlatin [extern]
```

4.28 internal.h ファイル

被依存関係図:



データ構造

- struct [M17NObjectRecord](#)
- struct [M17NObject](#)
- struct [M17NObjectArray](#)
- struct [MText](#)

[MText](#) の型宣言.

マクロ定義

- #define [_\(String\)](#) (String)
- #define [MERROR](#)(err, ret)
- #define [MERROR_GOTO](#)(err, label)
- #define [MWARNING](#)(err)
- #define [MFATAL](#)(err)
- #define [MFAILP](#)(cond) ((cond) ? 0 : [mdebug_hook](#)())
- #define [M_CHECK_CHAR](#)(c, ret)
- #define [MEMORY_FULL](#)(err)
- #define [MTABLE_MALLOC](#)(p, size, err)
- #define [MTABLE_CALLOC](#)(p, size, err)
- #define [MTABLE_CALLOC_SAFE](#)(p, size) ((p) = (void *) calloc (sizeof (*p)), (size)))
- #define [MTABLE_REALLOC](#)(p, size, err)
- #define [MTABLE_ALLOCA](#)(p, size, err)
- #define [MSTRUCT_MALLOC](#)(p, err)
- #define [MSTRUCT_CALLOC](#)(p, err) [MTABLE_CALLOC](#) ((p), 1, (err))
- #define [MSTRUCT_CALLOC_SAFE](#)(p) [MTABLE_CALLOC_SAFE](#) ((p), 1)
- #define [USE_SAFE_ALLOCA](#) int sa_must_free = 0, sa_size = 0
- #define [SAFE_ALLOCA](#)(P, SIZE)
- #define [SAFE_FREE](#)(P)
- #define [MLIST_RESET](#)(list) ((list)->used = 0)
- #define [MLIST_INIT1](#)(list, mem, increment)
- #define [MLIST_APPEND1](#)(list, mem, elt, err)
- #define [MLIST_PREPEND1](#)(list, mem, elt, err)
- #define [MLIST_INSERT1](#)(list, mem, idx, len, err)
- #define [MLIST_DELETE1](#)(list, mem, idx, len)

- #define `MLIST_COPY1`(list0, list1, mem, err)
- #define `MLIST_FREE1`(list, mem)
- #define `M17N_OBJECT`(object, free_func, err)
- #define `M17N_OBJECT_REF`(object)
- #define `M17N_OBJECT_REF_NTIMES`(object, n)
- #define `M17N_OBJECT_UNREF`(object)
- #define `M17N_OBJECT_ADD_ARRAY`(array, name)
- #define `M17N_OBJECT_REGISTER`(array, object)
- #define `M17N_OBJECT_UNREGISTER`(array, object)
- #define `M_CHECK_POS`(mt, pos, ret)
- #define `M_CHECK_POS_X`(mt, pos, ret)
- #define `M_CHECK_RANGE`(mt, from, to, ret, ret2)
- #define `M_CHECK_RANGE_X`(mt, from, to, ret)
- #define `M_CHECK_POS_NCHARS`(mt, pos, nchars, ret, ret2)
- #define `MTEXT_READ_ONLY_P`(mt) ((mt)->allocated < 0)
- #define `M_CHECK_READONLY`(mt, ret)
- #define `mtext_nchars`(mt) ((mt)->nchars)
- #define `mtext_nbytes`(mt) ((mt)->nbytes)
- #define `mtext_allocated`(mt) ((mt)->allocated)
- #define `mtext_reset`(mt) (`mtext_del` ((mt), 0, (mt)->nchars))
- #define `MDEBUG_FLAG`() `mdebug_flags`[`mdebug_flag`]
- #define `MDEBUG_PRINT0`(FPRINTF)
- #define `MDEBUG_PRINT`(msg) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, "%s", (msg)))
- #define `MDEBUG_PRINT1`(fmt, arg) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, (fmt), (arg)))
- #define `MDEBUG_PRINT2`(fmt, arg1, arg2) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, (fmt), (arg1), (arg2)))
- #define `MDEBUG_PRINT3`(fmt, arg1, arg2, arg3) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, (fmt), (arg1), (arg2), (arg3)))
- #define `MDEBUG_PRINT4`(fmt, arg1, arg2, arg3, arg4) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, (fmt), (arg1), (arg2), (arg3), (arg4)))
- #define `MDEBUG_PRINT5`(fmt, arg1, arg2, arg3, arg4, arg5) `MDEBUG_PRINT0` (fprintf (`mdebug_output`, (fmt), (arg1), (arg2), (arg3), (arg4), (arg5)))
- #define `MDEBUG_DUMP`(prefix, postfix, call)
- #define `MDEBUG_PUSH_TIME`()
- #define `MDEBUG_POP_TIME`()
- #define `MDEBUG_PRINT_TIME`(tag, ARG_LIST)
- #define `SWAP_16`(c) (((c) >> 8) | (((c) & 0xFF) << 8))
- #define `SWAP_32`(c)

列挙型

- enum `MTextCoverage` {
`MTEXT_COVERAGE_ASCII` ,
`MTEXT_COVERAGE_UNICODE` ,
`MTEXT_COVERAGE_FULL` }
- enum `MDebugFlag` {
`MDEBUG_INIT` ,
`MDEBUG_FINI` ,
`MDEBUG_CHARSET` ,
`MDEBUG_CODING` ,
`MDEBUG_DATABASE` ,
`MDEBUG_FONT` ,

```

MDEBUG.FLT ,
MDEBUG.FONTSET ,
MDEBUG.INPUT ,
MDEBUG.ALL ,
MDEBUG.MAX = MDEBUG.ALL }

```

関数

- int [mdebug_hook](#) ()
エラーの際に呼ばれるフック関数.
- void [mdebug_add_object_array](#) (M17NObjectArray *array, char *name)
- void [mdebug_register_object](#) (M17NObjectArray *array, void *object)
- void [mdebug_unregister_object](#) (M17NObjectArray *array, void *object)
- void [mdebug_push_time](#) ()
- void [mdebug_pop_time](#) ()
- void [mdebug_print_time](#) ()
- int [msymbol_init](#) ()
- void [msymbol_fini](#) ()
- int [mplist_init](#) ()
- void [mplist_fini](#) ()
- int [mtext_init](#) ()
- void [mtext_fini](#) ()
- int [mtext_prop_init](#) ()
- void [mtext_prop_fini](#) ()
- int [mchartable_init](#) ()
- void [mchartable_fini](#) ()
- int [mcharset_init](#) ()
- void [mcharset_fini](#) ()
- int [mcoding_init](#) ()
- void [mcoding_fini](#) ()
- int [mdatabase_init](#) (void)
- void [mdatabase_fini](#) (void)
- int [mchar_init](#) ()
- void [mchar_fini](#) ()
- int [mlang_init](#) ()
- void [mlang_fini](#) ()
- int [mlocale_init](#) ()
- void [mlocale_fini](#) ()
- int [minput_init](#) ()
- void [minput_fini](#) ()

変数

- int [m17n_core_initialized](#)
- int [m17n_shell_initialized](#)
- int [m17n_gui_initialized](#)
- int [mdebug_flags](#) [MDEBUG.MAX]
- FILE * [mdebug_output](#)

4.28.1 マクロ定義詳解

4.28.1.1 _

```
#define _(  
    String ) (String)
```

4.28.1.2 MERROR

```
#define MERROR(  
    err,  
    ret )
```

値:

```
do {  
    error_code = (err);  
    mdebug_hook();  
    return (ret);  
} while (0)
```

4.28.1.3 MERROR_GOTO

```
#define MERROR_GOTO(  
    err,  
    label )
```

値:

```
do {  
    if ((err))  
        error_code = (err);  
    mdebug_hook();  
    goto label;  
} while (0)
```

4.28.1.4 MWARNING

```
#define MWARNING(  
    err )
```

値:

```
do {  
    mdebug_hook();  
    goto warning;  
} while (0)
```

4.28.1.5 MFATAL

```
#define MFATAL(
    err )
```

値:

```
do {
    mdebug_hook(); \
    exit (err); \
} while (0)
```

4.28.1.6 MFAILP

```
#define MFAILP(
    cond ) ((cond) ? 0 : mdebug_hook())
```

4.28.1.7 M_CHECK_CHAR

```
#define M_CHECK_CHAR(
    c,
    ret )
```

値:

```
if ((c) < 0 || (c) > MCHAR_MAX) \
    MERROR (MERROR_CHAR, (ret)); \
else
```

4.28.1.8 MEMORY_FULL

```
#define MEMORY_FULL(
    err )
```

値:

```
do {
    (*m17n_memory_full_handler) (err); \
    exit (err); \
} while (0)
```

4.28.1.9 MTABLE_MALLOC

```
#define MTABLE_MALLOC(
    p,
    size,
    err )
```

値:

```
do {
    if (! ((p) = (void *) malloc (sizeof (*p) * (size)))) \
        MEMORY_FULL (err); \
} while (0)
```

4.28.1.10 MTABLE_CALLOC

```
#define MTABLE_CALLOC(
    p,
    size,
    err )
```

値:

```
do {
    if (! ((p) = (void *) calloc (sizeof (*(p)), size))) \
        MEMORY_FULL (err);
} while (0)
```

4.28.1.11 MTABLE_CALLOC_SAFE

```
#define MTABLE_CALLOC_SAFE(
    p,
    size ) ((p) = (void *) calloc (sizeof (*(p)), (size)))
```

4.28.1.12 MTABLE_REALLOC

```
#define MTABLE_REALLOC(
    p,
    size,
    err )
```

値:

```
do {
    if (! ((p) = (void *) realloc ((p), sizeof (*(p)) * (size)))) \
        MEMORY_FULL (err);
} while (0)
```

4.28.1.13 MTABLE_ALLOCA

```
#define MTABLE_ALLOCA(
    p,
    size,
    err )
```

値:

```
do {
    int allocasize = sizeof (*(p)) * (size); \
    if (! ((p) = (void *) alloca (allocasize))) \
        MEMORY_FULL (err);
    memset ((p), 0, allocasize);
} while (0)
```

4.28.1.14 MSTRUCT_MALLOC

```
#define MSTRUCT_MALLOC(
    p,
    err )
```

値:

```
do {
    if (! ((p) = (void *) malloc (sizeof (*(p)))) ) \
        MEMORY_FULL (err);
} while (0)
```

4.28.1.15 MSTRUCT_CALLOC

```
#define MSTRUCT_CALLOC(
    p,
    err ) MTABLE_CALLOC ((p), 1, (err))
```

4.28.1.16 MSTRUCT_CALLOC_SAFE

```
#define MSTRUCT_CALLOC_SAFE(
    p ) MTABLE_CALLOC_SAFE ((p), 1)
```

4.28.1.17 USE_SAFE_ALLOCA

```
#define USE_SAFE_ALLOCA int sa_must_free = 0, sa_size = 0
```

4.28.1.18 SAFE_ALLOCA

```
#define SAFE_ALLOCA(
    P,
    SIZE )
```

値:

```
do {
    if (sa_size < (SIZE)) \
        {
            if (sa_must_free) \
                (P) = realloc ((P), (SIZE)); \
            else
                {
                    (P) = alloca ((SIZE)); \
                    if (! (P)) \
                        {
                            (P) = malloc (SIZE); \
                            sa_must_free = 1; \
                        }
                }
            if (! (P)) \
                MEMORY_FULL (1); \
            sa_size = (SIZE); \
        }
} while (0)
```

4.28.1.19 SAFE_FREE

```
#define SAFE_FREE(
    P )
```

値:

```
do {
    if (sa_must_free && sa_size > 0) \
    {
        free ((P));
        sa_must_free = sa_size = 0; \
    }
} while (0)
```

4.28.1.20 MLIST_RESET

```
#define MLIST_RESET(
    list ) ((list)->used = 0)
```

4.28.1.21 MLIST_INIT1

```
#define MLIST_INIT1(
    list,
    mem,
    increment )
```

値:

```
do {
    (list)->size = (list)->used = 0; \
    (list)->inc = (increment); \
    (list)->mem = NULL; \
} while (0)
```

4.28.1.22 MLIST_APPEND1

```
#define MLIST_APPEND1(
    list,
    mem,
    elt,
    err )
```

値:

```
do {
    if ((list)->inc <= 0) \
        mdebug_hook(); \
    if ((list)->size == (list)->used) \
    {
        (list)->size += (list)->inc; \
        MTABLE_REALLOC ((list)->mem, (list)->size, (err)); \
    }
    (list)->mem[(list)->used++] = (elt); \
} while (0)
```

4.28.1.23 MLIST_PREPEND1

```
#define MLIST_PREPEND1(
    list,
    mem,
    elt,
    err )
```

値:

```
do {
    if ((list)->inc <= 0)
        mdebug_hook();
    if ((list)->size == (list)->used)
    {
        (list)->size += (list)->inc;
        MTABLE_REALLOC ((list)->mem, (list)->size, (err));
    }
    memmove ((list)->mem + 1, (list)->mem,
        sizeof *((list)->mem) * ((list)->used));
    (list)->mem[0] = (elt);
    (list)->used++;
} while (0)
```

4.28.1.24 MLIST_INSERT1

```
#define MLIST_INSERT1(
    list,
    mem,
    idx,
    len,
    err )
```

値:

```
do {
    while ((list)->used + (len) > (list)->size)
    {
        (list)->size += (list)->inc;
        MTABLE_REALLOC ((list)->mem, (list)->size, (err));
    }
    memmove ((list)->mem + ((idx) + (len)), (list)->mem + (idx),
        (sizeof *((list)->mem)) * ((list)->used - (idx)));
    (list)->used += (len);
} while (0)
```

4.28.1.25 MLIST_DELETE1

```
#define MLIST_DELETE1(
    list,
    mem,
    idx,
    len )
```

値:

```
do {
    memmove ((list)->mem + (idx), (list)->mem + (idx) + (len),
        (sizeof *((list)->mem)) * ((list)->used - (idx) - (len)));
    (list)->used -= (len);
} while (0)
```


4.28.1.26 MLIST_COPY1

```
#define MLIST_COPY1(
    list0,
    list1,
    mem,
    err )
```

値:

```
do {
    (list0)->size = (list0)->used = (list1)->used; \
    (list0)->inc = 1; \
    MTABLE_MALLOC ((list0)->mem, (list0)->used, (err)); \
    memcpy ((list0)->mem, (list1)->mem, \
        (sizeof (list0)->mem) * (list0)->used); \
} while (0)
```

4.28.1.27 MLIST_FREE1

```
#define MLIST_FREE1(
    list,
    mem )
```

値:

```
if ((list)->size) \
{
    free ((list)->mem); \
    (list)->mem = NULL; \
    (list)->size = (list)->used = 0; \
} \
else
```

4.28.1.28 M17N_OBJECT

```
#define M17N_OBJECT(
    object,
    free_func,
    err )
```

値:

```
do {
    MSTRUCT_CALLOC ((object), (err)); \
    ((M17NObject *) (object))->ref_count = 1; \
    ((M17NObject *) (object))->u.freer = free_func; \
} while (0)
```

4.28.1.29 M17N_OBJECT_REF

```
#define M17N_OBJECT_REF(
    object )
```

値:

```
do {
    if (((M17NObject *) (object))->ref_count_extended) \
        m17n_object_ref (object); \
    else if (((M17NObject *) (object))->ref_count > 0) \
    { \
        ((M17NObject *) (object))->ref_count++; \
        if (! ((M17NObject *) (object))->ref_count) \
        { \
            ((M17NObject *) (object))->ref_count--; \
            m17n_object_ref (object); \
        } \
    } \
} while (0)
```

4.28.1.30 M17N_OBJECT_REF_NTICES

```
#define M17N_OBJECT_REF_NTICES(
    object,
    n )
```

値:

```
do {
    int i; \
    if (((M17NObject *) (object))->ref_count_extended) \
        for (i = 0; i < n; i++) \
            m17n_object_ref (object); \
    else if (((M17NObject *) (object))->ref_count > 0) \
    { \
        int orig_ref_count = ((M17NObject *) (object))->ref_count; \
        for (i = 0; i < n; i++) \
            if (! ++((M17NObject *) (object))->ref_count) \
            { \
                ((M17NObject *) (object))->ref_count = orig_ref_count; \
                for (i = 0; i < n; i++) \
                    m17n_object_ref (object); \
            } \
    } \
} while (0)
```

4.28.1.31 M17N_OBJECT_UNREF

```
#define M17N_OBJECT_UNREF(
    object )
```

値:

```
do {
    if (object) \
    { \
        if (((M17NObject *) (object))->ref_count_extended \
            || mdebug_flags[MDEBUG_FINI]) \
        { \
            if (m17n_object_unref (object) == 0) \
                (object) = NULL; \
        } \
        else if (((M17NObject *) (object))->ref_count == 0) \
    }
```

```

        break;
    else
    {
        ((M17NObject *) (object))->ref_count--;
        if (((M17NObject *) (object))->ref_count == 0)
        {
            if (((M17NObject *) (object))->u.freer)
            {
                ((M17NObject *) (object))->u.freer (object);
            }
            else
            {
                free (object);
                (object) = NULL;
            }
        }
    }
} while (0)

```

4.28.1.32 M17N_OBJECT_ADD_ARRAY

```

#define M17N_OBJECT_ADD_ARRAY(
    array,
    name )

```

値:

```

    if (mdebug_flags[MDEBUG_FINI])
        mdebug_add_object_array (&array, name);
    else

```

4.28.1.33 M17N_OBJECT_REGISTER

```

#define M17N_OBJECT_REGISTER(
    array,
    object )

```

値:

```

    if (mdebug_flags[MDEBUG_FINI])
        mdebug_register_object (&array, object);
    else

```

4.28.1.34 M17N_OBJECT_UNREGISTER

```

#define M17N_OBJECT_UNREGISTER(
    array,
    object )

```

値:

```

    if (mdebug_flags[MDEBUG_FINI])
        mdebug_unregister_object (&array, object);
    else

```

4.28.1.35 M_CHECK_POS

```
#define M_CHECK_POS(
    mt,
    pos,
    ret )
```

値:

```
do {
    if ((pos) < 0 || (pos) >= (mt)->nchars) \
        MERROR (MERROR_RANGE, (ret)); \
} while (0)
```

4.28.1.36 M_CHECK_POS_X

```
#define M_CHECK_POS_X(
    mt,
    pos,
    ret )
```

値:

```
do {
    if ((pos) < 0 || (pos) > (mt)->nchars) \
        MERROR (MERROR_RANGE, (ret)); \
} while (0)
```

4.28.1.37 M_CHECK_RANGE

```
#define M_CHECK_RANGE(
    mt,
    from,
    to,
    ret,
    ret2 )
```

値:

```
do {
    if ((from) < 0 || (to) < (from) || (to) > (mt)->nchars) \
        MERROR (MERROR_RANGE, (ret)); \
    if ((from) == (to)) \
        return (ret2); \
} while (0)
```

4.28.1.38 M_CHECK_RANGE_X

```
#define M_CHECK_RANGE_X(
    mt,
    from,
    to,
    ret )
```

値:

```
do {
    if ((from) < 0 || (to) < (from) || (to) > (mt)->nchars) \
        MERROR (MERROR_RANGE, (ret)); \
} while (0)
```

4.28.1.39 M_CHECK_POS_NCHARS

```
#define M_CHECK_POS_NCHARS(
    mt,
    pos,
    nchars,
    ret,
    ret2 )
```

値:

```
do {
    int to = (pos) + (nchars); \
    M_CHECK_RANGE ((mt), (pos), (to), (ret), (ret2)); \
} while (0)
```

4.28.1.40 MTEXT_READ_ONLY_P

```
#define MTEXT_READ_ONLY_P(
    mt ) ((mt)->allocated < 0)
```

4.28.1.41 M_CHECK_READONLY

```
#define M_CHECK_READONLY(
    mt,
    ret )
```

値:

```
do {
    if ((mt)->allocated < 0) \
        MERROR (MERROR_MTEXT, (ret)); \
} while (0)
```

4.28.1.42 mtext_nchars

```
#define mtext_nchars(
    mt ) ((mt)->nchars)
```

4.28.1.43 mtext_nbytes

```
#define mtext_nbytes(
    mt ) ((mt)->nbytes)
```

4.28.1.44 mtext_allocated

```
#define mtext_allocated(
    mt ) ((mt)->allocated)
```

4.28.1.45 mtext_reset

```
#define mtext_reset(
    mt ) (mtext_del ((mt), 0, (mt)->nchars))
```

4.28.1.46 MDEBUG_FLAG

```
#define MDEBUG_FLAG( ) mdebug_flags[mdebug_flag]
```

4.28.1.47 MDEBUG_PRINT0

```
#define MDEBUG_PRINT0(
    FPRINTF )
```

値:

```
do {
    if (MDEBUG_FLAG())
    {
        FPRINTF;
        fflush (mdebug_output);
    }
} while (0)
```

4.28.1.48 MDEBUG_PRINT

```
#define MDEBUG_PRINT(
    msg ) MDEBUG_PRINT0 (fprintf (mdebug_output, "%s", (msg)))
```

4.28.1.49 MDEBUG_PRINT1

```
#define MDEBUG_PRINT1(
    fmt,
    arg ) MDEBUG_PRINT0 (fprintf (mdebug_output, (fmt), (arg)))
```

4.28.1.50 MDEBUG_PRINT2

```
#define MDEBUG_PRINT2(
    fmt,
    arg1,
    arg2 ) MDEBUG_PRINT0 (fprintf (mdebug_output, (fmt), (arg1), (arg2)))
```

4.28.1.51 MDEBUG_PRINT3

```
#define MDEBUG_PRINT3(
    fmt,
    arg1,
    arg2,
    arg3 ) MDEBUG_PRINT0 (fprintf (mdebug_output, (fmt), (arg1), (arg2), (arg3)))
```

4.28.1.52 MDEBUG_PRINT4

```
#define MDEBUG_PRINT4(
    fmt,
    arg1,
    arg2,
    arg3,
    arg4 ) MDEBUG_PRINT0 (fprintf (mdebug_output, (fmt), (arg1), (arg2), (arg3),
    (arg4)))
```

4.28.1.53 MDEBUG_PRINT5

```
#define MDEBUG_PRINT5(
    fmt,
    arg1,
    arg2,
    arg3,
    arg4,
    arg5 ) MDEBUG_PRINT0 (fprintf (mdebug_output, (fmt), (arg1), (arg2), (arg3),
    (arg4), (arg5)))
```

4.28.1.54 MDEBUG_DUMP

```
#define MDEBUG_DUMP(
    prefix,
    postfix,
    call )
```

値:

```
do {
    if (MDEBUG_FLAG())
    {
        fprintf (mdebug_output, "%s", prefix);
        call;
        fprintf (mdebug_output, "%s", postfix);
        fflush (mdebug_output);
    }
} while (0)
```

4.28.1.55 MDEBUG_PUSH_TIME

```
#define MDEBUG_PUSH_TIME( )
```

値:

```
do {
    if (MDEBUG_FLAG())
        mdebug_push_time();
} while (0)
```

4.28.1.56 MDEBUG_POP_TIME

```
#define MDEBUG_POP_TIME( )
```

値:

```
do {
    if (MDEBUG_FLAG())
        mdebug_pop_time();
} while (0)
```

4.28.1.57 MDEBUG_PRINT_TIME

```
#define MDEBUG_PRINT_TIME(
    tag,
    ARG_LIST )
```

値:

```
do {
    if (MDEBUG_FLAG())
    {
        fprintf (mdebug_output, " [%s] ", tag);
        mdebug_print_time();
        fprintf ARG_LIST;
        fprintf (mdebug_output, "\n");
    }
} while (0)
```

4.28.1.58 SWAP_16

```
#define SWAP_16(
    c ) ((c) >> 8) | (((c) & 0xFF) << 8)
```

4.28.1.59 SWAP_32

```
#define SWAP_32(
    c )
```

値:

```
((c) >> 24) | (((c) >> 8) & 0xFF00) |
| (((c) & 0xFF00) << 8) | (((c) & 0xFF) << 24))
```


4.28.2 列挙型詳解

4.28.2.1 MTextCoverage

enum `MTextCoverage`

列挙値

MTEXT_COVERAGE_ASCII	
MTEXT_COVERAGE_UNICODE	
MTEXT_COVERAGE_FULL	

4.28.2.2 MDebugFlag

enum `MDebugFlag`

列挙値

MDEBUG_INIT	
MDEBUG_FINI	
MDEBUG_CHARSET	
MDEBUG_CODING	
MDEBUG_DATABASE	
MDEBUG_FONT	
MDEBUG_FLT	
MDEBUG_FONTSET	
MDEBUG_INPUT	
MDEBUG_ALL	
MDEBUG_MAX	

4.28.3 関数詳解

4.28.3.1 mdebug__add_object_array()

```
void mdebug__add_object_array (
    M17NObjectArray * array,
    char * name )
```

4.28.3.2 mdebug_register_object()

```
void mdebug_register_object (
    M17NObjectArray * array,
    void * object )
```

4.28.3.3 mdebug_unregister_object()

```
void mdebug_unregister_object (
    M17NObjectArray * array,
    void * object )
```

4.28.3.4 mdebug_push_time()

```
void mdebug_push_time ( )
```

4.28.3.5 mdebug_pop_time()

```
void mdebug_pop_time ( )
```

4.28.3.6 mdebug_print_time()

```
void mdebug_print_time ( )
```

4.28.3.7 msymbol_init()

```
int msymbol_init ( )
```

4.28.3.8 msymbol_fini()

```
void msymbol_fini ( )
```

4.28.3.9 mplist_init()

```
int mplist_init ( )
```

4.28.3.10 mplist_fini()

```
void mplist_fini ( )
```

4.28.3.11 mtext_init()

```
int mtext_init ( )
```

4.28.3.12 mtext_fini()

```
void mtext_fini ( )
```

4.28.3.13 mtext_prop_init()

```
int mtext_prop_init ( )
```

4.28.3.14 mtext_prop_fini()

```
void mtext_prop_fini ( )
```

4.28.3.15 mchartable_init()

```
int mchartable_init ( )
```

4.28.3.16 mchartable_fini()

```
void mchartable_fini ( )
```

4.28.3.17 mcharset_init()

```
int mcharset_init ( )
```

4.28.3.18 mcharset_fini()

```
void mcharset_fini ( )
```

4.28.3.19 mcoding_init()

```
int mcoding_init ( )
```

4.28.3.20 mcoding_fini()

```
void mcoding_fini ( )
```

4.28.3.21 mdatabase_init()

```
int mdatabase_init (
    void )
```

4.28.3.22 mdatabase_fini()

```
void mdatabase_fini (
    void )
```

4.28.3.23 mchar__init()

```
int mchar__init ( )
```

4.28.3.24 mchar__fini()

```
void mchar__fini ( )
```

4.28.3.25 mlang__init()

```
int mlang__init ( )
```

4.28.3.26 mlang__fini()

```
void mlang__fini ( )
```

4.28.3.27 mlocale__init()

```
int mlocale__init ( )
```

4.28.3.28 mlocale__fini()

```
void mlocale__fini ( )
```

4.28.3.29 minput__init()

```
int minput__init ( )
```

4.28.3.30 minput_fini()

```
void minput_fini ( )
```

4.28.4 変数詳解

4.28.4.1 m17n_core_initialized

```
int m17n_core_initialized [extern]
```

4.28.4.2 m17n_shell_initialized

```
int m17n_shell_initialized [extern]
```

4.28.4.3 m17n_gui_initialized

```
int m17n_gui_initialized [extern]
```

4.28.4.4 mdebug_flags

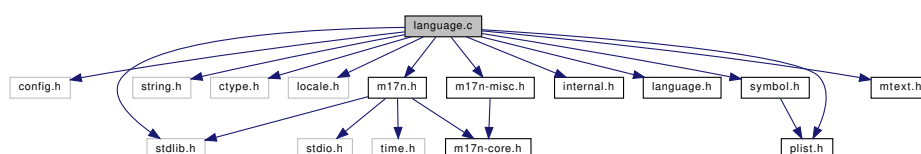
```
int mdebug_flags[MDEBUG_MAX] [extern]
```

4.28.4.5 mdebug_output

```
FILE* mdebug_output [extern]
```

4.29 language.c ファイル

language.c の依存先関係図:



関数

- [MPlist * mlanguage_jlist](#) (void)
3 文字言語コードをリストする.
- [MSymbol mlanguage_code](#) ([MSymbol](#) language, int len)
言語コードを得る.
- [MPlist * mlanguage_name_list](#) ([MSymbol](#) language, [MSymbol](#) target, [MSymbol](#) script, [MSymbol](#) territory)
- [MText * mlanguage_text](#) ([MSymbol](#) language)
与えられた言語自身で書かれた言語名を返す.
- [MPlist * mscript_list](#) (void)
スクリプト名をリストする.
- [MPlist * mscript_language_list](#) ([MSymbol](#) script)
与えられたスクリプトを用いる言語をリストする.

Obsolete な関数

言語の英語名を得る.

関数 [mlanguage_name\(\)](#) は、language の英語名を名前とするようなシンボルを返す。language はシンボルであり、その名前は、ISO639-2 3 文字言語コード、ISO639-1 2 文字言語コード、英語名、のいずれかである。

戻り値:

求めている情報が得られるなら、この関数は [Mnil](#) 以外のシンボルを返す。そうでなければ [Mnil](#) を返す。

参照:

[mlanguage_code\(\)](#), [mlanguage_text\(\)](#).

- [MSymbol mlanguage_name](#) ([MSymbol](#) language)

変数

- [MSymbol Miso639_1](#)
- [MSymbol Miso639_2](#)

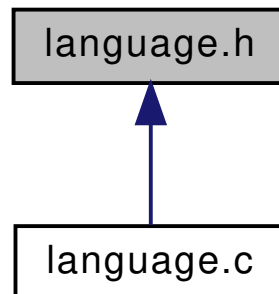
4.29.1 関数詳解

4.29.1.1 mlanguage_name()

```
MSymbol mlanguage_name (  
    MSymbol language )
```

4.30 language.h ファイル

被依存関係図:



関数

- [MPlist * mscript_char_list](#) ([MSymbol](#) script)
- [MSymbol mscript_otf_tag](#) ([MSymbol](#) script)
- [MSymbol mscript_from_otf_tag](#) ([MSymbol](#) otf_tag)

4.30.1 関数詳解

4.30.1.1 mscript_char_list()

```
MPlist* mscript_char_list (  
    MSymbol script )
```

4.30.1.2 mscript_otf_tag()

```
MSymbol mscript_otf_tag (  
    MSymbol script )
```

4.30.1.3 mscript_from_otf_tag()

```
MSymbol mscript_from_otf_tag (  
    MSymbol otf_tag )
```


4.31 locale.c ファイル

関数

- `MLocale * mlocale_set` (int category, const char *name)
現在のロケールを設定する.
- `MSymbol mlocale_get_prop` (MLocale *locale, MSymbol key)
ロケールプロパティの値を得る.
- `int mtextftime` (MText *mt, const char *format, const struct tm *tm, MLocale *locale)
日付と時間をフォーマットする.
- `MText * mtext_getenv` (const char *name)
環境変数を得る.
- `int mtext_putenv` (MText *mt)
環境変数を変更／追加する.
- `int mtext_coll` (MText *mt1, MText *mt2)
現在のロケールを用いて 2 つの M-text を比較する.

変数

- `MSymbol Mterritory`
- `MSymbol Mmodifier`
- `MSymbol Mcodeset`

4.32 m17n-config.txt ファイル

4.33 m17n-core.c ファイル

マクロ定義

- `#define M17NLIB_MAJOR_VERSION`
- `#define M17NLIB_MINOR_VERSION`
- `#define M17NLIB_PATCH_LEVEL`
- `#define M17NLIB_VERSION_NAME`
- `#define M17N_INIT()`
m17n ライブラリを初期化する.
- `#define M17N_FINI()`
m17n ライブラリを終了する.

関数

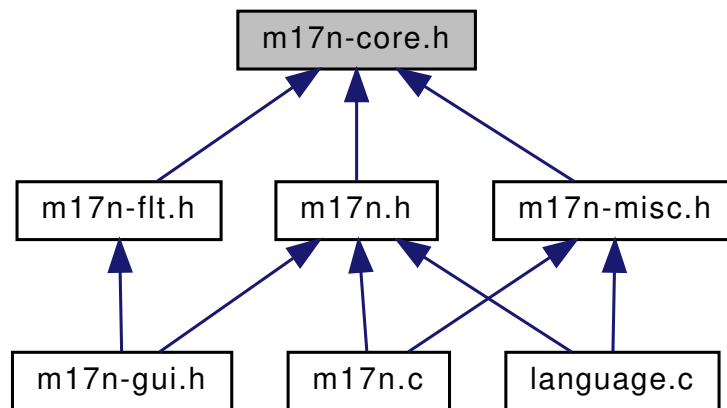
- `enum M17NStatus m17n_status` (void)
m17n ライブラリのどの部分が初期化されたか報告する.
- `void * m17n_object` (int size, void(*freer)(void *))
- `int m17n_object_ref` (void *object)
管理下オブジェクトの参照数を 1 増やす.
- `int m17n_object_unref` (void *object)
管理下オブジェクトの参照数を 1 減らす.
- `int mdebug_hook` ()
エラーの際に呼ばれるフック関数.

変数

- int `merror_code`
m17n ライブラリのエラーコードを保持する外部変数.
- void(* `m17n_memory_full_handler`)(enum `MErrorCode` err)
メモリ割当てエラーハンドラ.

4.34 m17n-core.h ファイル

被依存関係図:



データ構造

- struct `M17NObjectHead`
管理下オブジェクトの最初のメンバ.

マクロ定義

- #define `M17N_BEGIN_HEADER`
- #define `M17N_END_HEADER`
- #define `M17N_FUNC(func) ((M17NFunc) (func))`
汎関数型へのラップ.

型定義

- typedef void(* `M17NFunc`) (void)
汎関数型.
- typedef struct `MCharTable` `MCharTable`
文字テーブルの型宣言.
- typedef `MPlist` *(* `MTextPropSerializeFunc`) (void *val)
シリアライズ関数の型宣言.
- typedef void *(* `MTextPropDeserializeFunc`) (`MPlist` *plist)
デシリアライズ関数の型宣言.
- typedef struct `MDatabase` `MDatabase`
データベースの型宣言.

列挙型

- enum `M17NStatus` {
`M17N_NOT_INITIALIZED` ,
`M17N_CORE_INITIALIZED` ,
`M17N_SHELL_INITIALIZED` ,
`M17N_GUI_INITIALIZED` }
 m17n ライブラリの状態を示す列挙型.
- enum `MTextFormat` {
`MTEXT_FORMAT_US_ASCII` ,
`MTEXT_FORMAT_UTF_8` ,
`MTEXT_FORMAT_UTF_16LE` ,
`MTEXT_FORMAT_UTF_16BE` ,
`MTEXT_FORMAT_UTF_32LE` ,
`MTEXT_FORMAT_UTF_32BE` ,
`MTEXT_FORMAT_MAX` }
 M-text のフォーマットを指定する列挙型.
- enum `MTextLineBreakOption` {
`MTEXT_LBO_SP_CM` = 1 ,
`MTEXT_LBO_KOREAN_SP` = 2 ,
`MTEXT_LBO_AI_AS_ID` = 4 ,
`MTEXT_LBO_MAX` }
- enum `MTextPropertyControl` {
`MTEXTPROP_FRONT_STICKY` = 0x01 ,
`MTEXTPROP_REAR_STICKY` = 0x02 ,
`MTEXTPROP_VOLATILE_WEAK` = 0x04 ,
`MTEXTPROP_VOLATILE_STRONG` = 0x08 ,
`MTEXTPROP_NO_MERGE` = 0x10 ,
`MTEXTPROP_CONTROL_MAX` = 0x1F }
 テキストプロパティを制御するフラグビット.

関数

- enum `M17NStatus m17n_status` (void)
 m17n ライブラリのどの部分が初期化されたか報告する.
- void * `m17n_object` (int size, void(*freer)(void *))
- int `m17n_object_ref` (void *object)
 管理下オブジェクトの参照数を 1 増やす.
- int `m17n_object_unref` (void *object)
 管理下オブジェクトの参照数を 1 減らす.
- `MSymbol msymbol` (const char *name)
 シンボルを得る.
- `MSymbol msymbol_as_managing_key` (const char *name)
 管理キーを作る.
- int `msymbol_is_managing_key` (`MSymbol` symbol)
- `MSymbol msymbol_exist` (const char *name)
 指定された名前を持つシンボルを探す.
- char * `msymbol_name` (`MSymbol` symbol)
 シンボルの名前を得る.
- int `msymbol_put` (`MSymbol` symbol, `MSymbol` key, void *val)
 シンボルプロパティに値を設定する.

- `void * msymbol_get (MSymbol symbol, MSymbol key)`
シンボルプロパティの値を得る.
- `int msymbol_put_func (MSymbol symbol, MSymbol key, M17NFunc func)`
シンボルプロパティの値 (関数ポインタ) を設定する.
- `M17NFunc msymbol_get_func (MSymbol symbol, MSymbol key)`
シンボルプロパティの値 (関数ポインタ) を得る.
- `MPList * mplist ()`
プロパティリストオブジェクトを作る.
- `MPList * mplist_copy (MPList *plist)`
プロパティリストをコピーする.
- `MPList * mplist_add (MPList *plist, MSymbol key, void *val)`
プロパティリスト末尾にプロパティを追加する.
- `MPList * mplist_push (MPList *plist, MSymbol key, void *val)`
プロパティリストの先頭にプロパティを挿入する.
- `void * mplist_pop (MPList *plist)`
プロパティリストの先頭からプロパティを削除する.
- `MPList * mplist_put (MPList *plist, MSymbol key, void *val)`
プロパティリスト中のプロパティの値を設定する.
- `void * mplist_get (MPList *plist, MSymbol key)`
プロパティリスト中のプロパティの値を得る.
- `MPList * mplist_put_func (MPList *plist, MSymbol key, M17NFunc func)`
プロパティリスト中のプロパティに関数ポインタである値を設定する.
- `M17NFunc mplist_get_func (MPList *plist, MSymbol key)`
プロパティリストからプロパティの関数ポインタである値を得る.
- `MPList * mplist_find_by_key (MPList *plist, MSymbol key)`
プロパティリスト中から指定のキーを持つプロパティを探す.
- `MPList * mplist_find_by_value (MPList *plist, void *val)`
プロパティリスト中から指定の値を持つプロパティを探す.
- `MPList * mplist_next (MPList *plist)`
プロパティリストの次の部分リストを返す.
- `MPList * mplist_set (MPList *plist, MSymbol key, void *val)`
プロパティリストの最初のプロパティを設定する.
- `int mplist_length (MPList *plist)`
プロパティリストの長さを返す.
- `MSymbol mplist_key (MPList *plist)`
プロパティリスト中の最初のプロパティのキーを返す.
- `void * mplist_value (MPList *plist)`
プロパティリスト中の最初のプロパティの値を返す.
- `MSymbol mchar_define_property (const char *name, MSymbol type)`
文字プロパティを定義する.
- `void * mchar_get_prop (int c, MSymbol key)`
文字プロパティの値を得る.
- `int mchar_put_prop (int c, MSymbol key, void *val)`
文字プロパティの値を設定する.
- `MCharTable * mchartable (MSymbol key, void *default_value)`
新しい文字テーブルを作る.
- `int mchartable_min_char (MCharTable *table)`
- `int mchartable_max_char (MCharTable *table)`

- `void * mchartable_lookup (MCharTable *table, int c)`
文字テーブル中で文字に割り当てられた値を返す.
- `int mchartable_set (MCharTable *table, int c, void *val)`
文字テーブル中での文字の値を設定する.
- `int mchartable_set_range (MCharTable *table, int from, int to, void *val)`
指定範囲の文字に値を設定する.
- `int mchartable_map (MCharTable *table, void *ignore, void(*func)(int, int, void *, void *), void *func_arg)`
文字テーブル中の文字に対して指定の関数を呼ぶ.
- `void mchartable_range (MCharTable *table, int *from, int *to)`
値がデフォルトと異なる文字を探す.
- `MCharTable * mchar_get_prop_table (MSymbol key, MSymbol *type)`
文字プロパティの文字テーブルを得る.
- `MText * mtext ()`
新しいM-text を割り当てる.
- `void * mtext_data (MText *mt, enum MTextFormat *fmt, int *nunits, int *pos_idx, int *unit_idx)`
- `int mtext_len (MText *mt)`
M-text 中の文字の数.
- `int mtext_ref_char (MText *mt, int pos)`
M-text 中の指定された位置の文字を返す.
- `int mtext_set_char (MText *mt, int pos, int c)`
M-text に一文字を設定する.
- `MText * mtext_copy (MText *mt1, int pos, MText *mt2, int from, int to)`
M-text に指定範囲の文字をコピーする.
- `int mtext_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域同士を比較する.
- `int mtext_case_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域を、大文字／小文字の区別を無視して比較する.
- `int mtext_character (MText *mt, int from, int to, int c)`
M-text 中で文字を探す.
- `int mtext_del (MText *mt, int from, int to)`
指定範囲の文字を破壊的に取り除く.
- `int mtext_ins (MText *mt1, int pos, MText *mt2)`
M-text を別の M-text に挿入する.
- `int mtext_insert (MText *mt1, int pos, MText *mt2, int from, int to)`
M-text の一部を別の M-text に挿入する.
- `int mtext_ins_char (MText *mt, int pos, int c, int n)`
M-text に文字を挿入する.
- `int mtext_replace (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
M-text の一部を別の M-text の一部で置換する.
- `MText * mtext_cat_char (MText *mt, int c)`
M-text に一文字追加する.
- `MText * mtext_duplicate (MText *mt, int from, int to)`
既存の M-text の一部から新しい M-text をつくる.
- `MText * mtext_dup (MText *mt)`
M-text のコピーを作る.
- `MText * mtext_cat (MText *mt1, MText *mt2)`
2 個の M-text を連結する.
- `MText * mtext_ncat (MText *mt1, MText *mt2, int n)`

- M-text の一部を別の M-text に付加する.
- `MText * mtext_cpy (MText *mt1, MText *mt2)`
M-text を別の M-text にコピーする.
- `MText * mtext_ncpy (MText *mt1, MText *mt2, int n)`
M-text に含まれる最初の何文字かをコピーする.
- `int mtext_chr (MText *mt, int c)`
M-text 中で指定された文字が最初に現れる位置を返す.
- `int mtext_rchr (MText *mt, int c)`
M-text 中で指定された文字が最後に現れる位置を返す.
- `int mtext_cmp (MText *mt1, MText *mt2)`
二つの M-text を文字単位で比較する.
- `int mtextncmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を文字単位で比較する.
- `int mtext_spn (MText *mt1, MText *mt2)`
ある集合の文字を M-text の中で探す.
- `int mtext_cspn (MText *mt1, MText *mt2)`
ある集合に属さない文字を M-text の中で探す.
- `int mtext_pbrk (MText *mt1, MText *mt2)`
ある集合に属す文字を M-text の中から探す.
- `int mtext_text (MText *mt1, int pos, MText *mt2)`
M-text 中で別の M-text を探す.
- `int mtext_search (MText *mt1, int from, int to, MText *mt2)`
M-text 中の特定の領域で別の M-text を探す.
- `MText * mtext_tok (MText *mt, MText *delim, int *pos)`
M-text 中のトークンを探す.
- `int mtext_casecmp (MText *mt1, MText *mt2)`
二つの M-text を大文字／小文字の区別を無視して比較する.
- `int mtext_ncasecmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を大文字／小文字の区別を無視して比較する.
- `int mtext_lowercase (MText *mt)`
M-text を小文字にする.
- `int mtext_titlecase (MText *mt)`
M-text をタイトルケースにする.
- `int mtext_uppercase (MText *mt)`
M-text を大文字にする.
- `int mtext_line_break (MText *mt, int pos, int option, int *after)`
- `MPlist * mplist_deserialize (MText *mt)`
M-text をデシリアライズしてプロパティリストを作る.
- `void * mtext_get_prop (MText *mt, int pos, MSymbol key)`
テキストプロパティの一番上の値を得る.
- `int mtext_get_prop_values (MText *mt, int pos, MSymbol key, void **values, int num)`
テキストプロパティの値を複数個得る.
- `int mtext_get_prop_keys (MText *mt, int pos, MSymbol **keys)`
M-text の指定した位置のテキストプロパティのキーのリストを得る.
- `int mtext_put_prop (MText *mt, int from, int to, MSymbol key, void *val)`
- `int mtext_put_prop_values (MText *mt, int from, int to, MSymbol key, void **values, int num)`
同じキーのテキストプロパティを複数設定する.
- `int mtext_push_prop (MText *mt, int from, int to, MSymbol key, void *val)`
- `int mtext_pop_prop (MText *mt, int from, int to, MSymbol key)`

- `int mtext_prop_range (MText *mt, MSymbol key, int pos, int *from, int *to, int deeper)`
テキストプロパティが同じ値をとる範囲を調べる。
- `MTextProperty * mtext_property (MSymbol key, void *val, int control_bits)`
テキストプロパティを生成する。
- `MText * mtext_property_mtext (MTextProperty *prop)`
あるテキストプロパティを持つ M-text を返す。
- `MSymbol mtext_property_key (MTextProperty *prop)`
テキストプロパティのキーを返す。
- `void * mtext_property_value (MTextProperty *prop)`
テキストプロパティの値を返す。
- `int mtext_property_start (MTextProperty *prop)`
テキストプロパティの開始位置を返す。
- `int mtext_property_end (MTextProperty *prop)`
テキストプロパティの終了位置を返す。
- `MTextProperty * mtext_get_property (MText *mt, int pos, MSymbol key)`
一番上のテキストプロパティを得る。
- `int mtext_get_properties (MText *mt, int pos, MSymbol key, MTextProperty **props, int num)`
複数のテキストプロパティを得る。
- `int mtext_attach_property (MText *mt, int from, int to, MTextProperty *prop)`
M-text にテキストプロパティを付加する。
- `int mtext_detach_property (MTextProperty *prop)`
M-text からテキストプロパティを分離する。
- `int mtext_push_property (MText *mt, int from, int to, MTextProperty *prop)`
M-text にテキストプロパティをプッシュする。
- `MText * mtext_serialize (MText *mt, int from, int to, MPlist *property_list)`
- `MText * mtext_deserialize (MText *mt)`
- `MDatabase * mdatabase_find (MSymbol tag1, MSymbol tag2, MSymbol tag3, MSymbol tag4)`
データベース中のデータを探す。
- `MPlist * mdatabase_list (MSymbol tag0, MSymbol tag1, MSymbol tag2, MSymbol tag3)`
m17n データベースのデータリストを返す。
- `void * mdatabase_load (MDatabase *mdb)`
データベースからデータをロードする。
- `MSymbol * mdatabase_tag (MDatabase *mdb)`
データのタグを得る。
- `MDatabase * mdatabase_define (MSymbol tag1, MSymbol tag2, MSymbol tag3, MSymbol tag4, void *(*loader)(MSymbol *, void *), void *extra_info)`
m17n データベースのデータを定義する。

変数

- `MSymbol Mnil`
"nil" を名前として持つシンボル。
- `MSymbol Mt`
"t" を名前として持つシンボル。
- `MSymbol Mstring`
"string" を名前として持つシンボル。
- `MSymbol Msymbol`
"symbol" を名前として持つシンボル。

- [MSymbol Mtext](#)
"mtext" を名前として持つシンボル.
- [MSymbol Mcharset](#)
- [MSymbol Mplist](#)
"plist" を名前として持つシンボル.
- [MSymbol Minteger](#)
- [MSymbol Mscript](#)
スクリプトを表わすキー.
- [MSymbol Mname](#)
名前を表わすキー.
- [MSymbol Mcategory](#)
一般カテゴリを表わすキー.
- [MSymbol Mcombining_class](#)
標準結合クラスを表わすキー.
- [MSymbol Mbidi_category](#)
双方向カテゴリを表わすキー.
- [MSymbol Msimple_case_folding](#)
対応する小文字一文字を表わすキー.
- [MSymbol Mcomplicated_case_folding](#)
対応する小文字の列を表わすキー.
- [MSymbol Mcased](#)
Case 処理に用いられる値のキー.
- [MSymbol Msoft_dotted](#)
- [MSymbol Mcase_mapping](#)
- [MSymbol Mblock](#)
スクリプトブロック名を表すキー.
- [MSymbol Mchar_table](#)
- [MSymbol Mlanguage](#)
- [MSymbol Mtext_prop_serializer](#)
シリアライザ関数を指定するシンボル.
- [MSymbol Mtext_prop_deserializer](#)
デシリアライザ関数を指定するシンボル.
- `char * mdatabase_dir`

変数: UTF-16 と UTF-32 のデフォルトのエンディアン

- `enum MTextFormat MTEXT_FORMAT_UTF_16`
値が `MTEXT_FORMAT_UTF_16LE` か `MTEXT_FORMAT_UTF_16BE` である変数
- `const int MTEXT_FORMAT_UTF_32`
値が `MTEXT_FORMAT_UTF_32LE` か `MTEXT_FORMAT_UTF_32BE` である変数
- `MText * mtext_from_data (const void *data, int nitems, enum MTextFormat format)`
指定のデータを元に新しい M-text を割り当てる.

4.34.1 マクロ定義詳解

4.34.1.1 M17N_BEGIN_HEADER

```
#define M17N_BEGIN_HEADER
```

4.34.1.2 M17N_END_HEADER

```
#define M17N_END_HEADER
```

4.34.2 変数詳解

4.34.2.1 Minteger

MSymbol Minteger

4.34.2.2 Msoft_dotted

MSymbol Msoft_dotted

4.34.2.3 Mcase_mapping

MSymbol Mcase_mapping

4.35 m17n-db.txt ファイル

4.36 m17n-flt.c ファイル

関数

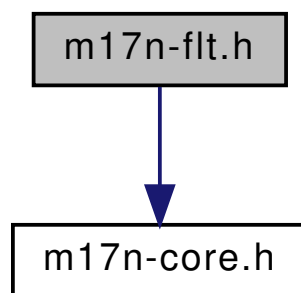
- **MFLT *** **mflt_get** (**MSymbol** name)
指定された名前を持つ FLT オブジェクトを返す.
- **MFLT *** **mflt_find** (int c, **MFLTFont ***font)
指定された文字とフォントに合った FLT を探す.
- const char * **mflt_name** (**MFLT ***flt)
FLT の名前を返す.
- **MCharTable *** **mflt_coverage** (**MFLT ***flt)
FLT の範囲を返す.
- int **mflt_run** (**MFLTGlyphString ***gstring, int from, int to, **MFLTFont ***font, **MFLT ***flt)
FLT を使って文字をレイアウトする.
- **MFLT *** **mdebug_dump_fl** (**MFLT ***flt, int indent)
- void **mflt_dump_gstring** (**MFLTGlyphString ***gstring)

変数

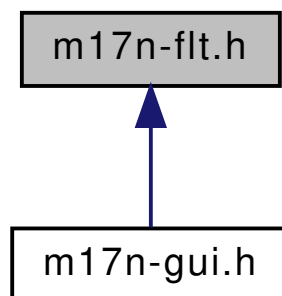
- int [mflt_enable_new_feature](#)
- int(* [mflt_iterate_otf_feature](#))(struct _MFLTFont *font, [MFLTOfSpec](#) *spec, int from, int to, unsigned char *table)
- [MSymbol](#)(* [mflt_font_id](#))(struct _MFLTFont *font)
- int(* [mflt_try_otf](#))(struct _MFLTFont *font, [MFLTOfSpec](#) *spec, [MFLTGlyphString](#) *gstring, int from, int to)

4.37 m17n-flt.h ファイル

m17n-flt.h の依存先関係図:



被依存関係図:



データ構造

- struct [MFLTGlyph](#)
グリフに関する情報の型.
- struct [MFLTGlyphAdjustment](#)
グリフ位置調整情報のための型.
- struct [MFLTGlyphString](#)
グリフ列の情報のための型.
- struct [MFLTOfSpec](#)
GSUB および GPOS OpenType テーブルの仕様のための型.
- struct [MFLTFont](#)
FLT ドライバが使うフォントの型.

型定義

- `typedef struct _MFLT MFLT`
FLT (Font Layout Table) の型.

関数

- `MFLT * mflt_get (MSymbol name)`
指定された名前を持つ FLT オブジェクトを返す.
- `MFLT * mflt_find (int c, MFLTFont *font)`
指定された文字とフォントに合った FLT を探す.
- `const char * mflt_name (MFLT *flt)`
FLT の名前を返す.
- `MCharTable * mflt_coverage (MFLT *flt)`
FLT の範囲を返す.
- `int mflt_run (MFLTGlyphString *gstring, int from, int to, MFLTFont *font, MFLT *flt)`
FLT を使って文字をレイアウトする.

変数

- `int mflt_enable_new_feature`
- `MSymbol(* mflt_font_id)(MFLTFont *font)`
- `int(* mflt_iterate_otf_feature)(MFLTFont *font, MFLTOtfSpec *spec, int from, int to, unsigned char *table)`
- `int(* mflt_try_otf)(struct _MFLTFont *font, MFLTOtfSpec *spec, MFLTGlyphString *gstring, int from, int to)`

4.37.1 変数詳解

4.37.1.1 mflt_font_id

```
MSymbol(* mflt_font_id) (MFLTFont *font) (
    MFLTFont * font ) [extern]
```

4.37.1.2 mflt_iterate_otf_feature

```
int(* mflt_iterate_otf_feature) (MFLTFont *font, MFLTOtfSpec *spec, int from, int to, unsigned
char *table) (
    MFLTFont * font,
    MFLTOtfSpec * spec,
    int from,
    int to,
    unsigned char * table ) [extern]
```

4.38 m17n-gd.c ファイル

4.39 m17n-gui.c ファイル

関数

- `MFrame * mframe (MPlist *plist)`
新しいフレームを作る。
- `void * mframe_get_prop (MFrame *frame, MSymbol key)`

変数

- `MFrame * mframe_default`
デフォルトのフレーム。

変数：フレームパラメータ用キー

フレームを生成する際のパラメータに用いるシンボル。詳しくは関数 `mframe()` の説明参照。

`Mdevice`、`Mdisplay`、`Mscreen`、`Mdrawable`、`Mdepth`、`Mcolormap` はフレームプロパティのキーでもある。

- `MSymbol Mdevice`
- `MSymbol Mdisplay`
- `MSymbol Mscreen`
- `MSymbol Mdrawable`
- `MSymbol Mdepth`
- `MSymbol Mcolormap`
- `MSymbol Mwidget`
- `MSymbol Mgd`

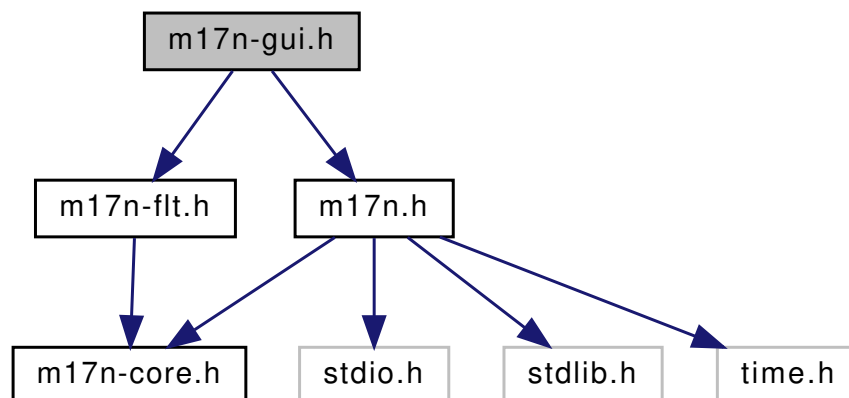
変数：フレームプロパティのキー

関数 `mframe_get_prop()` の引数に用いられるシンボル。

- `MSymbol Mfont`
- `MSymbol Mfont.width`
- `MSymbol Mfont.ascent`
- `MSymbol Mfont.descent`

4.40 m17n-gui.h ファイル

m17n-gui.h の依存先関係図:



データ構造

- struct [MFaceHLineProp](#)
フェースの水平線指定用型宣言.
- struct [MFaceBoxProp](#)
フェースの囲み枠指定用型宣言.
- struct [MDrawControl](#)
テキスト表示制御の型宣言.
- struct [MDrawMetric](#)
グリフとテキストの寸法の型宣言.
- struct [MDrawGlyphInfo](#)
グリフに関する情報の型宣言.
- struct [MDrawGlyph](#)
グリフの寸法とフォントに関する情報の型宣言.
- struct [MDrawTextItem](#)
textitem の型宣言.
- struct [MInputGUIArgIC](#)
関数 [minput_create_ic\(\)](#) の引数の型宣言.

型定義

- typedef struct [MFontset](#) [MFontset](#)
- typedef void(* [MFaceHookFunc](#)) ([MFace](#) *face, void *arg, void *info)
フェースのフック関数の型宣言.
- typedef void * [MDrawWindow](#)
ウィンドウシステムに依存する、ウィンドウの型宣言.
- typedef void * [MDrawRegion](#)
ウィンドウシステムに依存する、領域の型宣言.

関数

- [MFrame](#) * [mframe](#) ([MPlist](#) *plist)
新しいフレームを作る.
- void * [mframe_get_prop](#) ([MFrame](#) *frame, [MSymbol](#) key)
- [MFont](#) * [mfont](#) ()
新しいフォントを作る.
- [MFont](#) * [mfont_copy](#) ([MFont](#) *font)
フォントのコピーを作る.
- [MFont](#) * [mfont_parse_name](#) (const char *name, [MSymbol](#) format)
フォント名からフォントを作る.
- char * [mfont_unparse_name](#) ([MFont](#) *font, [MSymbol](#) format)
フォントからフォント名を作る.
- char * [mfont_name](#) ([MFont](#) *font)
フォント名からフォントを作る.
- [MFont](#) * [mfont_from_name](#) (const char *name)
フォントからフォント名を作る.
- void * [mfont_get_prop](#) ([MFont](#) *font, [MSymbol](#) key)

フォントのプロパティの値を得る.

- `int mfont_put_prop (MFont *font, MSymbol key, void *val)`
フォントのプロパティに値を設定する.
- `int mfont_set_encoding (MFont *font, MSymbol encoding_name, MSymbol repertory_name)`
フォントのエンコーディングを設定する.
- `MFont * mfont_find (MFrame *frame, MFont *spec, int *score, int limited_size)`
フォントを探す.
- `MSymbol * mfont_selection_priority ()`
フォント選択の優先度を返す.
- `int mfont_set_selection_priority (MSymbol *keys)`
フォント選択優先度を設定する.
- `int mfont_resize_ratio (MFont *font)`
フォントのリサイズ情報を得る
- `MPlist * mfont_list (MFrame *frame, MFont *font, MSymbol language, int maxnum)`
フォントのリストを得る
- `MPlist * mfont_list_family_names (MFrame *frame)`
- `int mfont_check (MFrame *frame, MFontset *fontset, MSymbol script, MSymbol language, MFont *font)`
- `int mfont_match_p (MFont *font, MFont *spec)`
- `MFont * mfont_open (MFrame *frame, MFont *font)`
- `MFont * mfont_encapsulate (MFrame *frame, MSymbol data_type, void *data)`
- `int mfont_close (MFont *font)`
- `MFontset * mfontset (char *name)`
フォントセットを返す.
- `MSymbol mfontset_name (MFontset *fontset)`
フォントセットの名前を返す.
- `MFontset * mfontset_copy (MFontset *fontset, char *name)`
フォントセットのコピーを作る.
- `int mfontset_modify_entry (MFontset *fontset, MSymbol script, MSymbol language, MSymbol charset, MFont *spec, MSymbol layouter_name, int how)`
フォントセットの内容を変更する.
- `MPlist * mfontset_lookup (MFontset *fontset, MSymbol script, MSymbol language, MSymbol charset)`
フォントセットを検索する.
- `MFace * mface ()`
新しいフェースをつくる.
- `int mface_equal (MFace *face1, MFace *face2)`
- `MFace * mface_copy (MFace *face)`
フェースのコピーを作る.
- `MFace * mface_merge (MFace *dst, MFace *src)`
フェースを統合する.
- `MFace * mface_from_font (MFont *font)`
フォントからフェースを作る.
- `void * mface_get_prop (MFace *face, MSymbol key)`
フェースのプロパティの値を得る.
- `int mface_put_prop (MFace *face, MSymbol key, void *val)`
フェースプロパティの値を設定する.
- `MFaceHookFunc mface_get_hook (MFace *face)`
フェースのフック関数を得る.
- `int mface_put_hook (MFace *face, MFaceHookFunc func)`
フェースのフック関数を設定する.

- void `mface_update` (`MFrame` *frame, `MFace` *face)
フェースを更新する.
- int `mdraw_text` (`MFrame` *frame, `MDrawWindow` win, int x, int y, `MText` *mt, int from, int to)
ウィンドウに M-text を描画する.
- int `mdraw_image_text` (`MFrame` *frame, `MDrawWindow` win, int x, int y, `MText` *mt, int from, int to)
ディスプレイに M-text を画像として描く.
- int `mdraw_text_with_control` (`MFrame` *frame, `MDrawWindow` win, int x, int y, `MText` *mt, int from, int to, `MDrawControl` *control)
ディスプレイに M-text を詳細な制御つきで描く.
- int `mdraw_coordinates_position` (`MFrame` *frame, `MText` *mt, int from, int to, int x, int y, `MDrawControl` *control)
指定した座標に最も近い文字の文字位置を得る.
- int `mdraw_text_extents` (`MFrame` *frame, `MText` *mt, int from, int to, `MDrawControl` *control, `MDrawMetric` *overall_link_return, `MDrawMetric` *overall_logical_return, `MDrawMetric` *overall_line_return)
テキストの幅 (ピクセル単位) を計算する.
- int `mdraw_text_per_char_extents` (`MFrame` *frame, `MText` *mt, int from, int to, `MDrawControl` *control, `MDrawMetric` *ink_array_return, `MDrawMetric` *logical_array_return, int array_size, int *num_chars_return, `MDrawMetric` *overall_link_return, `MDrawMetric` *overall_logical_return)
M-text の各文字の表示範囲を計算する.
- int `mdraw_glyph_info` (`MFrame` *frame, `MText` *mt, int from, int pos, `MDrawControl` *control, `MDrawGlyphInfo` *info)
グリフに関する情報を計算する.
- int `mdraw_glyph_list` (`MFrame` *frame, `MText` *mt, int from, int to, `MDrawControl` *control, `MDrawGlyph` *glyphs, int array_size, int *num_glyphs_return)
グリフ列に関する情報を計算する.
- void `mdraw_text_items` (`MFrame` *frame, `MDrawWindow` win, int x, int y, `MDrawTextItem` *items, int nitems)
textitem を表示する.
- void `mdraw_per_char_extents` (`MFrame` *frame, `MText` *mt, `MDrawMetric` *array_return, `MDrawMetric` *overall_return)
M-text の文字毎の表示範囲情報を得る.
- int `mdraw_default_line_break` (`MText` *mt, int pos, int from, int to, int line, int y)
改行位置を計算する.
- void `mdraw_clear_cache` (`MText` *mt)
キャッシュ情報を消す.
- `MSymbol` `minput_event_to_key` (`MFrame` *frame, void *event)
イベントを入力キーに変換する.
- `MFace` * `mdebug_dump_face` (`MFace` *face, int indent)
フェースをダンプする.
- `Font` * `mdebug_dump_font` (`Font` *font)
フォントをダンプする.
- `Fontset` * `mdebug_dump_fontset` (`Fontset` *fontset, int indent)
フォントセットをダンプする.

変数

- `MSymbol` Mdevice
- `MSymbol` Mfont
- `MSymbol` Mfont_width
- `MSymbol` Mfont_ascent

- [MSymbol Mfont_descent](#)
- [MFrame * mframe_default](#)
デフォルトのフレーム.
- [MSymbol Mdisplay](#)
- [MSymbol Mscreen](#)
- [MSymbol Mdrawable](#)
- [MSymbol Mwidget](#)
- [MSymbol Mdepth](#)
- [MSymbol Mcolormap](#)
- [MSymbol Mx](#)
"x" という名前を持つシンボル.
- [MSymbol Mfreetype](#)
- [MSymbol Mxft](#)
- [MPIlist * mfont_freetype_path](#)
フォントファイルとフォントファイルを含むディレクトリのリスト.
- [MSymbol Mfoundry](#)
開発元を指定するフォントプロパティのキー.
- [MSymbol Mfamily](#)
ファミリを指定するフォントプロパティのキー.
- [MSymbol Mweight](#)
太さを指定するフォントプロパティのキー.
- [MSymbol Mstyle](#)
スタイルを指定するフォントプロパティのキー.
- [MSymbol Mstretch](#)
幅を指定するフォントプロパティのキー.
- [MSymbol Madstyle](#)
adstyle を指定するフォントプロパティのキー.
- [MSymbol Mspacing](#)
spacing を指定するフォントプロパティのキー.
- [MSymbol Mregistry](#)
レジストリを指定するフォントプロパティのキー.
- [MSymbol Msize](#)
サイズを指定するフォントプロパティのキー.
- [MSymbol Mresolution](#)
解像度を指定するフォントプロパティのキー.
- [MSymbol Mmax_advance](#)
- [MSymbol Motf](#)
- [MSymbol Mfontfile](#)
フォントファイルを指定するフォントプロパティのキー.
- [MSymbol Mfontconfig](#)
"fontconfig" という名前を持つシンボル.
- [MSymbol Mforeground](#)
前景色を指定するフェースプロパティのキー.
- [MSymbol Mbackground](#)
背景色を指定するためのフェースプロパティのキー.
- [MSymbol Mvideomode](#)
ビデオモードを指定するためのフェースプロパティのキー.
- [MSymbol Mnormal](#)
- [MSymbol Mreverse](#)

- [MSymbol Mhline](#)
水平線を指定するためのフェースプロパティのキー.
- [MSymbol Mbox](#)
囲み枠を指定するためのフェースプロパティのキー.
- [MSymbol Mfontset](#)
フォントセットを指定するためのフェースプロパティのキー.
- [MSymbol Mratio](#)
フォントのサイズの比率を指定するためのフェースプロパティのキー.
- [MSymbol Mhook_func](#)
フックを指定するためのフェースプロパティのキー.
- [MSymbol Mhook_arg](#)
フックの引数を指定するためのフェースプロパティのキー.
- [MFace * mface_normal_video](#)
標準ビデオフェース.
- [MFace * mface_reverse_video](#)
リバーズビデオフェース.
- [MFace * mface_underline](#)
- [MFace * mface_medium](#)
ミディアムフェース.
- [MFace * mface_bold](#)
ボールドフェース.
- [MFace * mface_italic](#)
イタリックフェース.
- [MFace * mface_bold_italic](#)
ボールドイタリックフェース.
- [MFace * mface_xx_small](#)
最小のフェース.
- [MFace * mface_x_small](#)
より小さいフェース.
- [MFace * mface_small](#)
小さいフェース.
- [MFace * mface_normalsize](#)
標準の大きさのフェース.
- [MFace * mface_large](#)
大きいフェース.
- [MFace * mface_x_large](#)
もっと大きいフェース.
- [MFace * mface_xx_large](#)
最大のフェース.
- [MFace * mface_black](#)
黒フェース.
- [MFace * mface_white](#)
白フェース.
- [MFace * mface_red](#)
赤フェース.
- [MFace * mface_green](#)
緑フェース.
- [MFace * mface_blue](#)

- 青フェース.
- `MFace * mface_cyan`
シアンフェース.
- `MFace * mface_yellow`
黄フェース.
- `MFace * mface_magenta`
マゼンタフェース.
- `MSymbol Mface`
フェースを指定するテキストプロパティのキー.
- `int mdraw_line_break_option`
- `MInputDriver minput_gui_driver`
ウィンドウシステムの内部入力メソッド用入力ドライバ.

4.40.1 型定義詳解

4.40.1.1 MFontset

```
typedef struct MFontset MFontset
```

4.40.2 関数詳解

4.40.2.1 mdebug_dump_font()

```
MFont* mdebug_dump_font (  
    MFont * font )
```

フォントをダンプする.

関数 `mdebug_dump_font()` はフォント `font` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で出力する。

戻り値:

この関数は `font` を返す。

4.40.2.2 mdebug_dump_fontset()

```
MFontset* mdebug_dump_fontset (
    MFontset * fontset,
    int indent )
```

フォントセットをダンプする。

関数 `mdebug_dump_face()` はフォントセット `fontset` を標準エラー出力 もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読 な形で出力する。 `indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `fontset` を返す。

4.40.3 変数詳解

4.40.3.1 Mfreetype

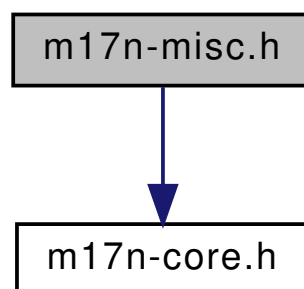
```
MSymbol Mfreetype
```

4.40.3.2 Mxft

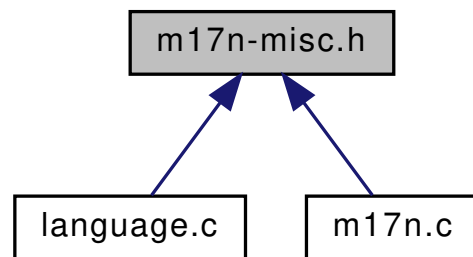
```
MSymbol Mxft
```

4.41 m17n-misc.h ファイル

m17n-misc.h の依存先関係図:



被依存関係図:



列挙型

- `enum MErrorCode` {
 MERROR_NONE ,
 MERROR_OBJECT ,
 MERROR_SYMBOL ,
 MERROR_MTEXT ,
 MERROR_TEXTPROP ,
 MERROR_CHAR ,
 MERROR_CHARTABLE ,
 MERROR_CHARSET ,
 MERROR_CODING ,
 MERROR_RANGE ,
 MERROR_LANGUAGE ,
 MERROR_LOCALE ,
 MERROR_PLIST ,
 MERROR_MISC ,
 MERROR_WIN ,
 MERROR_X ,
 MERROR_FRAME ,
 MERROR_FACE ,
 MERROR_DRAW ,
 MERROR_FLT ,
 MERROR_FONT ,
 MERROR_FONTSET ,
 MERROR_FONT_OTF ,
 MERROR_FONT_X ,
 MERROR_FONT_FT ,
 MERROR_IM ,
 MERROR_DB ,
 MERROR_IO ,
 MERROR_DEBUG ,
 MERROR_MEMORY ,
 MERROR_GD ,
 MERROR_MAX }
 m17n ライブラリエラーコードの列挙.

関数

- `int mdebug_hook (void)`

エラーの際に呼ばれるフック関数.

- `MSymbol mdebug_dump_symbol (MSymbol sym, int indent)`
シンボルをダンプする.
- `MSymbol mdebug_dump_all_symbols (int indent)`
すべてのシンボル名をダンプする.
- `MPList * mdebug_dump_plist (MPList *plist, int indent)`
プロパティリストをダンプする.
- `MText * mdebug_dump_mtext (MText *mt, int fullp, int indent)`
M-text をダンプする.
- `MCharTable * mdebug_dump_chartab (MCharTable *table, int indent)`
文字テーブルをダンプする.

変数

- `void(* m17n_memory_full_handler)(enum MErrorCode err)`
メモリ割当てエラーハンドラ.

4.41.1 関数詳解

4.41.1.1 mdebug_dump_plist()

```
MPList* mdebug_dump_plist (
    MPList * plist,
    int indent )
```

プロパティリストをダンプする.

関数 `mdebug_dump_plist()` はプロパティリスト `plist` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。`indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `plist` を返す。

4.41.1.2 mdebug_dump_chartab()

```
MCharTable* mdebug_dump_chartab (
    MCharTable * table,
    int indent )
```

文字テーブルをダンプする.

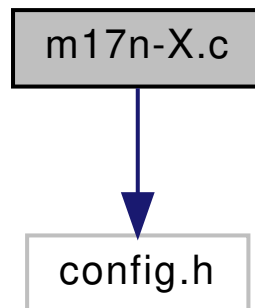
関数 `mdebug_dump_chartab()` は文字テーブル `table` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。`indent` は 2 行目以降のインデントを指定する。

戻り値:

この関数は `table` を返す。

4.42 m17n-X.c ファイル

m17n-X.c の依存先関係図:



関数

- int [device_open](#) ()

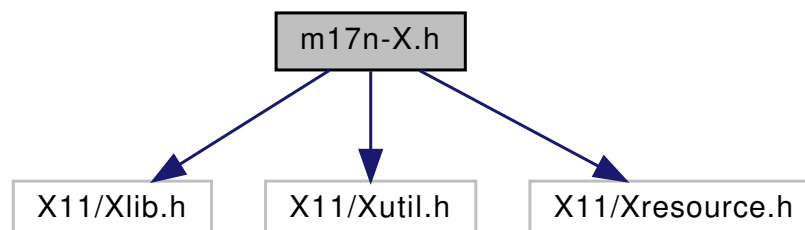
4.42.1 関数詳解

4.42.1.1 device_open()

```
int device_open ( )
```

4.43 m17n-X.h ファイル

m17n-X.h の依存先関係図:



データ構造

- struct [MInputXIMArgIM](#)
関数 [minput_open_im\(\)](#) の引数 `arg` によって指される構造体.
- struct [MInputXIMArgIC](#)
関数 [minput_create_ic\(\)](#) の引数 `arg` によって指される構造体.

変数

- `MInputDriver minput_xim_driver`
- `MSymbol Mxim`
"xim"を名前として持つシンボル.

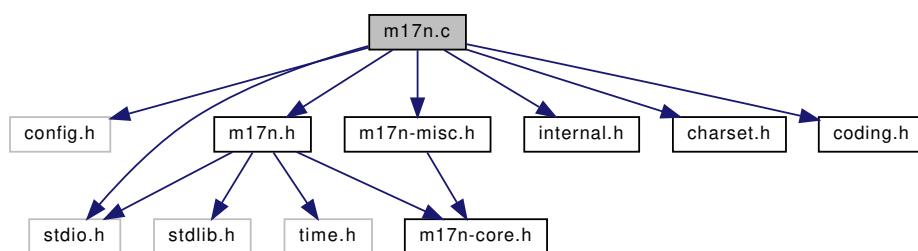
4.43.1 変数詳解

4.43.1.1 minput_xim_driver

```
MInputDriver minput_xim_driver [extern]
```

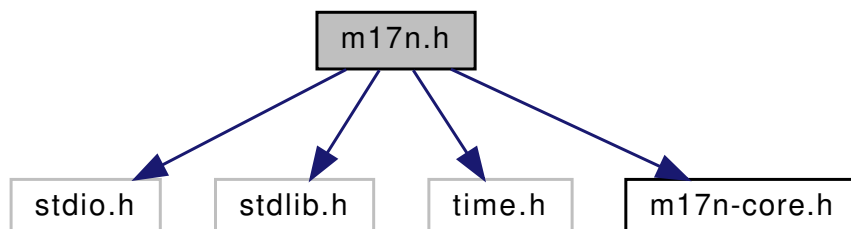
4.44 m17n.c ファイル

m17n.c の依存先関係図:

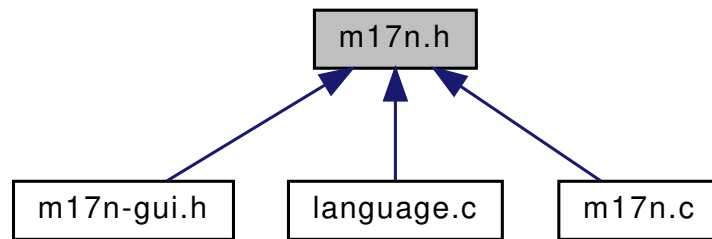


4.45 m17n.h ファイル

m17n.h の依存先関係図:



被依存関係図:



データ構造

- struct [MConverter](#)
コード変換に用いられる構造体.
- struct [MCodingInfoISO2022](#)
[MCODING_TYPE_ISO_2022](#) タイプのコード系に必要な付加情報用構造体.
- struct [MCodingInfoUTF](#)
[MCODING_TYPE_UTF](#) タイプのコード系に必要な付加情報用の構造体.
- struct [MInputDriver](#)
入力ドライバ用構造体.
- struct [MInputMethod](#)
入力メソッドの構造体.
- struct [MInputContext](#)
入力コンテキスト用構造体.

型定義

- typedef struct [MLocale](#) [MLocale](#)
[MLocale](#) 構造体.
- typedef void(* [MInputCallbackFunc](#)) ([MInputContext](#) *ic, [MSymbol](#) command)
入力メソッドコールバック関数の型宣言.

列挙型

- enum [MConversionResult](#) {
[MCONVERSION_RESULT_SUCCESS](#) ,
[MCONVERSION_RESULT_INVALID_BYTE](#) ,
[MCONVERSION_RESULT_INVALID_CHAR](#) ,
[MCONVERSION_RESULT_INSUFFICIENT_SRC](#) ,
[MCONVERSION_RESULT_INSUFFICIENT_DST](#) ,
[MCONVERSION_RESULT_IO_ERROR](#) }
コード変換の結果を示すコード.
- enum [MCodingType](#) {
[MCODING_TYPE_CHARSET](#) ,
[MCODING_TYPE_UTF](#) ,
[MCODING_TYPE_ISO_2022](#) ,
[MCODING_TYPE_MISC](#) }

コード系のタイプ.

- enum `MCodingFlagISO2022` {
`MCODING_ISO_RESET_AT_EOL` = 0x1 ,
`MCODING_ISO_RESET_AT_CNTL` = 0x2 ,
`MCODING_ISO_EIGHT_BIT` = 0x4 ,
`MCODING_ISO_LONG_FORM` = 0x8 ,
`MCODING_ISO_DESIGNATION_G0` = 0x10 ,
`MCODING_ISO_DESIGNATION_G1` = 0x20 ,
`MCODING_ISO_DESIGNATION_CTEXT` = 0x40 ,
`MCODING_ISO_DESIGNATION_CTEXT_EXT` = 0x80 ,
`MCODING_ISO_LOCKING_SHIFT` = 0x100 ,
`MCODING_ISO_SINGLE_SHIFT` = 0x200 ,
`MCODING_ISO_SINGLE_SHIFT_7` = 0x400 ,
`MCODING_ISO_EUC_TW_SHIFT` = 0x800 ,
`MCODING_ISO_ISO6429` = 0x1000 ,
`MCODING_ISO_REVISION_NUMBER` = 0x2000 ,
`MCODING_ISO_FULL_SUPPORT` = 0x3000 ,
`MCODING_ISO_FLAG_MAX` }

`MCODING_TYPE_ISO_2022` タイプのコード系の詳細を表わすビットマスク.

- enum `MInputCandidatesChanged` {
`MINPUT_CANDIDATES_LIST_CHANGED` = 1 ,
`MINPUT_CANDIDATES_INDEX_CHANGED` = 2 ,
`MINPUT_CANDIDATES_SHOW_CHANGED` = 4 ,
`MINPUT_CANDIDATES_CHANGED_MAX` }

入力メソッドの入力候補がどう変更されたかを示すビットマスク.

関数

- `MSymbol mchar_define_charset` (const char *name, `MPList` *plist)
- `MSymbol mchar_resolve_charset` (`MSymbol` symbol)
文字セット名を解決する.
- int `mchar_list_charset` (`MSymbol` **symbols)
文字セットを表わすシンボルを列挙する.
- int `mchar_decode` (`MSymbol` charset_name, unsigned code)
コードポイントをデコードする.
- unsigned `mchar_encode` (`MSymbol` charset_name, int c)
文字コードをエンコードする.
- int `mchar_map_charset` (`MSymbol` charset_name, void(*func)(int from, int to, void *arg), void *func_arg)
指定した文字セットのすべての文字に対して関数を呼ぶ.
- `MSymbol mconv_define_coding` (const char *name, `MPList` *plist, int(*resetter)(`MConverter` *),
int(*decoder)(const unsigned char *, int, `MText` *, `MConverter` *), int(*encoder)(`MText` *, int, int, unsigned
char *, int, `MConverter` *), void *extra_info)
- `MSymbol mconv_resolve_coding` (`MSymbol` symbol)
コード系の名前を解決する.
- int `mconv_list_codings` (`MSymbol` **symbols)
コード系を表わすシンボルを列挙する.
- `MConverter` * `mconv_buffer_converter` (`MSymbol` coding, const unsigned char *buf, int n)
バッファに結び付けられたコードコンバータを作る.
- `MConverter` * `mconv_stream_converter` (`MSymbol` coding, FILE *fp)
ストリームに結び付けられたコードコンバータを作る.
- int `mconv_reset_converter` (`MConverter` *converter)

コードコンバータをリセットする。

- void `mconv_free_converter` (`MConverter` *converter)
コードコンバータを解放する。
- `MConverter` * `mconv_rebind_buffer` (`MConverter` *converter, const unsigned char *buf, int n)
コードコンバータにバッファ領域を結び付ける。
- `MConverter` * `mconv_rebind_stream` (`MConverter` *converter, FILE *fp)
コードコンバータにストリームを結び付ける。
- `MText` * `mconv_decode` (`MConverter` *converter, `MText` *mt)
バイト列を M-text にデコードする。
- `MText` * `mconv_decode_buffer` (`MSymbol` name, const unsigned char *buf, int n)
コード系に基づいてバッファ領域をデコードする。
- `MText` * `mconv_decode_stream` (`MSymbol` name, FILE *fp)
コード系に基づいてストリーム入力をデコードする。
- int `mconv_encode` (`MConverter` *converter, `MText` *mt)
M-text をバイト列にエンコードする。
- int `mconv_encode_range` (`MConverter` *converter, `MText` *mt, int from, int to)
M-text の一部をバイト列にエンコードする。
- int `mconv_encode_buffer` (`MSymbol` name, `MText` *mt, unsigned char *buf, int n)
M-text をエンコードしてバッファ領域に書き込む。
- int `mconv_encode_stream` (`MSymbol` name, `MText` *mt, FILE *fp)
M-text をエンコードしてストリームに書き込む。
- int `mconv_getc` (`MConverter` *converter)
コードコンバータ経由で一文字を読みこむ。
- int `mconv_ungetc` (`MConverter` *converter, int c)
コードコンバータに一文字戻す。
- int `mconv_putc` (`MConverter` *converter, int c)
コードコンバータを経由して一文字書き出す。
- `MText` * `mconv_gets` (`MConverter` *converter, `MText` *mt)
コードコンバータを使って一行読み込む。
- `MPlist` * `mlanguage_jlist` (void)
3 文字言語コードをリストする。
- `MSymbol` `mlanguage_code` (`MSymbol` language, int len)
言語コードを得る。
- `MPlist` * `mlanguage_name_jlist` (`MSymbol` language, `MSymbol` target, `MSymbol` script, `MSymbol` territory)
- `MText` * `mlanguage_text` (`MSymbol` language)
与えられた言語自身で書かれた言語名を返す。
- `MPlist` * `mscript_list` (void)
スクリプト名をリストする。
- `MPlist` * `mscript_language_jlist` (`MSymbol` script)
与えられたスクリプトを用いる言語をリストする。
- `MSymbol` `mlanguage_name` (`MSymbol` language)
- `MLocale` * `mlocale_set` (int category, const char *locale)
現在のロケールを設定する。
- `MSymbol` `mlocale_get_prop` (`MLocale` *locale, `MSymbol` key)
ロケールプロパティの値を得る。
- int `mtextftime` (`MText` *mt, const char *format, const struct tm *tm, `MLocale` *locale)
日付と時間をフォーマットする。
- `MText` * `mtext_getenv` (const char *name)

環境変数を得る.

- int `mtext_putenv` (`MText` *mt)
環境変数を変更／追加する.
- int `mtext_coll` (`MText` *mt1, `MText` *mt2)
現在のロケールを用いて 2 つの M-text を比較する.
- `MinputMethod` * `minput_open_im` (`MSymbol` language, `MSymbol` name, void *arg)
入力メソッドをオープンする.
- void `minput_close_im` (`MinputMethod` *im)
入力メソッドをクローズする.
- `MinputContext` * `minput_create_ic` (`MinputMethod` *im, void *arg)
入力コンテキストを生成する.
- void `minput_destroy_ic` (`MinputContext` *ic)
入力コンテキストを破壊する.
- int `minput_filter` (`MinputContext` *ic, `MSymbol` key, void *arg)
入力キーをフィルタする.
- int `minput_lookup` (`MinputContext` *ic, `MSymbol` key, void *arg, `MText` *mt)
入力コンテキスト中のテキストを探す.
- void `minput_set_spot` (`MinputContext` *ic, int x, int y, int ascent, int descent, int fontsize, `MText` *mt, int pos)
入力コンテキストのスポットを設定する.
- void `minput_toggle` (`MinputContext` *ic)
入力メソッドを切替える.
- void `minput_reset_ic` (`MinputContext` *ic)
入力コンテキストをリセットする.
- `MText` * `minput_get_description` (`MSymbol` language, `MSymbol` name)
入力メソッドの説明テキストを得る.
- `MPlist` * `minput_get_title_icon` (`MSymbol` language, `MSymbol` name)
入力メソッドのタイトルとアイコン用ファイル名を得る.
- `MPlist` * `minput_get_command` (`MSymbol` language, `MSymbol` name, `MSymbol` command)
- int `minput_config_command` (`MSymbol` language, `MSymbol` name, `MSymbol` command, `MPlist` *keyseq)
- `MPlist` * `minput_get_variable` (`MSymbol` language, `MSymbol` name, `MSymbol` variable)
- int `minput_config_variable` (`MSymbol` language, `MSymbol` name, `MSymbol` variable, `MPlist` *value)
入力メソッドの変数の値を設定する.
- char * `minput_config_file` (void)
ユーザ毎のカスタマイズファイルの名前を得る.
- int `minput_save_config` (void)
設定をユーザ毎のカスタマイズファイルに保存する.
- int `minput_callback` (`MinputContext` *ic, `MSymbol` command)
- `MPlist` * `minput_get_commands` (`MSymbol` language, `MSymbol` name)
入力メソッドのコマンドに関する情報を得る.
- int `minput_assign_command_keys` (`MSymbol` language, `MSymbol` name, `MSymbol` command, `MPlist` *keys)
入力メソッドコマンドにキーシーケンスを割り当てる.
- `MPlist` * `minput_get_variables` (`MSymbol` language, `MSymbol` name)
- int `minput_set_variable` (`MSymbol` language, `MSymbol` name, `MSymbol` variable, void *value)
入力メソッド変数の初期値を設定する.
- `MPlist` * `minput_parse_im_names` (`MText` *mt)
- `MPlist` * `minput_list` (`MSymbol` lang)
- `MinputMethod` * `mdebug_dump_im` (`MinputMethod` *im, int indent)
入力メソッドをダンプする.

変数

- [MSymbol Mcharset_ascii](#)
ASCII 文字セットを表現するシンボル.
- [MSymbol Mcharset_iso_8859_1](#)
ISO/IEC 8859-1:1998 文字セットを表現するシンボル.
- [MSymbol Mcharset_unicode](#)
Unicode 文字セットを表現するシンボル.
- [MSymbol Mcharset_m17n](#)
全文字を含む文字セットを表現するシンボル.
- [MSymbol Mcharset_binary](#)
正しくデコードできない文字の文字セットを表現するシンボル.
- [MSymbol Mmethod](#)
- [MSymbol Mdimension](#)
- [MSymbol Mmin_range](#)
- [MSymbol Mmax_range](#)
- [MSymbol Mmin_code](#)
- [MSymbol Mmax_code](#)
- [MSymbol Mascii_compatible](#)
- [MSymbol Mfinal_byte](#)
- [MSymbol Mrevision](#)
- [MSymbol Mmin_char](#)
- [MSymbol Mmapfile](#)
- [MSymbol Mparents](#)
- [MSymbol Msubset_offset](#)
- [MSymbol Mdefine_coding](#)
- [MSymbol Malias](#)
- [MSymbol Moffset](#)
- [MSymbol Mmap](#)
マップ型のメソッドを示すシンボル.
- [MSymbol Munify](#)
ユニファイ型のメソッドを示すシンボル.
- [MSymbol Msubset](#)
サブセット型のメソッドを示すシンボル.
- [MSymbol Msuperset](#)
スーパーセット型のメソッドを示すシンボル.
- [MSymbol Mcoding_us_ascii](#)
US-ASCII コード系のシンボル.
- [MSymbol Mcoding_iso_8859_1](#)
ISO-8859-1 コード系のシンボル.
- [MSymbol Mcoding_utf_8](#)
UTF-8 コード系のシンボル.
- [MSymbol Mcoding_utf_8_full](#)
UTF-8-FULL コード系のシンボル.
- [MSymbol Mcoding_utf_16](#)
UTF-16 コード系のシンボル.
- [MSymbol Mcoding_utf_16be](#)
UTF-16BE コード系のシンボル.
- [MSymbol Mcoding_utf_16le](#)
UTF-16LE コード系のシンボル.

- [MSymbol Mcoding_utf_32](#)
UTF-32 コード系のシンボル.
- [MSymbol Mcoding_utf_32be](#)
UTF-32BE コード系のシンボル.
- [MSymbol Mcoding_utf_32le](#)
UTF-32LE コード系のシンボル.
- [MSymbol Mcoding_sjis](#)
SJIS コード系のシンボル.
- [MSymbol Mtype](#)
- [MSymbol Mcharsets](#)
- [MSymbol Mflags](#)
- [MSymbol Mdesignation](#)
- [MSymbol Minvocation](#)
- [MSymbol Mcode_unit](#)
- [MSymbol Mbom](#)
- [MSymbol Mlittle_endian](#)
- [MSymbol Mutf](#)
- [MSymbol Miso_2022](#)
- [MSymbol Mreset_at_eol](#)
- [MSymbol Mreset_at_cntl](#)
- [MSymbol Meight_bit](#)
- [MSymbol Mlong_form](#)
- [MSymbol Mdesignation_g0](#)
- [MSymbol Mdesignation_g1](#)
- [MSymbol Mdesignation_ccontext](#)
- [MSymbol Mdesignation_ccontext_ext](#)
- [MSymbol Mlocking_shift](#)
- [MSymbol Msingle_shift](#)
- [MSymbol Msingle_shift_7](#)
- [MSymbol Meuc_tw_shift](#)
- [MSymbol Miso_6429](#)
- [MSymbol Mrevision_number](#)
- [MSymbol Mfull_support](#)
- [MSymbol Mcoding](#)
シンボル Mcoding.
- [MSymbol Mmaybe](#)
"maybe" という名前を持つシンボル.
- [MSymbol Miso639_1](#)
- [MSymbol Miso639_2](#)
- [MSymbol Mterritory](#)
- [MSymbol Mmodifier](#)
- [MSymbol Mcodeset](#)
- [MInputDriver minput_default_driver](#)
内部入力メソッド用デフォルトドライバ.
- [MSymbol Minput_method](#)
"input-method" を名前として持つシンボル.
- [MSymbol Minput_driver](#)
- [MInputDriver * minput_driver](#)
内部入力メソッド用ドライバ.
- [MSymbol Minput_preedit_start](#)
- [MSymbol Minput_preedit_draw](#)

- [MSymbol Minput_preedit_done](#)
- [MSymbol Minput_status_start](#)
- [MSymbol Minput_status_draw](#)
- [MSymbol Minput_status_done](#)
- [MSymbol Minput_candidates_start](#)
- [MSymbol Minput_candidates_draw](#)
- [MSymbol Minput_candidates_done](#)
- [MSymbol Minput_set_spot](#)
- [MSymbol Minput_toggle](#)
- [MSymbol Minput_reset](#)
- [MSymbol Minput_get_surrounding_text](#)
- [MSymbol Minput_delete_surrounding_text](#)
- [MSymbol Minput_focus_move](#)
- [MSymbol Minput_focus_in](#)
- [MSymbol Minput_focus_out](#)
- [MSymbol Minherited](#)
- [MSymbol Mcustomized](#)
- [MSymbol Mconfigured](#)

4.45.1 関数詳解

4.45.1.1 mlanguage_name()

```
MSymbol mlanguage_name (
    MSymbol language )
```

4.45.2 変数詳解

4.45.2.1 Miso639_2

```
MSymbol Miso639_2
```

4.46 mainpage.txt ファイル

4.47 mlocale.h ファイル

変数

- [MLocale * mlocale_collate](#)
- [MLocale * mlocale_ctype](#)
- [MLocale * mlocale_messages](#)
- [MLocale * mlocale_time](#)

4.47.1 変数詳解

4.47.1.1 mlocale_collate

```
MLocale* mlocale_collate [extern]
```

4.47.1.2 mlocale_ctype

```
MLocale * mlocale_ctype
```

4.47.1.3 mlocale_messages

```
MLocale* mlocale_messages [extern]
```

4.47.1.4 mlocale_time

```
MLocale * mlocale_time
```

4.48 mtext-lbrk.c ファイル

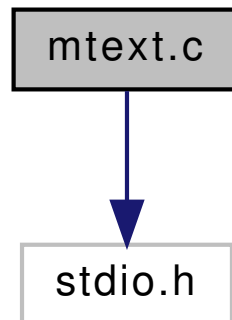
関数

- int [mtext_line_break](#) (MText *mt, int pos, int option, int *after)

4.49 mtext-wseg.c ファイル

4.50 mtext.c ファイル

mtext.c の依存先関係図:



関数

- `MText * mtext ()`
新しいM-text を割り当てる.
- `MText * mtext_from_data (const void *data, int nitems, enum MTextFormat format)`
指定のデータを元に新しい M-text を割り当てる.
- `void * mtext_data (MText *mt, enum MTextFormat *fmt, int *nunits, int *pos_idx, int *unit_idx)`
- `int mtext_len (MText *mt)`
M-text 中の文字の数.
- `int mtext_ref_char (MText *mt, int pos)`
M-text 中の指定された位置の文字を返す.
- `int mtext_set_char (MText *mt, int pos, int c)`
M-text に一文字を設定する.
- `MText * mtext_cat_char (MText *mt, int c)`
M-text に一文字追加する.
- `MText * mtext_dup (MText *mt)`
M-text のコピーを作る.
- `MText * mtext_cat (MText *mt1, MText *mt2)`
2 個の M-text を連結する.
- `MText * mtext_ncat (MText *mt1, MText *mt2, int n)`
M-text の一部を別の M-text に付加する.
- `MText * mtext_cpy (MText *mt1, MText *mt2)`
M-text を別の M-text にコピーする.
- `MText * mtext_ncpy (MText *mt1, MText *mt2, int n)`
M-text に含まれる最初の何文字かをコピーする.
- `MText * mtext_duplicate (MText *mt, int from, int to)`
既存の M-text の一部から新しい M-text をつくる.
- `MText * mtext_copy (MText *mt1, int pos, MText *mt2, int from, int to)`
M-text に指定範囲の文字をコピーする.
- `int mtext_del (MText *mt, int from, int to)`
指定範囲の文字を破壊的に取り除く.
- `int mtext_ins (MText *mt1, int pos, MText *mt2)`
M-text を別の M-text に挿入する.
- `int mtext_insert (MText *mt1, int pos, MText *mt2, int from, int to)`
M-text の一部を別の M-text に挿入する.
- `int mtext_ins_char (MText *mt, int pos, int c, int n)`
M-text に文字を挿入する.
- `int mtext_replace (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
M-text の一部を別の M-text の一部で置換する.
- `int mtext_character (MText *mt, int from, int to, int c)`
M-text 中で文字を探す.
- `int mtext_chr (MText *mt, int c)`
M-text 中で指定された文字が最初に現れる位置を返す.
- `int mtext_rchr (MText *mt, int c)`
M-text 中で指定された文字が最後に現れる位置を返す.
- `int mtext_cmp (MText *mt1, MText *mt2)`
二つの M-text を文字単位で比較する.
- `int mtextncmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を文字単位で比較する.

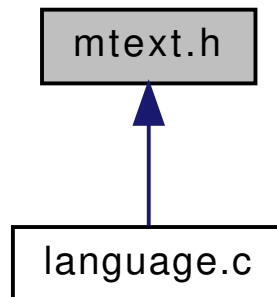
- `int mtext_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域同士を比較する。
- `int mtext_spn (MText *mt, MText *accept)`
ある集合の文字を M-text の中で探す。
- `int mtext_cspn (MText *mt, MText *reject)`
ある集合に属さない文字を M-text の中で探す。
- `int mtext_pbrk (MText *mt, MText *accept)`
ある集合に属す文字を M-text の中から探す。
- `MText * mtext_tok (MText *mt, MText *delim, int *pos)`
M-text 中のトークンを探す。
- `int mtext_text (MText *mt1, int pos, MText *mt2)`
M-text 中で別の M-text を探す。
- `int mtext_search (MText *mt1, int from, int to, MText *mt2)`
M-text 中の特定の領域で別の M-text を探す。
- `int mtext_casecmp (MText *mt1, MText *mt2)`
二つの M-text を大文字／小文字の区別を無視して比較する。
- `int mtext_ncasecmp (MText *mt1, MText *mt2, int n)`
二つの M-text の先頭部分を大文字／小文字の区別を無視して比較する。
- `int mtext_case_compare (MText *mt1, int from1, int to1, MText *mt2, int from2, int to2)`
二つの M-text の指定した領域を、大文字／小文字の区別を無視して比較する。
- `int mtext_lowercase (MText *mt)`
M-text を小文字にする。
- `int mtext_titlecase (MText *mt)`
M-text をタイトルケースにする。
- `int mtext_uppercase (MText *mt)`
M-text を大文字にする。
- `MText * mdebug_dump_mtext (MText *mt, int indent, int fullp)`
M-text をダンプする。

変数

- `enum MTextFormat MTEXT_FORMAT_UTF_16 = MTEXT_FORMAT_UTF_16LE`
値が MTEXT_FORMAT_UTF_16LE か MTEXT_FORMAT_UTF_16BE である変数
- `const int MTEXT_FORMAT_UTF_32 = MTEXT_FORMAT_UTF_32LE`
値が MTEXT_FORMAT_UTF_32LE か MTEXT_FORMAT_UTF_32BE である変数
- `MSymbol Mlanguage`

4.51 mtext.h ファイル

被依存関係図:



マクロ定義

- `#define POS_CHAR_TO_BYTE(mt, pos)`
- `#define POS_BYTE_TO_CHAR(mt, pos_byte)`
- `#define MTEXT_DATA(mt) ((mt)->data)`
- `#define MTEXT_CAT_ASCII(mt, str)`

関数

- `int mtext__char_to_byte (MText *mt, int pos)`
- `int mtext__byte_to_char (MText *mt, int pos_byte)`
- `void mtext__enlarge (MText *mt, int nbytes)`
- `int mtext__takein (MText *mt, int nchars, int nbytes)`
- `int mtext__cat_data (MText *mt, unsigned char *p, int nbytes, enum MTextFormat format)`
- `MText * mtext__from_data (const void *data, int nitems, enum MTextFormat format, int need_copy)`
- `void mtext__adjust_format (MText *mt, enum MTextFormat format)`
- `int mtext__bol (MText *mt, int pos)`
- `int mtext__eol (MText *mt, int pos)`
- `void mtext__wseg_fini ()`
- `int mtext__word_segment (MText *mt, int pos, int *from, int *to)`

4.51.1 マクロ定義詳解

4.51.1.1 POS_CHAR_TO_BYTE

```
#define POS_CHAR_TO_BYTE(
    mt,
    pos )
```

値:

```
(mtext_nchars (mt) == mtext_nbytes (mt) ? (pos) \
: (pos) == (mt)->cache_char_pos ? (mt)->cache_byte_pos \
: mtext__char_to_byte ((mt), (pos)))
```

4.51.1.2 POS_BYTE_TO_CHAR

```
#define POS_BYTE_TO_CHAR(  
    mt,  
    pos_byte )
```

値:

```
(mtext_nchars (mt) == mtext_nbytes (mt) ? (pos_byte) \  
: (pos_byte) == (mt)->cache_byte_pos ? (mt)->cache_char_pos \  
: mtext_byte_to_char ((mt), (pos_byte)))
```

4.51.1.3 MTEXT_DATA

```
#define MTEXT_DATA(  
    mt ) ((mt)->data)
```

4.51.1.4 MTEXT_CAT_ASCII

```
#define MTEXT_CAT_ASCII(  
    mt,  
    str )
```

値:

```
mtext_cat_data ((mt), (unsigned char *) (str), strlen (str), \  
    MTEXT_FORMAT_US_ASCII)
```

4.51.2 関数詳解

4.51.2.1 mtext_char_to_byte()

```
int mtext_char_to_byte (  
    MText * mt,  
    int pos )
```

4.51.2.2 mtext_byte_to_char()

```
int mtext_byte_to_char (  
    MText * mt,  
    int pos_byte )
```

4.51.2.3 mtext_enlarge()

```
void mtext_enlarge (
    MText * mt,
    int nbytes )
```

4.51.2.4 mtext_takein()

```
int mtext_takein (
    MText * mt,
    int nchars,
    int nbytes )
```

4.51.2.5 mtext_cat_data()

```
int mtext_cat_data (
    MText * mt,
    unsigned char * p,
    int nbytes,
    enum MTextFormat format )
```

4.51.2.6 mtext_from_data()

```
MText* mtext_from_data (
    const void * data,
    int nitems,
    enum MTextFormat format,
    int need_copy )
```

4.51.2.7 mtext_adjust_format()

```
void mtext_adjust_format (
    MText * mt,
    enum MTextFormat format )
```

4.51.2.8 mtext_bol()

```
int mtext_bol (
    MText * mt,
    int pos )
```

4.51.2.9 mtext_eol()

```
int mtext_eol (
    MText * mt,
    int pos )
```

4.51.2.10 mtext_wseg_fini()

```
void mtext_wseg_fini ( )
```

4.51.2.11 mtext_word_segment()

```
int mtext_word_segment (
    MText * mt,
    int pos,
    int * from,
    int * to )
```

4.52 plist.c ファイル

関数

- **MPlist * mplist** (void)
プロパティリストオブジェクトを作る.
- **MPlist * mplist_copy** (MPlist *plist)
プロパティリストをコピーする.
- **MPlist * mplist_put** (MPlist *plist, MSymbol key, void *val)
プロパティリスト中のプロパティの値を設定する.
- **void * mplist_get** (MPlist *plist, MSymbol key)
プロパティリスト中のプロパティの値を得る.
- **MPlist * mplist_put_func** (MPlist *plist, MSymbol key, M17NFunc func)
プロパティリスト中のプロパティに関数ポインタである値を設定する.
- **M17NFunc mplist_get_func** (MPlist *plist, MSymbol key)
プロパティリストからプロパティの関数ポインタである値を得る.
- **MPlist * mplist_add** (MPlist *plist, MSymbol key, void *val)
プロパティリスト末尾にプロパティを追加する.
- **MPlist * mplist_push** (MPlist *plist, MSymbol key, void *val)
プロパティリストの先頭にプロパティを挿入する.
- **void * mplist_pop** (MPlist *plist)
プロパティリストの先頭からプロパティを削除する.
- **MPlist * mplist_find_by_key** (MPlist *plist, MSymbol key)
プロパティリスト中から指定のキーを持つプロパティを探す.
- **MPlist * mplist_find_by_value** (MPlist *plist, void *val)

プロパティリストの中から指定の値を持つプロパティを探す。

- `MPList * mplist_next (MPList *plist)`
プロパティリストの次の部分リストを返す。
- `MPList * mplist_set (MPList *plist, MSymbol key, void *val)`
プロパティリストの最初のプロパティを設定する。
- `int mplist_length (MPList *plist)`
プロパティリストの長さを返す。
- `MSymbol mplist_key (MPList *plist)`
プロパティリスト中の最初のプロパティのキーを返す。
- `void * mplist_value (MPList *plist)`
プロパティリスト中の最初のプロパティの値を返す。
- `MPList * mplist_deserialize (MText *mt)`
M-text をデシリアライズしてプロパティリストを作る。

- `MPList * mdebug_dump_plist (MPList *plist, int indent)`
プロパティリストをダンプする。

変数

- `MSymbol Minteger`
"integer" を名前として持つシンボル。
- `MSymbol Mplist`
"plist" を名前として持つシンボル。
- `MSymbol Mtext`
"mtext" を名前として持つシンボル。

4.52.1 関数詳解

4.52.1.1 mdebug_dump_plist()

```
MPList* mdebug_dump_plist (
    MPList * plist,
    int indent )
```

プロパティリストをダンプする。

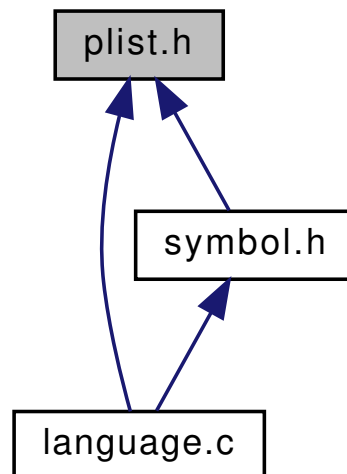
関数 `mdebug_dump_plist()` はプロパティリスト `plist` を標準エラー出力もしくは環境変数 `MDEBUG_DUMP_FONT` で指定されたファイルに人間に可読な形で印刷する。`indent` は2行目以降のインデントを指定する。

戻り値:

この関数は `plist` を返す。

4.53 plist.h ファイル

被依存関係図:



データ構造

- struct **MPlist**

プロパティリスト・オブジェクトの型宣言.

マクロ定義

- #define **MPLIST_KEY**(plist) ((plist)->key)
- #define **MPLIST_VAL**(plist) ((plist)->val.pointer)
- #define **MPLIST_FUNC**(plist) ((plist)->val.func)
- #define **MPLIST_NEXT**(plist) ((plist)->next)
- #define **MPLIST_TAIL_P**(plist) ((plist)->key == **Mnil**)
- #define **MPLIST_SYMBOL_P**(plist) (**MPLIST_KEY** (plist) == **Msymbol**)
- #define **MPLIST_STRING_P**(plist) (**MPLIST_KEY** (plist) == **Mstring**)
- #define **MPLIST_MTEXT_P**(plist) (**MPLIST_KEY** (plist) == **Mtext**)
- #define **MPLIST_INTEGER_P**(plist) (**MPLIST_KEY** (plist) == **Minteger**)
- #define **MPLIST_PLIST_P**(plist) (**MPLIST_KEY** (plist) == **Mplist**)
- #define **MPLIST_NESTED_P**(plist) ((plist)->control.flag & 1)
- #define **MPLIST_SET_NESTED_P**(plist) ((plist)->control.flag |= 1)
- #define **MPLIST_VAL_FUNC_P**(plist) ((plist)->control.flag & 2)
- #define **MPLIST_SET_VAL_FUNC_P**(plist) ((plist)->control.flag |= 2)
- #define **MPLIST_SYMBOL**(plist) ((**Msymbol**) **MPLIST_VAL** (plist))
- #define **MPLIST_STRING**(plist) ((char *) **MPLIST_VAL** (plist))
- #define **MPLIST_MTEXT**(plist) ((**Mtext** *) **MPLIST_VAL** (plist))
- #define **MPLIST_INTEGER**(plist) ((int) **MPLIST_VAL** (plist))
- #define **MPLIST_PLIST**(plist) ((**Mplist** *) **MPLIST_VAL** (plist))
- #define **MPLIST_FIND**(plist, key)
- #define **MPLIST_DO**(elt, plist) for ((elt) = (plist); ! **MPLIST_TAIL_P** (elt); (elt) = **MPLIST_NEXT** (elt))
- #define **MPLIST_LENGTH**(plist)

- #define `MPLIST_ADD_PLIST`(PLIST, KEY, VAL) `MPLIST_SET_NESTED_P` (`mplist_add` ((PLIST), (KEY), (VAL)))
- #define `MPLIST_PUSH_PLIST`(PLIST, KEY, VAL) `MPLIST_SET_NESTED_P` (`mplist_push` ((PLIST), (KEY), (VAL)))
- #define `MPLIST_PUT_PLIST`(PLIST, KEY, VAL) `MPLIST_SET_NESTED_P` (`mplist_put` ((PLIST), (KEY), (VAL)))

関数

- `MPlist *` `mplist_from_file` (`FILE *fp`, `MPlist *keys`)
- `MPlist *` `mplist_from_plist` (`MPlist *plist`)
- `MPlist *` `mplist_from_alist` (`MPlist *plist`)
- `MPlist *` `mplist_from_string` (`unsigned char *str`, `int n`)
- `int` `mplist_serialize` (`MText *mt`, `MPlist *plist`, `int pretty`)
- `MPlist *` `mplist_conc` (`MPlist *plist`, `MPlist *tail`)
- `void` `mplist_pop_unref` (`MPlist *plist`)
- `MPlist *` `mplist_assq` (`MPlist *plist`, `MSymbol key`)

変数

- `unsigned char` `hex_mnemonic` [256]
- `unsigned char` `escape_mnemonic` [256]

4.53.1 マクロ定義詳解

4.53.1.1 MPLIST_KEY

```
#define MPLIST_KEY(  
    plist ) ((plist)->key)
```

4.53.1.2 MPLIST_VAL

```
#define MPLIST_VAL(  
    plist ) ((plist)->val.pointer)
```

4.53.1.3 MPLIST_FUNC

```
#define MPLIST_FUNC(  
    plist ) ((plist)->val.func)
```


4.53.1.4 MPLIST_NEXT

```
#define MPLIST_NEXT(  
    plist ) ((plist)->next)
```

4.53.1.5 MPLIST_TAIL_P

```
#define MPLIST_TAIL_P(  
    plist ) ((plist)->key == Mnil)
```

4.53.1.6 MPLIST_SYMBOL_P

```
#define MPLIST_SYMBOL_P(  
    plist ) (MPLIST_KEY (plist) == Msymbol)
```

4.53.1.7 MPLIST_STRING_P

```
#define MPLIST_STRING_P(  
    plist ) (MPLIST_KEY (plist) == Mstring)
```

4.53.1.8 MPLIST_MTEXT_P

```
#define MPLIST_MTEXT_P(  
    plist ) (MPLIST_KEY (plist) == Mtext)
```

4.53.1.9 MPLIST_INTEGER_P

```
#define MPLIST_INTEGER_P(  
    plist ) (MPLIST_KEY (plist) == Minteger)
```

4.53.1.10 MPLIST_PLIST_P

```
#define MPLIST_PLIST_P(  
    plist ) (MPLIST_KEY (plist) == Mplist)
```

4.53.1.11 MPLIST_NESTED_P

```
#define MPLIST_NESTED_P(  
    plist )    ((plist)->control.flag & 1)
```

4.53.1.12 MPLIST_SET_NESTED_P

```
#define MPLIST_SET_NESTED_P(  
    plist )    ((plist)->control.flag |= 1)
```

4.53.1.13 MPLIST_VAL_FUNC_P

```
#define MPLIST_VAL_FUNC_P(  
    plist )    ((plist)->control.flag & 2)
```

4.53.1.14 MPLIST_SET_VAL_FUNC_P

```
#define MPLIST_SET_VAL_FUNC_P(  
    plist )    ((plist)->control.flag |= 2)
```

4.53.1.15 MPLIST_SYMBOL

```
#define MPLIST_SYMBOL(  
    plist )    ((MSymbol) MPLIST_VAL (plist))
```

4.53.1.16 MPLIST_STRING

```
#define MPLIST_STRING(  
    plist )    ((char *) MPLIST_VAL (plist))
```

4.53.1.17 MPLIST_MTEXT

```
#define MPLIST_MTEXT(  
    plist )    ((MText *) MPLIST_VAL (plist))
```

4.53.1.18 MPLIST_INTEGER

```
#define MPLIST_INTEGER(
    plist ) ((int) MPLIST_VAL (plist))
```

4.53.1.19 MPLIST_PLIST

```
#define MPLIST_PLIST(
    plist ) ((Mplist *) MPLIST_VAL (plist))
```

4.53.1.20 MPLIST_FIND

```
#define MPLIST_FIND(
    plist,
    key )
```

値:

```
do {
    while (! MPLIST_TAIL_P (plist) && MPLIST_KEY (plist) != (key)) \
        (plist) = (plist)->next;
} while (0)
```

4.53.1.21 MPLIST_DO

```
#define MPLIST_DO(
    elt,
    plist ) for ((elt) = (plist); ! MPLIST_TAIL_P (elt); (elt) = MPLIST_NEXT (elt))
```

4.53.1.22 MPLIST_LENGTH

```
#define MPLIST_LENGTH(
    plist )
```

値:

```
(MPLIST_TAIL_P (plist) ? 0
 : MPLIST_TAIL_P ((plist)->next) ? 1
 : MPLIST_TAIL_P ((plist)->next->next) ? 2
 : mplist_length (plist))
```

4.53.1.23 MPLIST_ADD_PLIST

```
#define MPLIST_ADD_PLIST(  
    PLIST,  
    KEY,  
    VAL )  MPLIST_SET_NESTED_P (mplist_add ((PLIST), (KEY), (VAL)))
```

4.53.1.24 MPLIST_PUSH_PLIST

```
#define MPLIST_PUSH_PLIST(  
    PLIST,  
    KEY,  
    VAL )  MPLIST_SET_NESTED_P (mplist_push ((PLIST), (KEY), (VAL)))
```

4.53.1.25 MPLIST_PUT_PLIST

```
#define MPLIST_PUT_PLIST(  
    PLIST,  
    KEY,  
    VAL )  MPLIST_SET_NESTED_P (mplist_put ((PLIST), (KEY), (VAL)))
```

4.53.2 関数詳解

4.53.2.1 mplist_from_file()

```
MPlist* mplist_from_file (  
    FILE * fp,  
    MPlist * keys )
```

4.53.2.2 mplist_from_plist()

```
MPlist* mplist_from_plist (  
    MPlist * plist )
```

4.53.2.3 mplist_from_alist()

```
MPlist* mplist_from_alist (
    MPlist * plist )
```

4.53.2.4 mplist_from_string()

```
MPlist* mplist_from_string (
    unsigned char * str,
    int n )
```

4.53.2.5 mplist_serialize()

```
int mplist_serialize (
    MText * mt,
    MPlist * plist,
    int pretty )
```

4.53.2.6 mplist_conc()

```
MPlist* mplist_conc (
    MPlist * plist,
    MPlist * tail )
```

4.53.2.7 mplist_pop_unref()

```
void mplist_pop_unref (
    MPlist * plist )
```

4.53.2.8 mplist_assq()

```
MPlist* mplist_assq (
    MPlist * plist,
    MSymbol key )
```

4.53.3 変数詳解

4.53.3.1 hex_mnemonic

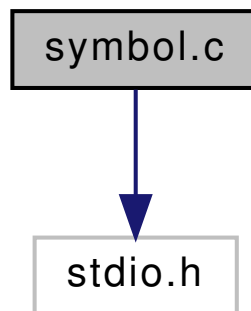
```
unsigned char hex_mnemonic[256] [extern]
```

4.53.3.2 escape_mnemonic

```
unsigned char escape_mnemonic[256] [extern]
```

4.54 symbol.c ファイル

symbol.c の依存先関係図:



関数

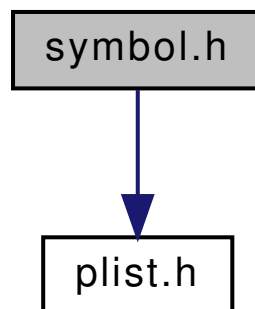
- **MSymbol msymbol** (const char *name)
シンボルを得る.
- **MSymbol msymbol_as_managing_key** (const char *name)
管理キーを作る.
- int **msymbol_is_managing_key** (MSymbol symbol)
- **MSymbol msymbol_exist** (const char *name)
指定された名前を持つシンボルを探す.
- char * **msymbol_name** (MSymbol symbol)
シンボルの名前を得る.
- int **msymbol_put** (MSymbol symbol, MSymbol key, void *val)
シンボルプロパティに値を設定する.
- void * **msymbol_get** (MSymbol symbol, MSymbol key)
シンボルプロパティの値を得る.
- int **msymbol_put_func** (MSymbol symbol, MSymbol key, M17NFunc func)
シンボルプロパティの値 (関数ポインタ) を設定する.
- M17NFunc **msymbol_get_func** (MSymbol symbol, MSymbol key)
シンボルプロパティの値 (関数ポインタ) を得る.
- **MSymbol mdebug_dump_symbol** (MSymbol symbol, int indent)
シンボルをダンプする.
- **MSymbol mdebug_dump_all_symbols** (int indent)
すべてのシンボル名をダンプする.

変数

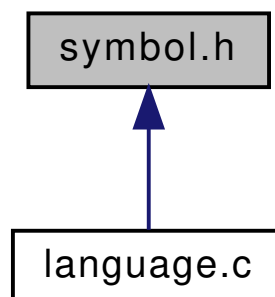
- [MSymbol Mnil](#)
"nil" を名前として持つシンボル.
- [MSymbol Mt](#)
"t" を名前として持つシンボル.
- [MSymbol Mstring](#)
"string" を名前として持つシンボル.
- [MSymbol Msymbol](#)
"symbol" を名前として持つシンボル.

4.55 symbol.h ファイル

symbol.h の依存先関係図:



被依存関係図:



データ構造

- `struct MSymbol`
シンボルの型宣言.

マクロ定義

- `#define MSYMBOL_NAME(sym) ((sym)->name)`
- `#define MSYMBOL_NAMELEN(sym) ((sym)->length - 1)`

関数

- `void msymbol_free_table ()`
- `MSymbol msymbol_with_len (const char *name, int len)`
- `MPlist * msymbol_list (MSymbol prop)`
- `MSymbol msymbol_canonicalize (MSymbol sym)`

変数

- `MTextPropSerializeFunc msymbol_serializer`
- `MTextPropDeserializeFunc msymbol_deserializer`

4.55.1 マクロ定義詳解

4.55.1.1 MSYMBOL_NAME

```
#define MSYMBOL_NAME(  
    sym ) ((sym)->name)
```

4.55.1.2 MSYMBOL_NAMELEN

```
#define MSYMBOL_NAMELEN(  
    sym ) ((sym)->length - 1)
```

4.55.2 関数詳解

4.55.2.1 msymbol_free_table()

```
void msymbol_free_table ( )
```


4.55.2.2 msymbol__with_len()

```
MSymbol msymbol__with_len (
    const char * name,
    int len )
```

4.55.2.3 msymbol__list()

```
MList* msymbol__list (
    MSymbol prop )
```

4.55.2.4 msymbol__canonicalize()

```
MSymbol msymbol__canonicalize (
    MSymbol sym )
```

4.55.3 変数詳解

4.55.3.1 msymbol__serializer

```
MTextPropSerializeFunc msymbol__serializer [extern]
```

4.55.3.2 msymbol__deserializer

```
MTextPropDeserializeFunc msymbol__deserializer [extern]
```

4.56 textprop.c ファイル

関数

- void * `mtext_get_prop` (MText *mt, int pos, MSymbol key)
テキストプロパティの一番上の値を得る。
- int `mtext_get_prop_values` (MText *mt, int pos, MSymbol key, void **values, int num)
テキストプロパティの値を複数個得る。
- int `mtext_get_prop_keys` (MText *mt, int pos, MSymbol **keys)
M-text の指定した位置のテキストプロパティのキーのリストを得る。
- int `mtext_put_prop` (MText *mt, int from, int to, MSymbol key, void *val)
- int `mtext_put_prop_values` (MText *mt, int from, int to, MSymbol key, void **values, int num)
同じキーのテキストプロパティを複数設定する。
- int `mtext_push_prop` (MText *mt, int from, int to, MSymbol key, void *val)
- int `mtext_pop_prop` (MText *mt, int from, int to, MSymbol key)
- int `mtext_prop_range` (MText *mt, MSymbol key, int pos, int *from, int *to, int deeper)
テキストプロパティが同じ値をとる範囲を調べる。
- MTextProperty * `mtext_property` (MSymbol key, void *val, int control_bits)
テキストプロパティを生成する。
- MText * `mtext_property_mtext` (MTextProperty *prop)
あるテキストプロパティを持つ M-text を返す。
- MSymbol `mtext_property_key` (MTextProperty *prop)
テキストプロパティのキーを返す。
- void * `mtext_property_value` (MTextProperty *prop)
テキストプロパティの値を返す。
- int `mtext_property_start` (MTextProperty *prop)
テキストプロパティの開始位置を返す。
- int `mtext_property_end` (MTextProperty *prop)
テキストプロパティの終了位置を返す。
- MTextProperty * `mtext_get_property` (MText *mt, int pos, MSymbol key)
一番上のテキストプロパティを得る。
- int `mtext_get_properties` (MText *mt, int pos, MSymbol key, MTextProperty **props, int num)
複数のテキストプロパティを得る。
- int `mtext_attach_property` (MText *mt, int from, int to, MTextProperty *prop)
M-text にテキストプロパティを付加する。
- int `mtext_detach_property` (MTextProperty *prop)
M-text からテキストプロパティを分離する。
- int `mtext_push_property` (MText *mt, int from, int to, MTextProperty *prop)
M-text にテキストプロパティをプッシュする。
- MText * `mtext_serialize` (MText *mt, int from, int to, MPlist *property_list)
- MText * `mtext_deserialize` (MText *mt)

変数

- MSymbol `Mtext_prop_serializer`
シリアライザ関数を指定するシンボル。
- MSymbol `Mtext_prop_deserializer`
デシリアライザ関数を指定するシンボル。

4.57 textprop.h ファイル

データ構造

- struct [MTextProperty](#)
テキストプロパティの型宣言.

マクロ定義

- #define [MTEXTPROP_START](#)(prop) (prop)->start
- #define [MTEXTPROP_END](#)(prop) (prop)->end
- #define [MTEXTPROP_KEY](#)(prop) (prop)->key
- #define [MTEXTPROP_VAL](#)(prop) (prop)->val

関数

- struct MTextPlist * [mtext__copy_plist](#) (struct MTextPlist *, int from, int to, [MText](#) *mt, int pos)
- void [mtext__free_plist](#) ([MText](#) *mt)
- void [mtext__adjust_plist_for_delete](#) ([MText](#) *, int, int)
- void [mtext__adjust_plist_for_insert](#) ([MText](#) *, int, int, struct MTextPlist *)
- void [mtext__adjust_plist_for_change](#) ([MText](#) *mt, int pos, int len1, int len2)
- void [dump_textplist](#) (struct MTextPlist *plist, int indent)

4.57.1 マクロ定義詳解

4.57.1.1 MTEXTPROP_START

```
#define MTEXTPROP_START(  
    prop ) (prop)->start
```

4.57.1.2 MTEXTPROP_END

```
#define MTEXTPROP_END(  
    prop ) (prop)->end
```

4.57.1.3 MTEXTPROP_KEY

```
#define MTEXTPROP_KEY(  
    prop ) (prop)->key
```

4.57.1.4 MTEXTPROP_VAL

```
#define MTEXTPROP_VAL(  
    prop ) (prop)->val
```

4.57.2 関数詳解

4.57.2.1 mtext_copy_plist()

```
struct MTextPlist* mtext_copy_plist (  
    struct MTextPlist * ,  
    int from,  
    int to,  
    MText * mt,  
    int pos )
```

4.57.2.2 mtext_free_plist()

```
void mtext_free_plist (  
    MText * mt )
```

4.57.2.3 mtext_adjust_plist_for_delete()

```
void mtext_adjust_plist_for_delete (  
    MText * ,  
    int ,  
    int )
```

4.57.2.4 mtext_adjust_plist_for_insert()

```
void mtext_adjust_plist_for_insert (  
    MText * ,  
    int ,  
    int ,  
    struct MTextPlist * )
```

4.57.2.5 mtext_adjust_plist_for_change()

```
void mtext_adjust_plist_for_change (
    MText * mt,
    int pos,
    int len1,
    int len2 )
```

4.57.2.6 dump_textplist()

```
void dump_textplist (
    struct MTextPlist * plist,
    int indent )
```


Index

- - internal.h, [375](#)
- absolute.filename
 - MDatabaseInfo, [213](#)
- active
 - MInputContext, [276](#)
- adjust_window
 - MDeviceDriver, [218](#)
- adjusted
 - MFLTGlyph, [245](#)
- advance_is_absolute
 - MFLTGlyphAdjustment, [246](#)
- align_head
 - MDrawControl, [219](#)
- allocated
 - MFLTGlyphString, [248](#)
 - MText, [309](#)
- anti_alias
 - MDrawControl, [220](#)
 - MGlyphString, [273](#)
- APPEND_GLYPH
 - internal-gui.h, [368](#)
- arg
 - MInputContext, [276](#)
 - MInputMethod, [290](#)
- as_image
 - MDrawControl, [219](#)
- ascent
 - MDrawGlyph, [226](#)
 - MFLTGlyph, [244](#)
 - MFrame, [265](#)
 - MGlyphString, [271](#)
 - MInputContext, [276](#)
 - MRealizedFace, [301](#)
 - MRealizedFont, [304](#)
- ascii_compatible
 - MCharset, [203](#)
- ascii_rface
 - MRealizedFace, [301](#)
- at_most
 - MConverter, [210](#)
- attach_count
 - MTextProperty, [311](#)
- average_width
 - MFrame, [265](#)
 - MRealizedFace, [302](#)
- MRealizedFont, [305](#)
- back
 - MFLTGlyphAdjustment, [246](#)
- background
 - MFrame, [264](#)
- base_face_list
 - MRealizedFace, [300](#)
- baseline_offset
 - MRealizedFont, [305](#)
- bc_cmds
 - MInputMethodInfo, [293](#)
- bc_vars
 - MInputMethodInfo, [293](#)
- bidirectional
 - MGlyph, [268](#)
- bom
 - MCodingInfoUTF, [209](#)
- box
 - MRealizedFace, [301](#)
- c
 - MConverter, [211](#)
 - MFLTGlyph, [243](#)
- cache_byte_pos
 - MText, [309](#)
- cache_char_pos
 - MText, [309](#)
- callback_list
 - MInputDriver, [287](#)
- candidate_from
 - MInputContext, [278](#)
- candidate_index
 - MInputContext, [278](#)
- candidate_list
 - MInputContext, [278](#)
- candidate_show
 - MInputContext, [279](#)
- candidate_to
 - MInputContext, [278](#)
- candidates_changed
 - MInputContext, [279](#)
- capability
 - MFont, [252](#)
- category
 - MGlyph, [268](#)
- CHAR_BYTES

- character.h, [317](#)
- CHAR_BYTES_AT
 - character.h, [317](#)
- CHAR_BYTES_BY_HEAD
 - character.h, [318](#)
- CHAR_HEAD_P
 - character.h, [322](#)
- CHAR_HEAD_P_UTF16
 - character.h, [322](#)
- CHAR_HEAD_P_UTF8
 - character.h, [322](#)
- CHAR_STRING
 - character.h, [322](#)
- CHAR_STRING_UTF16
 - character.h, [321](#)
- CHAR_STRING_UTF8
 - character.h, [321](#)
- CHAR_UNITS
 - character.h, [316](#)
- CHAR_UNITS_ASCII
 - character.h, [316](#)
- CHAR_UNITS_AT
 - character.h, [317](#)
- CHAR_UNITS_AT_UTF16
 - character.h, [317](#)
- CHAR_UNITS_AT_UTF8
 - character.h, [317](#)
- CHAR_UNITS_BY_HEAD
 - character.h, [318](#)
- CHAR_UNITS_BY_HEAD_UTF16
 - character.h, [318](#)
- CHAR_UNITS_BY_HEAD_UTF8
 - character.h, [318](#)
- CHAR_UNITS_UTF16
 - character.h, [316](#)
- CHAR_UNITS_UTF32
 - character.h, [316](#)
- CHAR_UNITS_UTF8
 - character.h, [316](#)
- character.c, [313](#)
- character.h, [314](#)
 - CHAR_BYTES, [317](#)
 - CHAR_BYTES_AT, [317](#)
 - CHAR_BYTES_BY_HEAD, [318](#)
 - CHAR_HEAD_P, [322](#)
 - CHAR_HEAD_P_UTF16, [322](#)
 - CHAR_HEAD_P_UTF8, [322](#)
 - CHAR_STRING, [322](#)
 - CHAR_STRING_UTF16, [321](#)
 - CHAR_STRING_UTF8, [321](#)
 - CHAR_UNITS, [316](#)
 - CHAR_UNITS_ASCII, [316](#)
 - CHAR_UNITS_AT, [317](#)
 - CHAR_UNITS_AT_UTF16, [317](#)
 - CHAR_UNITS_AT_UTF8, [317](#)
 - CHAR_UNITS_BY_HEAD, [318](#)
- CHAR_UNITS_BY_HEAD_UTF16, [318](#)
- CHAR_UNITS_BY_HEAD_UTF8, [318](#)
- CHAR_UNITS_UTF16, [316](#)
- CHAR_UNITS_UTF32, [316](#)
- CHAR_UNITS_UTF8, [316](#)
- ISALNUM, [323](#)
- ISUPPER, [323](#)
- MAX_UNICODE_CHAR_BYTES, [315](#)
- MAX_UTF8_CHAR_BYTES, [315](#)
- mchar_define_prop, [323](#)
- STRING_CHAR, [319](#)
- STRING_CHAR_ADVANCE, [320](#)
- STRING_CHAR_ADVANCE_UTF16, [320](#)
- STRING_CHAR_ADVANCE_UTF8, [319](#)
- STRING_CHAR_AND_BYTES, [321](#)
- STRING_CHAR_AND_UNITS, [321](#)
- STRING_CHAR_AND_UNITS_UTF16, [320](#)
- STRING_CHAR_AND_UNITS_UTF8, [320](#)
- STRING_CHAR_UTF16, [319](#)
- STRING_CHAR_UTF8, [318](#)
- TOLOWER, [322](#)
- TOUPPER, [322](#)
- UINT_SIZE, [315](#)
- UNIT_BYTES, [315](#)
- USHORT_SIZE, [315](#)
- charset.c, [323](#)
- charset.h, [325](#)
 - CODE_POINT_TO_INDEX, [326](#)
 - DECODE_CHAR, [327](#)
 - ENCODE_CHAR, [327](#)
 - INDEX_TO_CODE_POINT, [327](#)
 - ISO_MAX_CHARS, [328](#)
 - ISO_MAX_DIMENSION, [328](#)
 - ISO_MAX_FINAL, [328](#)
 - MCHARSET, [326](#)
 - mcharset_ascii, [330](#)
 - mcharset_binary, [330](#)
 - mcharset_cache, [329](#)
 - mcharset_decode_char, [329](#)
 - mcharset_encode_char, [329](#)
 - mcharset_find, [329](#)
 - mcharset_iso_2022_table, [330](#)
 - mcharset_load_from_database, [329](#)
 - mcharset_m17n, [330](#)
 - mcharset_unicode, [330](#)
 - MCHARSET_ISO_2022, [328](#)
 - mcharset_method, [328](#)
 - MCHARSET_METHOD_DEFERRED, [329](#)
 - MCHARSET_METHOD_MAP, [329](#)
 - MCHARSET_METHOD_MAX, [329](#)
 - MCHARSET_METHOD_OFFSET, [329](#)
 - MCHARSET_METHOD_SUBSET, [329](#)
 - MCHARSET_METHOD_SUPERSET, [329](#)
- charsets
 - MCharsetISO2022Table, [207](#)
- chartab.c, [331](#)

- mdebug_dump_chartab, 331
- chartab.h, 332
 - mchartable_lookup, 332
- check_capability
 - MFontDriver, 257
- check_otf
 - MFLTFont, 241
 - MFontDriver, 258
- classified
 - MCharsetISO2022Table, 207
- client
 - MInputGUIArgIC, 288
- client_win
 - MInputXIMArgIC, 295
- clip_region
 - MDrawControl, 223
- close
 - MDeviceDriver, 215
 - MFontDriver, 257
- close_im
 - MInputDriver, 286
- cmds
 - MInputMethodInfo, 292
- code
 - MFLTGlyph, 243
- CODE_POINT_TO_INDEX
 - charset.h, 326
- code_range
 - MCharset, 202
- code_range_mask
 - MCharset, 203
- code_range_min_code
 - MCharset, 203
- code_unit_bits
 - MCodingInfoUTF, 208
- coding.c, 332
- coding.h, 334
 - mcoding_load_from_database, 335
 - mconv_register_charset_coding, 335
- color
 - MFaceHLineProp, 239
- color_bottom
 - MFaceBoxProp, 236
- color_left
 - MFaceBoxProp, 236
- color_right
 - MFaceBoxProp, 236
- color_top
 - MFaceBoxProp, 236
- COMBINING_CODE_ADD_X
 - internal-flt.h, 364
- COMBINING_CODE_ADD_Y
 - internal-flt.h, 365
- COMBINING_CODE_BASE_X
 - internal-flt.h, 364
- COMBINING_CODE_BASE_Y
 - internal-flt.h, 364
- COMBINING_CODE_OFF_X
 - internal-flt.h, 364
- COMBINING_CODE_OFF_Y
 - internal-flt.h, 364
- commit_key_head
 - MInputContextInfo, 282
- configured_cmds
 - MInputMethodInfo, 293
- configured_vars
 - MInputMethodInfo, 293
- control
 - MDrawTextItem, 233
 - MFace, 234
 - MFontCapability, 254
 - MFrame, 264
 - MGlyphString, 273
 - MPList, 298
 - MText, 308
 - MTextProperty, 311
- count
 - M17NObjectArray, 199
- counts
 - M17NObjectRecord, 201
- coverage
 - MText, 308
- create_ic
 - MInputDriver, 286
- create_window
 - MDeviceDriver, 217
- cursor_bidi
 - MDrawControl, 222
- cursor_pos
 - MDrawControl, 222
 - MInputContext, 278
- cursor_pos_changed
 - MInputContext, 278
- cursor_width
 - MDrawControl, 222
- data
 - MText, 309
- database.c, 335
- database.h, 336
 - M17NDIR, 336
 - mdatabase_check, 338
 - mdatabase_dir_list, 339
 - mdatabase_file, 338
 - mdatabase_find_file, 338
 - mdatabase_load_charset_func, 339
 - mdatabase_load_for_keys, 337
 - mdatabase_lock, 338
 - mdatabase_props, 339
 - mdatabase_save, 338
 - mdatabase_unlock, 338
 - mdatabase_update, 337

- MDatabaseStatus, 337
 - MDB.STATUS_AUTO, 337
 - MDB.STATUS_AUTO_WILDCARD, 337
 - MDB.STATUS_DISABLED, 337
 - MDB.STATUS_EXPLICIT, 337
 - MDB.STATUS_OUTDATED, 337
 - MDB.STATUS_UPDATED, 337
 - PATH_MAX, 337
 - PATH_SEPARATOR, 337
- db
 - MInputXIMArgIM, 296
- dbdata.txt, 339
- dbformat.txt, 339
- dbl
 - MConverter, 211
- dbtutorial.txt, 339
- DECODE_CHAR
 - charset.h, 327
- decoder
 - MCharset, 204
- DELETE_GLYPH
 - internal-gui.h, 368
- delta
 - MDrawTextItem, 233
- descent
 - MDrawGlyph, 226
 - MFLTGlyph, 244
 - MFrame, 265
 - MGlyphString, 272
 - MInputContext, 276
 - MRealizedFace, 301
 - MRealizedFont, 304
- description
 - MInputMethodInfo, 293
- designations
 - MCodingInfoISO2022, 208
- destroy_ic
 - MInputDriver, 286
- destroy_window
 - MDeviceDriver, 217
- device
 - MFrame, 265
- device_open
 - m17n-X.c, 418
- device_type
 - MFrame, 266
- dimension
 - MCharset, 202
- disable_caching
 - MDrawControl, 223
- disable_overlapping_adjustment
 - MDrawControl, 220
- display
 - MInputXIMArgIM, 296
- dpi
 - MFrame, 266
- draw.c, 339
- draw_box
 - MDeviceDriver, 216
- draw_empty_boxes
 - MDeviceDriver, 216
- draw_hline
 - MDeviceDriver, 216
- draw_points
 - MDeviceDriver, 216
- drive_otf
 - MFLTFont, 241
 - MFontDriver, 258
- driver
 - MFrame, 266
 - MInputMethod, 290
 - MRealizedFont, 303
- dump_region
 - MDeviceDriver, 217
- dump_textplist
 - textprop.h, 449
- enable_bidi
 - MDrawControl, 220
- enabled
 - MGlyph, 268
- encapsulate
 - MFontDriver, 257
- encapsulating
 - MRealizedFont, 304
- ENCODE_CHAR
 - charset.h, 327
- encode_char
 - MFontDriver, 257
- encoded
 - MFLTGlyph, 244
- encoder
 - MCharset, 204
- encoding
 - MFont, 252
- end
 - MTextProperty, 311
- endian
 - MCodingInfoUTF, 209
- escape_mnemonic
 - plist.h, 442
- exprog.txt, 340
- externals
 - MInputMethodInfo, 294
- extra
 - MInputMethodInfo, 292
- face
 - MDrawTextItem, 233
 - MFrame, 264
 - MRealizedFace, 300
- face.c, 340

- face.h, 342
 - mface_default, 345
 - mface_for_chars, 344
 - mface_free_realized, 344
 - mface_realize, 344
 - mface_update_frame_face, 344
 - MFACE_ADSTYLE, 343
 - MFACE_BACKGROUND, 344
 - MFACE_BOX, 344
 - MFACE_FAMILY, 343
 - MFACE_FONTSET, 344
 - MFACE_FOREGROUND, 344
 - MFACE_FOUNDRY, 343
 - MFACE_HLINE, 344
 - MFACE_HOOK_ARG, 344
 - MFACE_PROPERTY_MAX, 344
 - MFACE_RATIO, 344
 - MFACE_SIZE, 344
 - MFACE_STRETCH, 343
 - MFACE_STYLE, 343
 - MFACE_VIDEOMODE, 344
 - MFACE_WEIGHT, 343
 - MFaceProperty, 343
- fallbacks
 - MInputContextInfo, 284
- family
 - MFLTFont, 240
- fdl.txt, 345
- features
 - MFLTOtfSpec, 249
 - MFontCapability, 255
- file
 - MFont, 252
- filename
 - MDatabaseInfo, 213
- fill_space
 - MDeviceDriver, 215
- filler
 - M17NObjectHead, 200
- filter
 - MInputDriver, 286
- final_byte
 - MCharset, 204
- find_metric
 - MFontDriver, 256
- fixed_width
 - MDrawControl, 220
- flag
 - M17NObject, 198
- flags
 - MCodingInfoISO2022, 208
- FLT API, 135
 - mdebug_dump_ftl, 138
 - MFLT, 137
 - mflt_coverage, 138
 - mflt_dump_gstring, 138
 - mflt_enable_new_feature, 139
 - mflt_find, 137
 - mflt_font_id, 139
 - mflt_get, 137
 - mflt_iterate_otf_feature, 139
 - mflt_name, 137
 - mflt_run, 138
 - mflt_try_otf, 139
- focus
 - MInputGUIArgIC, 289
- focus_win
 - MInputXIMArgIC, 295
- following_text
 - MInputContextInfo, 283
- font
 - MDrawGlyph, 226
 - MDrawGlyphInfo, 229
 - MFLTFontForRealized, 242
 - MFontScore, 262
 - MFrame, 264
 - MRealizedFace, 300
 - MRealizedFont, 303
- font.c, 345
 - mdebug_dump_font, 347
- font.h, 347
 - FONT_PROPERTY, 349
 - Mapple_roman, 356
 - mfont_check_capability, 355
 - mfont_encode_char, 353
 - mfont_encoding_list, 355
 - mfont_ftl_encode_char, 355
 - mfont_ftl_fini, 351
 - mfont_ftl_init, 351
 - mfont_ftl_run, 355
 - mfont_free_realized, 352
 - mfont_get_capability, 355
 - mfont_get_glyph_id, 353
 - mfont_get_metric, 354
 - mfont_get_metrics, 354
 - mfont_has_char, 352
 - mfont_list, 353
 - mfont_match_p, 352
 - mfont_merge, 352
 - mfont_open, 353
 - mfont_parse_name_into_font, 354
 - mfont_property_table, 356
 - mfont_select, 353
 - mfont_set_property, 354
 - mfont_set_spec_from_face, 352
 - mfont_set_spec_from_plist, 352
 - mfont_split_name, 354
 - MFONT_ADSTYLE, 350
 - MFONT_FAMILY, 350
 - MFONT_FOUNDRY, 350
 - MFONT_INIT, 349
 - MFONT_OTT_GPOS, 351

- MFONT_OTT_GSUB, [351](#)
- MFONT_OTT_MAX, [351](#)
- MFONT_PROPERTY_MAX, [350](#)
- MFONT_REGISTRY, [350](#)
- MFONT_RESY, [350](#)
- MFONT_SIZE, [350](#)
- MFONT_SOURCE_FT, [351](#)
- MFONT_SOURCE_UNDECIDED, [351](#)
- MFONT_SOURCE_X, [351](#)
- MFONT_SPACING, [350](#)
- MFONT_SPACING_CHARCELL, [351](#)
- MFONT_SPACING_MONO, [351](#)
- MFONT_SPACING_PROPORTIONAL, [351](#)
- MFONT_SPACING_UNDECIDED, [351](#)
- MFONT_STRETCH, [350](#)
- MFONT_STYLE, [350](#)
- MFONT_TYPE_FAILURE, [350](#)
- MFONT_TYPE_OBJECT, [350](#)
- MFONT_TYPE_REALIZED, [350](#)
- MFONT_TYPE_SPEC, [350](#)
- MFONT_WEIGHT, [350](#)
- MFontEncoding, [349](#)
- MFontOpenTypeTable, [351](#)
- MFontProperty, [350](#)
- MFontSource, [350](#)
- MFontSpacing, [351](#)
- MFontType, [350](#)
- Miso10646_1, [356](#)
- Miso8859_1, [356](#)
- Mlayouter, [356](#)
- Municode_bmp, [356](#)
- Municode_full, [356](#)
- OTF_Tag, [349](#)
- font_driver_list
 - MFrame, [266](#)
- FONT_PROPERTY
 - font.h, [349](#)
- font_type
 - MDrawGlyph, [226](#)
- fontp
 - MDrawGlyph, [226](#)
 - MRealizedFont, [305](#)
- fonts
 - MFontList, [259](#)
- fontset.c, [357](#)
 - mdebug_dump_fontset, [357](#)
- fontset.h, [357](#)
 - mfont_free_realized_fontset, [358](#)
 - mfont_lookup_fontset, [358](#)
 - mfont_realize_fontset, [358](#)
 - mfontset_get_font, [358](#)
- fontsize
 - MInputContext, [276](#)
- for_full_width
 - MFont, [251](#)
- foreground
 - MFrame, [264](#)
- format
 - MDrawControl, [221](#)
 - MText, [308](#)
- frame
 - MGlyphString, [270](#)
 - MInputGUIArgIC, [288](#)
 - MRealizedFace, [300](#)
 - MRealizedFont, [303](#)
- frame_list
 - MFace, [235](#)
- free_realized_face
 - MDeviceDriver, [215](#)
- free_region
 - MDeviceDriver, [217](#)
- freer
 - M17NObject, [198](#)
 - M17NObjectRecord, [200](#)
- from
 - MDrawGlyph, [225](#)
 - MDrawGlyphInfo, [228](#)
 - MFLTGlyph, [243](#)
 - MGlyphString, [271](#)
- fully_loaded
 - MCharset, [205](#)
- func
 - MPList, [298](#)
- g
 - MGlyph, [267](#)
- get_glyph_id
 - MFLTFont, [240](#)
- get_metrics
 - MFLTFont, [240](#)
- get_prop
 - MDeviceDriver, [215](#)
- GLYPH_ANCHOR
 - internal-gui.h, [369](#)
- GLYPH_BOX
 - internal-gui.h, [369](#)
- glyph_category
 - internal-gui.h, [369](#)
- GLYPH_CATEGORY_FORMATTER
 - internal-gui.h, [370](#)
- GLYPH_CATEGORY_MODIFIER
 - internal-gui.h, [370](#)
- GLYPH_CATEGORY_NORMAL
 - internal-gui.h, [370](#)
- GLYPH_CHAR
 - internal-gui.h, [369](#)
- glyph_code
 - MDrawGlyph, [225](#)
- GLYPH_INDEX
 - internal-gui.h, [367](#)
- GLYPH_PAD
 - internal-gui.h, [369](#)

- glyph_size
 - MFLTGlyphString, 247
- GLYPH_SPACE
 - internal-gui.h, 369
- glyph_type
 - internal-gui.h, 369
- GLYPH_TYPE_MAX
 - internal-gui.h, 369
- glyphs
 - MFLTGlyphString, 248
 - MGlyphString, 271
- GUI API, 140
- has_char
 - FontDriver, 256
- head
 - MGlyphString, 270
- height
 - MDrawMetric, 231
 - MGlyphString, 271
- hex_mnemonic
 - plist.h, 441
- hline
 - MRealizedFace, 301
- hook
 - MFace, 235
- id
 - MRealizedFont, 303
- ignore_formatting_char
 - MDrawControl, 220
- im
 - MInputContext, 275
- inc
 - M17NObjectArray, 199
 - M17NObjectRecord, 201
 - MCharsetISO2022Table, 206
 - MFontPropertyTable, 261
 - MGlyphString, 270
 - MInputContextInfo, 281
- indent
 - MGlyphString, 273
- INDEX_TO_CODE_POINT
 - charset.h, 327
- info
 - MInputContext, 277
 - MInputMethod, 290
 - MRealizedFace, 302
 - MRealizedFont, 304
- INIT_GLYPH
 - internal-gui.h, 367
- initial_invocation
 - MCodingInfoISO2022, 207
- inner_hmargin
 - MFaceBoxProp, 236
- inner_vmargin
 - MFaceBoxProp, 237
- input-gui.c, 358
- input.c, 359
- input.h, 361
 - MIMInputStack, 363
 - MIMMap, 363
 - MIMState, 362
 - minput_char_to_key, 363
 - MINPUT_KEY_ALT_MODIFIER, 362
 - MINPUT_KEY_ALTGR_MODIFIER, 362
 - MINPUT_KEY_CONTROL_MODIFIER, 362
 - MINPUT_KEY_HYPER_MODIFIER, 362
 - MINPUT_KEY_META_MODIFIER, 362
 - MINPUT_KEY_SHIFT_MODIFIER, 361
 - MINPUT_KEY_SUPER_MODIFIER, 362
- input_style
 - MInputXIMArgIC, 295
- INSERT_GLYPH
 - internal-gui.h, 368
- internal
 - MFLTFont, 241
 - MFLTGlyph, 245
- internal-flt.h, 363
 - COMBINING_CODE_ADD_X, 364
 - COMBINING_CODE_ADD_Y, 365
 - COMBINING_CODE_BASE_X, 364
 - COMBINING_CODE_BASE_Y, 364
 - COMBINING_CODE_OFF_X, 364
 - COMBINING_CODE_OFF_Y, 364
 - MAKE_COMBINING_CODE, 364
 - Mcombining, 365
 - PACK_OTF_TAG, 365
- internal-gui.h, 365
 - APPEND_GLYPH, 368
 - DELETE_GLYPH, 368
 - GLYPH_ANCHOR, 369
 - GLYPH_BOX, 369
 - glyph_category, 369
 - GLYPH_CATEGORY_FORMATTER, 370
 - GLYPH_CATEGORY_MODIFIER, 370
 - GLYPH_CATEGORY_NORMAL, 370
 - GLYPH_CHAR, 369
 - GLYPH_INDEX, 367
 - GLYPH_PAD, 369
 - GLYPH_SPACE, 369
 - glyph_type, 369
 - GLYPH_TYPE_MAX, 369
 - INIT_GLYPH, 367
 - INSERT_GLYPH, 368
 - M_CHECK_READABLE, 367
 - M_CHECK_WRITABLE, 367
 - MDEVICE_SUPPORT_INPUT, 369
 - MDEVICE_SUPPORT_OUTPUT, 369
 - MDeviceType, 369
 - mdraw_fini, 370
 - mdraw_init, 370

- mface_fini, 370
- mface_init, 370
- mfont_fini, 370
- mfont_fontset_fini, 371
- mfont_fontset_init, 371
- mfont_init, 370
- MGLYPH, 367
- minput_win_fini, 371
- minput_win_init, 371
- Mlatin, 371
- MRealizedFontset, 369
- REPLACE_GLYPHS, 368
- internal.h, 372
 - _, 375
 - m17n_core_initialized, 394
 - m17n_gui_initialized, 394
 - m17n_shell_initialized, 394
 - M17N_OBJECT, 381
 - M17N_OBJECT_ADD_ARRAY, 383
 - M17N_OBJECT_REF, 381
 - M17N_OBJECT_REF_NTIMES, 382
 - M17N_OBJECT_REGISTER, 383
 - M17N_OBJECT_UNREF, 382
 - M17N_OBJECT_UNREGISTER, 383
 - M_CHECK_CHAR, 376
 - M_CHECK_POS, 383
 - M_CHECK_POS_NCHARS, 384
 - M_CHECK_POS_X, 384
 - M_CHECK_RANGE, 384
 - M_CHECK_RANGE_X, 384
 - M_CHECK_READONLY, 385
 - mchar_fini, 393
 - mchar_init, 392
 - mcharset_fini, 392
 - mcharset_init, 392
 - mchartable_fini, 391
 - mchartable_init, 391
 - mcoding_fini, 392
 - mcoding_init, 392
 - mdatabase_fini, 392
 - mdatabase_init, 392
 - mdebug_add_object_array, 389
 - mdebug_flags, 394
 - mdebug_output, 394
 - mdebug_pop_time, 390
 - mdebug_print_time, 390
 - mdebug_push_time, 390
 - mdebug_register_object, 389
 - mdebug_unregister_object, 390
 - MDEBUG_ALL, 389
 - MDEBUG_CHARSET, 389
 - MDEBUG_CODING, 389
 - MDEBUG_DATABASE, 389
 - MDEBUG_DUMP, 387
 - MDEBUG_FINI, 389
 - MDEBUG_FLAG, 386
 - MDEBUG_FLT, 389
 - MDEBUG_FONT, 389
 - MDEBUG_FONTSET, 389
 - MDEBUG_INIT, 389
 - MDEBUG_INPUT, 389
 - MDEBUG_MAX, 389
 - MDEBUG_POP_TIME, 388
 - MDEBUG_PRINT, 386
 - MDEBUG_PRINT0, 386
 - MDEBUG_PRINT1, 386
 - MDEBUG_PRINT2, 386
 - MDEBUG_PRINT3, 387
 - MDEBUG_PRINT4, 387
 - MDEBUG_PRINT5, 387
 - MDEBUG_PRINT_TIME, 388
 - MDEBUG_PUSH_TIME, 387
 - MDebugFlag, 389
 - MEMORY_FULL, 376
 - MERROR, 375
 - MERROR_GOTO, 375
 - MFALP, 376
 - MFATAL, 375
 - minput_fini, 393
 - minput_init, 393
 - mlang_fini, 393
 - mlang_init, 393
 - MLIST_APPEND1, 379
 - MLIST_COPY1, 380
 - MLIST_DELETE1, 380
 - MLIST_FREE1, 381
 - MLIST_INIT1, 379
 - MLIST_INSERT1, 380
 - MLIST_PREPEND1, 379
 - MLIST_RESET, 379
 - mlocale_fini, 393
 - mlocale_init, 393
 - mplist_fini, 391
 - mplist_init, 390
 - MSTRUCT_CALLOC, 378
 - MSTRUCT_CALLOC_SAFE, 378
 - MSTRUCT_MALLOC, 377
 - msymbol_fini, 390
 - msymbol_init, 390
 - MTABLE_ALLOCA, 377
 - MTABLE_CALLOC, 376
 - MTABLE_CALLOC_SAFE, 377
 - MTABLE_MALLOC, 376
 - MTABLE_REALLOC, 377
 - mtext_fini, 391
 - mtext_init, 391
 - mtext_prop_fini, 391
 - mtext_prop_init, 391
 - mtext_allocated, 385
 - MTEXT_COVERAGE_ASCII, 389
 - MTEXT_COVERAGE_FULL, 389
 - MTEXT_COVERAGE_UNICODE, 389

- mtext_nbytes, 385
- mtext_nchars, 385
- MTEXT_READ_ONLY_P, 385
- mtext_reset, 386
- MTextCoverage, 389
- MWARNING, 375
- SAFE_ALLOCA, 378
- SAFE_FREE, 378
- SWAP_16, 388
- SWAP_32, 388
- USE_SAFE_ALLOCA, 378
- internal_info
 - MConverter, 212
- intersect_region
 - MDeviceDriver, 216
- ISALNUM
 - character.h, 323
- ISO_MAX_CHARS
 - charset.h, 328
- ISO_MAX_DIMENSION
 - charset.h, 328
- ISO_MAX_FINAL
 - charset.h, 328
- ISUPPER
 - character.h, 323
- iterate_otf_feature
 - MFontDriver, 258
- key
 - MPlist, 298
 - MTextProperty, 311
- key_head
 - MInputContextInfo, 282
- key_unhandled
 - MInputContextInfo, 283
- keys
 - MInputContextInfo, 282
- langsys
 - MFLTOfSpec, 249
- langsys_tag
 - MFontCapability, 254
- language
 - MFontCapability, 254
 - MInputMethod, 290
 - MInputMethodInfo, 292
- language.c, 394
 - mlanguage_name, 395
- language.h, 396
 - mscript_char_list, 396
 - mscript_from_otf_tag, 396
 - mscript_otf_tag, 396
- last_block
 - MConverter, 210
- layouter
 - MRealizedFace, 301
 - MRealizedFont, 304
- lbearing
 - MDrawGlyph, 226
 - MFLTGlyph, 244
 - MGlyphString, 272
- left_from
 - MDrawGlyphInfo, 229
- left_padding
 - MGlyph, 268
- left_to
 - MDrawGlyphInfo, 229
- len
 - MDatabaseInfo, 213
- length
 - MSymbol, 307
- lenient
 - MConverter, 210
- libotf_positioning_type
 - MGlyph, 269
- line_ascent
 - MGlyphString, 273
- line_break
 - MDrawControl, 222
- line_descent
 - MGlyphString, 273
- line_from
 - MDrawGlyphInfo, 228
- line_to
 - MDrawGlyphInfo, 228
- list
 - MFontDriver, 257
- list_family_names
 - MFontDriver, 257
- locale
 - MInputXIMArgIM, 297
- locale.c, 397
- lock_file
 - MDatabaseInfo, 213
- logical_width
 - MDrawGlyphInfo, 230
- lookup
 - MInputDriver, 287
- M-text, 38
 - Mlanguage, 58
 - mtext, 42
 - mtext_case_compare, 55
 - mtext_casecmp, 55
 - mtext_cat, 45
 - mtext_cat_char, 44
 - mtext_character, 50
 - mtext_chr, 50
 - mtext_cmp, 51
 - mtext_compare, 52
 - mtext_copy, 47
 - mtext_cpy, 45

- mtext_cspn, 53
- mtext_data, 43
- mtext_del, 47
- mtext_dup, 44
- mtext_duplicate, 46
- MTEXT_FORMAT_MAX, 41
- MTEXT_FORMAT_US_ASCII, 41
- MTEXT_FORMAT_UTF_16, 57
- MTEXT_FORMAT_UTF_16BE, 41
- MTEXT_FORMAT_UTF_16LE, 41
- MTEXT_FORMAT_UTF_32, 58
- MTEXT_FORMAT_UTF_32BE, 41
- MTEXT_FORMAT_UTF_32LE, 41
- MTEXT_FORMAT_UTF_8, 41
- mtext_from_data, 42
- mtext_ins, 48
- mtext_ins_char, 49
- mtext_insert, 48
- MTEXT_LBO_ALAS_ID, 42
- MTEXT_LBO_KOREAN_SP, 42
- MTEXT_LBO_MAX, 42
- MTEXT_LBO_SP_CM, 42
- mtext_len, 43
- mtext_line_break, 42
- mtext_lowercase, 56
- mtext_ncasecmp, 55
- mtext_ncat, 45
- mtext_ncmp, 52
- mtext_ncpy, 46
- mtext_pbrk, 53
- mtext_rchr, 51
- mtext_ref_char, 43
- mtext_replace, 49
- mtext_search, 54
- mtext_set_char, 44
- mtext_spn, 53
- mtext_text, 54
- mtext_titlecase, 56
- mtext_tok, 54
- mtext_uppercase, 57
- MTextFormat, 41
- MTextLineBreakOption, 42
- m17n-config.txt, 397
- m17n-core.c, 397
- m17n-core.h, 398
 - M17N_BEGIN_HEADER, 404
 - M17N_END_HEADER, 405
 - Mcase_mapping, 405
 - Minteger, 405
 - Msoft_dotted, 405
- m17n-db.txt, 405
- m17n-flt.c, 405
- m17n-flt.h, 406
 - mflt_font_id, 407
 - mflt_iterate_otf_feature, 407
- m17n-gd.c, 408
- m17n-gui.c, 408
- m17n-gui.h, 408
 - mdebug_dump_font, 414
 - mdebug_dump_fontset, 414
 - MFontset, 414
 - Mfreetype, 415
 - Mxft, 415
- m17n-misc.h, 415
 - mdebug_dump_chartab, 417
 - mdebug_dump_plist, 417
- m17n-X.c, 418
 - device_open, 418
- m17n-X.h, 418
 - minput_xim_driver, 419
- m17n.c, 419
- m17n.h, 419
 - Miso639_2, 426
 - mlanguage_name, 426
- m17n_core_initialized
 - internal.h, 394
- m17n_gui_initialized
 - internal.h, 394
- m17n_shell_initialized
 - internal.h, 394
- M17N_BEGIN_HEADER
 - m17n-core.h, 404
- M17N_CORE_INITIALIZED
 - はじめに, 9
- M17N_END_HEADER
 - m17n-core.h, 405
- M17N_FINI
 - はじめに, 8
- M17N_FUNC
 - コア API, 11
- M17N_GUI_INITIALIZED
 - はじめに, 9
- M17N_INIT
 - はじめに, 8
- m17n_memory_full_handler
 - エラー処理, 192
- M17N_NOT_INITIALIZED
 - はじめに, 9
- M17N_OBJECT
 - internal.h, 381
- m17n_object
 - 管理下オブジェクト, 12
- M17N_OBJECT_ADD_ARRAY
 - internal.h, 383
- M17N_OBJECT_REF
 - internal.h, 381
- m17n_object_ref
 - 管理下オブジェクト, 13
- M17N_OBJECT_REF_NTIMES
 - internal.h, 382
- M17N_OBJECT_REGISTER
 - internal.h, 383

- M17N_OBJECT_UNREF
 - internal.h, [382](#)
- m17n_object_unref
 - 管理下オブジェクト, [13](#)
- M17N_OBJECT_UNREGISTER
 - internal.h, [383](#)
- M17N_SHELL_INITIALIZED
 - はじめに, [9](#)
- m17n_status
 - はじめに, [9](#)
- M17NDIR
 - database.h, [336](#)
- M17NFunc
 - コア API, [11](#)
- M17NLIB_MAJOR_VERSION
 - はじめに, [7](#)
- M17NLIB_MINOR_VERSION
 - はじめに, [7](#)
- M17NLIB_PATCH_LEVEL
 - はじめに, [7](#)
- M17NLIB_VERSION_NAME
 - はじめに, [7](#)
- M17NObject, [197](#)
 - flag, [198](#)
 - freer, [198](#)
 - record, [198](#)
 - ref_count, [197](#)
 - ref_count_extended, [198](#)
 - u, [198](#)
- M17NObjectArray, [198](#)
 - count, [199](#)
 - inc, [199](#)
 - name, [199](#)
 - next, [199](#)
 - objects, [199](#)
 - size, [199](#)
 - used, [199](#)
- M17NObjectHead, [200](#)
 - filler, [200](#)
- M17NObjectRecord, [200](#)
 - counts, [201](#)
 - freer, [200](#)
 - inc, [201](#)
 - size, [201](#)
 - used, [201](#)
- M17NStatus
 - はじめに, [8](#)
- M_CHECK_CHAR
 - internal.h, [376](#)
- M_CHECK_POS
 - internal.h, [383](#)
- M_CHECK_POS_NCHARS
 - internal.h, [384](#)
- M_CHECK_POS_X
 - internal.h, [384](#)
- M_CHECK_RANGE
 - internal.h, [384](#)
- M_CHECK_RANGE_X
 - internal.h, [384](#)
- M_CHECK_READABLE
 - internal-gui.h, [367](#)
- M_CHECK_READONLY
 - internal.h, [385](#)
- M_CHECK_WRITABLE
 - internal-gui.h, [367](#)
- macros
 - MInputMethodInfo, [294](#)
- Madstyle
 - フォント, [157](#)
- mainpage.txt, [426](#)
- MAKE_COMBINING_CODE
 - internal-flt.h, [364](#)
- Maliases
 - 文字セット, [84](#)
- managing_key
 - MSymbol, [306](#)
- map
 - MInputContextInfo, [281](#)
- map_window
 - MDeviceDriver, [217](#)
- Mapple_roman
 - font.h, [356](#)
- maps
 - MInputMethodInfo, [294](#)
- markers
 - MInputContextInfo, [282](#)
- Mascii_compatible
 - 文字セット, [83](#)
- max_advance
 - MRealizedFont, [305](#)
- max_char
 - MCharset, [204](#)
- max_code
 - MCharset, [203](#)
- max_line_ascent
 - MDrawControl, [221](#)
- max_line_descent
 - MDrawControl, [221](#)
- max_line_width
 - MDrawControl, [221](#)
- MAX_UNICODE_CHAR_BYTES
 - character.h, [315](#)
- MAX_UTF8_CHAR_BYTES
 - character.h, [315](#)
- Mbackground
 - フェース, [170](#)
- Mbidi_category
 - 文字, [31](#)
- Mblock
 - 文字, [33](#)
- Mbom
 - コード変換, [104](#)

- Mbox
 - フェイス, 171
- Mcase_mapping
 - m17n-core.h, 405
 - 文字, 33
- Mcased
 - 文字, 32
- Mcategory
 - 文字, 31
- mchar_define_prop
 - character.h, 323
- mchar_fini
 - internal.h, 393
- mchar_init
 - internal.h, 392
- mchar_decode
 - 文字セット, 80
- mchar_define_charset
 - 文字セット, 79
- mchar_define_property
 - 文字, 29
- mchar_encode
 - 文字セット, 80
- mchar_get_prop
 - 文字, 29
- mchar_get_prop_table
 - 文字, 30
- MCHAR_INVALID_CODE
 - 文字セット, 79
- mchar_list_charset
 - 文字セット, 80
- mchar_map_charset
 - 文字セット, 81
- MCHAR_MAX
 - 文字, 28
- mchar_put_prop
 - 文字, 29
- mchar_resolve_charset
 - 文字セット, 80
- Mchar_table
 - 文字テーブル, 38
- MCHARSET
 - charset.h, 326
- MCharset, 201
 - ascii_compatible, 203
 - code_range, 202
 - code_range_mask, 203
 - code_range_min_code, 203
 - decoder, 204
 - dimension, 202
 - encoder, 204
 - final_byte, 204
 - fully_loaded, 205
 - max_char, 204
 - max_code, 203
 - method, 204
 - min_char, 203
 - min_code, 203
 - name, 202
 - no_code_gap, 203
 - nparents, 205
 - parents, 205
 - ref_count, 202
 - revision, 204
 - simple, 205
 - subset_max_code, 205
 - subset_min_code, 205
 - subset_offset, 205
 - unified_max, 204
- Mcharset
 - 文字セット, 86
- mcharset_ascii
 - charset.h, 330
- mcharset_binary
 - charset.h, 330
- mcharset_cache
 - charset.h, 329
- mcharset_decode_char
 - charset.h, 329
- mcharset_encode_char
 - charset.h, 329
- mcharset_find
 - charset.h, 329
- mcharset_fini
 - internal.h, 392
- mcharset_init
 - internal.h, 392
- mcharset_iso_2022_table
 - charset.h, 330
- mcharset_load_from_database
 - charset.h, 329
- mcharset_m17n
 - charset.h, 330
- mcharset_unicode
 - charset.h, 330
- Mcharset_ascii
 - 文字セット, 81
- Mcharset_binary
 - 文字セット, 82
- MCHARSET_ISO_2022
 - charset.h, 328
- Mcharset_iso_8859_1
 - 文字セット, 82
- Mcharset_m17n
 - 文字セット, 82
- mcharset_method
 - charset.h, 328
- MCHARSET_METHOD_DEFERRED
 - charset.h, 329
- MCHARSET_METHOD_MAP
 - charset.h, 329
- MCHARSET_METHOD_MAX

- charset.h, 329
- MCHARSET_METHOD_OFFSET
 - charset.h, 329
- MCHARSET_METHOD_SUBSET
 - charset.h, 329
- MCHARSET_METHOD_SUPERSET
 - charset.h, 329
- Mcharset_unicode
 - 文字セット, 82
- MCharsetISO2022Table, 206
 - charsets, 207
 - classified, 207
 - inc, 206
 - size, 206
 - used, 207
- Mcharsets
 - コード変換, 103
- MCharTable
 - 文字テーブル, 34
- mchartable
 - 文字テーブル, 35
- mchartable_fini
 - internal.h, 391
- mchartable_init
 - internal.h, 391
- mchartable_lookup
 - chartab.h, 332
- mchartable_lookup
 - 文字テーブル, 35
- mchartable_map
 - 文字テーブル, 37
- mchartable_max_char
 - 文字テーブル, 35
- mchartable_min_char
 - 文字テーブル, 35
- mchartable_range
 - 文字テーブル, 37
- mchartable_set
 - 文字テーブル, 36
- mchartable_set_range
 - 文字テーブル, 36
- Mcode_unit
 - コード変換, 104
- Mcodeset
 - ロケール, 114
- Mcoding
 - コード変換, 107
- mcoding_fini
 - internal.h, 392
- mcoding_init
 - internal.h, 392
- mcoding_load_from_database
 - coding.h, 335
- Mcoding_iso_8859_1
 - コード変換, 101
- MCODING_ISO_DESIGNATION_CTEXT
 - コード変換, 92
- MCODING_ISO_DESIGNATION_CTEXT_EXT
 - コード変換, 92
- MCODING_ISO_DESIGNATION_G0
 - コード変換, 92
- MCODING_ISO_DESIGNATION_G1
 - コード変換, 92
- MCODING_ISO_EIGHT_BIT
 - コード変換, 91
- MCODING_ISO_EUC_TW_SHIFT
 - コード変換, 92
- MCODING_ISO_FLAG_MAX
 - コード変換, 92
- MCODING_ISO_FULL_SUPPORT
 - コード変換, 92
- MCODING_ISO_ISO6429
 - コード変換, 92
- MCODING_ISO_LOCKING_SHIFT
 - コード変換, 92
- MCODING_ISO_LONG_FORM
 - コード変換, 91
- MCODING_ISO_RESET_AT_CNTL
 - コード変換, 91
- MCODING_ISO_RESET_AT_EOL
 - コード変換, 91
- MCODING_ISO_REVISION_NUMBER
 - コード変換, 92
- MCODING_ISO_SINGLE_SHIFT
 - コード変換, 92
- MCODING_ISO_SINGLE_SHIFT_7
 - コード変換, 92
- Mcoding_sjis
 - コード変換, 103
- MCODING_TYPE_CHARSET
 - コード変換, 91
- MCODING_TYPE_ISO_2022
 - コード変換, 91
- MCODING_TYPE_MISC
 - コード変換, 91
- MCODING_TYPE_UTF
 - コード変換, 91
- Mcoding_us_ascii
 - コード変換, 101
- Mcoding_utf_16
 - コード変換, 102
- Mcoding_utf_16be
 - コード変換, 102
- Mcoding_utf_16le
 - コード変換, 102
- Mcoding_utf_32
 - コード変換, 102
- Mcoding_utf_32be
 - コード変換, 103
- Mcoding_utf_32le
 - コード変換, 103
- Mcoding_utf_8

- コード変換, 101
- Mcoding_utf_8_full
 - コード変換, 102
- MCodingFlagISO2022
 - コード変換, 91
- MCodingInfoISO2022, 207
 - designations, 208
 - flags, 208
 - initial_invocation, 207
- MCodingInfoUTF, 208
 - bom, 209
 - code_unit_bits, 208
 - endian, 209
- MCodingType
 - コード変換, 91
- Mcolormap
 - フレーム, 145
- Mcombining
 - internal-flt.h, 365
- Mcombining_class
 - 文字, 31
- Mcomplicated_case_folding
 - 文字, 32
- Mconfigured
 - 入力メソッド (基本部分), 134
- mconv_register_charset_coding
 - coding.h, 335
- mconv_buffer_converter
 - コード変換, 93
- mconv_decode
 - コード変換, 95
- mconv_decode_buffer
 - コード変換, 96
- mconv_decode_stream
 - コード変換, 96
- mconv_define_coding
 - コード変換, 92
- mconv_encode
 - コード変換, 97
- mconv_encode_buffer
 - コード変換, 98
- mconv_encode_range
 - コード変換, 97
- mconv_encode_stream
 - コード変換, 98
- mconv_free_converter
 - コード変換, 94
- mconv_getc
 - コード変換, 99
- mconv_gets
 - コード変換, 100
- mconv_list_codings
 - コード変換, 92
- mconv_putc
 - コード変換, 100
- mconv_rebind_buffer
 - コード変換, 94
- mconv_rebind_stream
 - コード変換, 95
- mconv_reset_converter
 - コード変換, 94
- mconv_resolve_coding
 - コード変換, 92
- mconv_stream_converter
 - コード変換, 93
- mconv_ungetc
 - コード変換, 99
- MCONVERSION_RESULT_INSUFFICIENT_DST
 - コード変換, 90
- MCONVERSION_RESULT_INSUFFICIENT_SRC
 - コード変換, 90
- MCONVERSION_RESULT_INVALID_BYTE
 - コード変換, 90
- MCONVERSION_RESULT_INVALID_CHAR
 - コード変換, 90
- MCONVERSION_RESULT_IO_ERROR
 - コード変換, 90
- MCONVERSION_RESULT_SUCCESS
 - コード変換, 90
- MConversionResult
 - コード変換, 90
- MConverter, 209
 - at_most, 210
 - c, 211
 - dbl, 211
 - internal_info, 212
 - last_block, 210
 - lenient, 210
 - nbytes, 211
 - nchars, 211
 - ptr, 211
 - result, 211
 - status, 211
- Mcustomized
 - 入力メソッド (基本部分), 134
- MDatabase
 - データベース, 73
- mdatabase_check
 - database.h, 338
- mdatabase_dir_list
 - database.h, 339
- mdatabase_file
 - database.h, 338
- mdatabase_find_file
 - database.h, 338
- mdatabase_fini
 - internal.h, 392
- mdatabase_init
 - internal.h, 392
- mdatabase_load_charset_func
 - database.h, 339
- mdatabase_load_for_keys

- database.h, [337](#)
- mdatabase_lock
 - database.h, [338](#)
- mdatabase_props
 - database.h, [339](#)
- mdatabase_save
 - database.h, [338](#)
- mdatabase_unlock
 - database.h, [338](#)
- mdatabase_update
 - database.h, [337](#)
- mdatabase_define
 - データベース, [74](#)
- mdatabase_dir
 - データベース, [76](#)
- mdatabase_find
 - データベース, [74](#)
- mdatabase_list
 - データベース, [74](#)
- mdatabase_load
 - データベース, [75](#)
- mdatabase_tag
 - データベース, [75](#)
- MDatabaseInfo, [212](#)
 - absolute_filename, [213](#)
 - filename, [213](#)
 - len, [213](#)
 - lock_file, [213](#)
 - properties, [214](#)
 - status, [213](#)
 - time, [213](#)
 - uniq_file, [213](#)
- MDatabaseStatus
 - database.h, [337](#)
- mdb
 - MInputMethodInfo, [292](#)
- MDB_STATUS_AUTO
 - database.h, [337](#)
- MDB_STATUS_AUTO_WILDCARD
 - database.h, [337](#)
- MDB_STATUS_DISABLED
 - database.h, [337](#)
- MDB_STATUS_EXPLICIT
 - database.h, [337](#)
- MDB_STATUS_OUTDATED
 - database.h, [337](#)
- MDB_STATUS_UPDATED
 - database.h, [337](#)
- mdebug_add_object_array
 - internal.h, [389](#)
- mdebug_flags
 - internal.h, [394](#)
- mdebug_output
 - internal.h, [394](#)
- mdebug_pop_time
 - internal.h, [390](#)
- mdebug_print_time
 - internal.h, [390](#)
- mdebug_push_time
 - internal.h, [390](#)
- mdebug_register_object
 - internal.h, [389](#)
- mdebug_unregister_object
 - internal.h, [390](#)
- MDEBUG_ALL
 - internal.h, [389](#)
- MDEBUG_CHARSET
 - internal.h, [389](#)
- MDEBUG_CODING
 - internal.h, [389](#)
- MDEBUG_DATABASE
 - internal.h, [389](#)
- MDEBUG_DUMP
 - internal.h, [387](#)
- mdebug_dump_all_symbols
 - デバッグサポート, [195](#)
- mdebug_dump_chartab
 - chartab.c, [331](#)
 - m17n-misc.h, [417](#)
- mdebug_dump_face
 - デバッグサポート, [194](#)
- mdebug_dump_ftl
 - FLT API, [138](#)
- mdebug_dump_font
 - font.c, [347](#)
 - m17n-gui.h, [414](#)
- mdebug_dump_fontset
 - fontset.c, [357](#)
 - m17n-gui.h, [414](#)
- mdebug_dump_im
 - デバッグサポート, [194](#)
- mdebug_dump_mtext
 - デバッグサポート, [194](#)
- mdebug_dump_plist
 - m17n-misc.h, [417](#)
 - plist.c, [434](#)
- mdebug_dump_symbol
 - デバッグサポート, [195](#)
- MDEBUG_FINI
 - internal.h, [389](#)
- MDEBUG_FLAG
 - internal.h, [386](#)
- MDEBUG_FLT
 - internal.h, [389](#)
- MDEBUG_FONT
 - internal.h, [389](#)
- MDEBUG_FONTSET
 - internal.h, [389](#)
- mdebug_hook
 - デバッグサポート, [194](#)
- MDEBUG_INIT
 - internal.h, [389](#)

- MDEBUG.INPUT
 - internal.h, [389](#)
- MDEBUG.MAX
 - internal.h, [389](#)
- MDEBUG_POP_TIME
 - internal.h, [388](#)
- MDEBUG_PRINT
 - internal.h, [386](#)
- MDEBUG_PRINT0
 - internal.h, [386](#)
- MDEBUG_PRINT1
 - internal.h, [386](#)
- MDEBUG_PRINT2
 - internal.h, [386](#)
- MDEBUG_PRINT3
 - internal.h, [387](#)
- MDEBUG_PRINT4
 - internal.h, [387](#)
- MDEBUG_PRINT5
 - internal.h, [387](#)
- MDEBUG_PRINT_TIME
 - internal.h, [388](#)
- MDEBUG_PUSH_TIME
 - internal.h, [387](#)
- MDebugFlag
 - internal.h, [389](#)
- Mdefine_coding
 - 文字セット, [84](#)
- Mdepth
 - フレーム, [145](#)
- Mdesignation
 - コード変換, [104](#)
- Mdesignation_cxtext
 - コード変換, [105](#)
- Mdesignation_cxtext_ext
 - コード変換, [106](#)
- Mdesignation_g0
 - コード変換, [105](#)
- Mdesignation_g1
 - コード変換, [105](#)
- Mdevice
 - フレーム, [144](#)
- MDEVICE_SUPPORT_INPUT
 - internal-gui.h, [369](#)
- MDEVICE_SUPPORT_OUTPUT
 - internal-gui.h, [369](#)
- MDeviceDriver, [214](#)
 - adjust_window, [218](#)
 - close, [215](#)
 - create_window, [217](#)
 - destroy_window, [217](#)
 - draw_box, [216](#)
 - draw_empty_boxes, [216](#)
 - draw_hline, [216](#)
 - draw_points, [216](#)
 - dump_region, [217](#)
 - fill_space, [215](#)
 - free_realized_face, [215](#)
 - free_region, [217](#)
 - get_prop, [215](#)
 - intersect_region, [216](#)
 - map_window, [217](#)
 - parse_event, [218](#)
 - realize_face, [215](#)
 - region_add_rect, [217](#)
 - region_from_rect, [216](#)
 - region_to_rect, [217](#)
 - union_rect_with_region, [216](#)
 - unmap_window, [218](#)
 - window_geometry, [218](#)
- MDeviceType
 - internal-gui.h, [369](#)
- Mdimension
 - 文字セット, [83](#)
- Mdisplay
 - フレーム, [144](#)
- mdraw_fini
 - internal-gui.h, [370](#)
- mdraw_init
 - internal-gui.h, [370](#)
- mdraw_clear_cache
 - 表示, [186](#)
- mdraw_coordinates_position
 - 表示, [183](#)
- mdraw_default_line_break
 - 表示, [185](#)
- mdraw_glyph_info
 - 表示, [184](#)
- mdraw_glyph_list
 - 表示, [184](#)
- mdraw_image_text
 - 表示, [181](#)
- mdraw_line_break_option
 - 表示, [186](#)
- mdraw_per_char_extents
 - 表示, [185](#)
- mdraw_text
 - 表示, [179](#)
- mdraw_text_extents
 - 表示, [182](#)
- mdraw_text_items
 - 表示, [185](#)
- mdraw_text_per_char_extents
 - 表示, [182](#)
- mdraw_text_with_control
 - 表示, [181](#)
- Mdrawable
 - フレーム, [145](#)
- MDrawControl, [218](#)
 - align_head, [219](#)
 - anti_alias, [220](#)
 - as_image, [219](#)

- clip_region, 223
 - cursor_bidi, 222
 - cursor_pos, 222
 - cursor_width, 222
 - disable_caching, 223
 - disable_overlapping_adjustment, 220
 - enable_bidi, 220
 - fixed_width, 220
 - format, 221
 - ignore_formatting_char, 220
 - line_break, 222
 - max_line_ascent, 221
 - max_line_descent, 221
 - max_line_width, 221
 - min_line_ascent, 221
 - min_line_descent, 221
 - orientation_reversed, 220
 - partial_update, 223
 - tab_width, 221
 - two_dimensional, 220
 - with_cursor, 222
- MDrawGlyph, 223
- ascent, 226
 - descent, 226
 - font, 226
 - font_type, 226
 - fontp, 226
 - from, 225
 - glyph_code, 225
 - lbearing, 226
 - rbearing, 226
 - to, 225
 - x_advance, 225
 - x_off, 225
 - y_advance, 225
 - y_off, 225
- MDrawGlyphInfo, 227
- font, 229
 - from, 228
 - left_from, 229
 - left_to, 229
 - line_from, 228
 - line_to, 228
 - logical_width, 230
 - metrics, 229
 - next_to, 229
 - prev_from, 229
 - right_from, 229
 - right_to, 230
 - to, 228
 - x, 228
 - y, 228
- MDrawMetric, 230
- height, 231
 - width, 231
 - x, 230
 - y, 231
- MDrawPoint, 231
- x, 231
 - y, 231
- MDrawRegion
- 表示, 179
- MDrawTextItem, 232
- control, 233
 - delta, 233
 - face, 233
 - mt, 233
- MDrawWindow
- 表示, 179
- measured
- MFLTGlyph, 245
- Meight_bit
- コード変換, 105
- MEMORY_FULL
- internal.h, 376
- MERROR
- internal.h, 375
- MERROR_CHAR
- エラー処理, 191
- MERROR_CHARSET
- エラー処理, 191
- MERROR_CHARTABLE
- エラー処理, 191
- merror_code
- エラー処理, 192
- MERROR_CODING
- エラー処理, 191
- MERROR_DB
- エラー処理, 191
- MERROR_DEBUG
- エラー処理, 191
- MERROR_DRAW
- エラー処理, 191
- MERROR_FACE
- エラー処理, 191
- MERROR_FLT
- エラー処理, 191
- MERROR_FONT
- エラー処理, 191
- MERROR_FONT_FT
- エラー処理, 191
- MERROR_FONT_OTF
- エラー処理, 191
- MERROR_FONT_X
- エラー処理, 191
- MERROR_FONTSET
- エラー処理, 191
- MERROR_FRAME
- エラー処理, 191
- MERROR_GD
- エラー処理, 191
- MERROR_GOTO

- internal.h, 375
- MERROR_IM
 - エラー処理, 191
- MERROR_IO
 - エラー処理, 191
- MERROR_LANGUAGE
 - エラー処理, 191
- MERROR_LOCALE
 - エラー処理, 191
- MERROR_MAX
 - エラー処理, 191
- MERROR_MEMORY
 - エラー処理, 191
- MERROR_MISC
 - エラー処理, 191
- MERROR_MTEXT
 - エラー処理, 191
- MERROR_NONE
 - エラー処理, 191
- MERROR_OBJECT
 - エラー処理, 191
- MERROR_PLIST
 - エラー処理, 191
- MERROR_RANGE
 - エラー処理, 191
- MERROR_SYMBOL
 - エラー処理, 191
- MERROR_TEXTPROP
 - エラー処理, 191
- MERROR_WIN
 - エラー処理, 191
- MERROR_X
 - エラー処理, 191
- MErrorCode
 - エラー処理, 191
- method
 - MCharset, 204
- metrics
 - MDrawGlyphInfo, 229
- Meuc.tw.shift
 - コード変換, 106
- MFace, 233
 - control, 234
 - frame_list, 235
 - hook, 235
 - property, 234
- Mface
 - フェース, 177
- mface
 - フェース, 167
- mface_default
 - face.h, 345
- mface_fini
 - internal-gui.h, 370
- mface_for_chars
 - face.h, 344
- mface_free_realized
 - face.h, 344
- mface_init
 - internal-gui.h, 370
- mface_realize
 - face.h, 344
- mface_update_frame_face
 - face.h, 344
- MFACE_ADSTYLE
 - face.h, 343
- MFACE_BACKGROUND
 - face.h, 344
- mface_black
 - フェース, 175
- mface_blue
 - フェース, 176
- mface_bold
 - フェース, 173
- mface_bold_italic
 - フェース, 174
- MFACE_BOX
 - face.h, 344
- mface_copy
 - フェース, 167
- mface_cyan
 - フェース, 176
- mface_equal
 - フェース, 167
- MFACE_FAMILY
 - face.h, 343
- MFACE_FONTSET
 - face.h, 344
- MFACE_FOREGROUND
 - face.h, 344
- MFACE_FOUNDRY
 - face.h, 343
- mface_from_font
 - フェース, 168
- mface_get_hook
 - フェース, 168
- mface_get_prop
 - フェース, 168
- mface_green
 - フェース, 176
- MFACE_HLINE
 - face.h, 344
- MFACE_HLINE_BOTTOM
 - MFaceHLineProp, 238
- MFACE_HLINE_OVER
 - MFaceHLineProp, 238
- MFACE_HLINE_STRIKE_THROUGH
 - MFaceHLineProp, 238
- MFACE_HLINE_TOP
 - MFaceHLineProp, 238
- MFACE_HLINE_UNDER
 - MFaceHLineProp, 238

- MFACE_HOOK_ARG
 - face.h, [344](#)
- mface_italic
 - フェース, [173](#)
- mface_large
 - フェース, [175](#)
- mface_magenta
 - フェース, [177](#)
- mface_medium
 - フェース, [173](#)
- mface_merge
 - フェース, [167](#)
- mface_normal_video
 - フェース, [172](#)
- mface_normalsize
 - フェース, [174](#)
- MFACE_PROPERTY_MAX
 - face.h, [344](#)
- mface_put_hook
 - フェース, [169](#)
- mface_put_prop
 - フェース, [169](#)
- MFACE_RATIO
 - face.h, [344](#)
- mface_red
 - フェース, [176](#)
- mface_reverse_video
 - フェース, [172](#)
- MFACE_SIZE
 - face.h, [344](#)
- mface_small
 - フェース, [174](#)
- MFACE_STRETCH
 - face.h, [343](#)
- MFACE_STYLE
 - face.h, [343](#)
- mface_underline
 - フェース, [173](#)
- mface_update
 - フェース, [169](#)
- MFACE_VIDEOMODE
 - face.h, [344](#)
- MFACE_WEIGHT
 - face.h, [343](#)
- mface_white
 - フェース, [175](#)
- mface_x_large
 - フェース, [175](#)
- mface_x_small
 - フェース, [174](#)
- mface_xx_large
 - フェース, [175](#)
- mface_xx_small
 - フェース, [174](#)
- mface_yellow
 - フェース, [176](#)
- MFaceBoxProp, [235](#)
 - color_bottom, [236](#)
 - color_left, [236](#)
 - color_right, [236](#)
 - color_top, [236](#)
 - inner_hmargin, [236](#)
 - inner_vmargin, [237](#)
 - outer_hmargin, [237](#)
 - outer_vmargin, [237](#)
 - width, [236](#)
- MFaceHLineProp, [237](#)
 - color, [239](#)
 - MFACE_HLINE_BOTTOM, [238](#)
 - MFACE_HLINE_OVER, [238](#)
 - MFACE_HLINE_STRIKE_THROUGH, [238](#)
 - MFACE_HLINE_TOP, [238](#)
 - MFACE_HLINE_UNDER, [238](#)
 - MFaceHLineType, [238](#)
 - type, [238](#)
 - width, [239](#)
- MFaceHLineType
 - MFaceHLineProp, [238](#)
- MFaceHookFunc
 - フェース, [166](#)
- MFaceProperty
 - face.h, [343](#)
- MFAILP
 - internal.h, [376](#)
- Mfamily
 - フォント, [156](#)
- MFATAL
 - internal.h, [375](#)
- Mfinal_byte
 - 文字セット, [83](#)
- Mflags
 - コード変換, [103](#)
- MFLT
 - FLT API, [137](#)
- mflt_coverage
 - FLT API, [138](#)
- mflt_dump_gstring
 - FLT API, [138](#)
- mflt_enable_new_feature
 - FLT API, [139](#)
- mflt_find
 - FLT API, [137](#)
- mflt_font_id
 - FLT API, [139](#)
 - m17n-flt.h, [407](#)
- mflt_get
 - FLT API, [137](#)
- mflt_iterate_of_feature
 - FLT API, [139](#)
 - m17n-flt.h, [407](#)
- mflt_name
 - FLT API, [137](#)

- mflt_run
 - FLT API, 138
- mflt_try_otf
 - FLT API, 139
- MFLTFont, 239
 - check_otf, 241
 - drive_otf, 241
 - family, 240
 - get_glyph_id, 240
 - get_metrics, 240
 - internal, 241
 - x_ppem, 240
 - y_ppem, 240
- MFLTFontForRealized, 241
 - font, 242
 - rfont, 242
- MFLTGlyph, 242
 - adjusted, 245
 - ascent, 244
 - c, 243
 - code, 243
 - descent, 244
 - encoded, 244
 - from, 243
 - internal, 245
 - lbearing, 244
 - measured, 245
 - rbearing, 244
 - to, 243
 - xadv, 243
 - xoff, 244
 - yadv, 243
 - yoff, 244
- MFLTGlyphAdjustment, 245
 - advance_is_absolute, 246
 - back, 246
 - set, 246
 - xadv, 246
 - xoff, 246
 - yadv, 246
 - yoff, 246
- MFLTGlyphString, 247
 - allocated, 248
 - glyph_size, 247
 - glyphs, 248
 - r2l, 248
 - used, 248
- MFLTOfSpec, 248
 - features, 249
 - langsys, 249
 - script, 249
 - sym, 249
- MFont, 250
 - capability, 252
 - encoding, 252
 - file, 252
 - for_full_width, 251
 - multiple_sizes, 252
 - property, 251
 - size, 252
 - source, 251
 - spacing, 251
 - type, 251
- Mfont
 - フレーム, 145
- mfont
 - フォント, 150
- mfont_check_capability
 - font.h, 355
- mfont_encode_char
 - font.h, 353
- mfont_encoding_list
 - font.h, 355
- mfont_fini
 - internal-gui.h, 370
- mfont_flt_encode_char
 - font.h, 355
- mfont_flt_fini
 - font.h, 351
- mfont_flt_init
 - font.h, 351
- mfont_flt_run
 - font.h, 355
- mfont_fontset_fini
 - internal-gui.h, 371
- mfont_fontset_init
 - internal-gui.h, 371
- mfont_free_realized
 - font.h, 352
- mfont_free_realized_fontset
 - fontset.h, 358
- mfont_get_capability
 - font.h, 355
- mfont_get_glyph_id
 - font.h, 353
- mfont_get_metric
 - font.h, 354
- mfont_get_metrics
 - font.h, 354
- mfont_has_char
 - font.h, 352
- mfont_init
 - internal-gui.h, 370
- mfont_list
 - font.h, 353
- mfont_lookup_fontset
 - fontset.h, 358
- mfont_match_p
 - font.h, 352
- mfont_merge
 - font.h, 352
- mfont_open

- font.h, 353
- mfont_parse_name_into_font
 - font.h, 354
- mfont_property_table
 - font.h, 356
- mfont_realize_fontset
 - fontset.h, 358
- mfont_select
 - font.h, 353
- mfont_set_property
 - font.h, 354
- mfont_set_spec_from_face
 - font.h, 352
- mfont_set_spec_from_plist
 - font.h, 352
- mfont_split_name
 - font.h, 354
- MFONT_ADSTYLE
 - font.h, 350
- Mfont_ascent
 - フレーム, 146
- mfont_check
 - フォント, 155
- mfont_close
 - フォント, 156
- mfont_copy
 - フォント, 151
- Mfont_descent
 - フレーム, 146
- mfont_encapsulate
 - フォント, 156
- MFONT_FAMILY
 - font.h, 350
- mfont_find
 - フォント, 153
- MFONT_FOUNDRY
 - font.h, 350
- mfont_freetype_path
 - フォント, 159
- mfont_from_name
 - フォント, 154
- mfont_get_prop
 - フォント, 152
- MFONT_INIT
 - font.h, 349
- mfont_list
 - フォント, 154
- mfont_list_family_names
 - フォント, 155
- mfont_match_p
 - フォント, 155
- mfont_name
 - フォント, 154
- mfont_open
 - フォント, 155
- MFONT_OTT_GPOS
 - font.h, 351
- MFONT_OTT_GSUB
 - font.h, 351
- MFONT_OTT_MAX
 - font.h, 351
- mfont_parse_name
 - フォント, 151
- MFONT_PROPERTY_MAX
 - font.h, 350
- mfont_put_prop
 - フォント, 152
- MFONT_REGISTRY
 - font.h, 350
- mfont_resize_ratio
 - フォント, 154
- MFONT_RESY
 - font.h, 350
- mfont_selection_priority
 - フォント, 152
- mfont_set_encoding
 - フォント, 153
- mfont_set_selection_priority
 - フォント, 153
- MFONT_SIZE
 - font.h, 350
- MFONT_SOURCE_FT
 - font.h, 351
- MFONT_SOURCE_UNDECIDED
 - font.h, 351
- MFONT_SOURCE_X
 - font.h, 351
- MFONT_SPACING
 - font.h, 350
- MFONT_SPACING_CHARGCELL
 - font.h, 351
- MFONT_SPACING_MONO
 - font.h, 351
- MFONT_SPACING_PROPORTIONAL
 - font.h, 351
- MFONT_SPACING_UNDECIDED
 - font.h, 351
- MFONT_STRETCH
 - font.h, 350
- MFONT_STYLE
 - font.h, 350
- MFONT_TYPE_FAILURE
 - font.h, 350
- MFONT_TYPE_OBJECT
 - font.h, 350
- MFONT_TYPE_REALIZED
 - font.h, 350
- MFONT_TYPE_SPEC
 - font.h, 350
- mfont_unparse_name
 - フォント, 151
- MFONT_WEIGHT

- font.h, 350
- Mfont_width
 - フレーム, 145
- MFontCapability, 253
 - control, 254
 - features, 255
 - langsys_tag, 254
 - language, 254
 - nfeatures, 255
 - otf, 254
 - script, 254
 - script_tag, 254
 - str, 254
 - tags, 255
- Mfontconfig
 - フォント, 159
- MFontDriver, 255
 - check_capability, 257
 - check_otf, 258
 - close, 257
 - drive_otf, 258
 - encapsulate, 257
 - encode_char, 257
 - find_metric, 256
 - has_char, 256
 - iterate_otf_feature, 258
 - list, 257
 - list_family_names, 257
 - open, 256
 - render, 257
 - select, 256
 - try_otf, 258
- MFontEncoding
 - font.h, 349
- Mfontfile
 - フォント, 158
- MFontList, 259
 - fonts, 259
 - nfonts, 260
 - object, 259
- MFontOpenTypeTable
 - font.h, 351
- MFontProperty
 - font.h, 350
- MFontPropertyTable, 260
 - inc, 261
 - names, 261
 - property, 261
 - size, 260
 - used, 261
- MFontScore, 262
 - font, 262
 - score, 262
- MFontset
 - m17n-gui.h, 414
- Mfontset
 - フェース, 171
- mfontset
 - フォントセット, 161
- mfontset_get_font
 - fontset.h, 358
- mfontset_copy
 - フォントセット, 161
- mfontset_lookup
 - フォントセット, 162
- mfontset_modify_entry
 - フォントセット, 161
- mfontset_name
 - フォントセット, 161
- MFontSource
 - font.h, 350
- MFontSpacing
 - font.h, 351
- MFontType
 - font.h, 350
- Mforeground
 - フェース, 170
- Mfoundry
 - フォント, 156
- MFrame, 263
 - ascent, 265
 - average_width, 265
 - background, 264
 - control, 264
 - descent, 265
 - device, 265
 - device_type, 266
 - dpi, 266
 - driver, 266
 - face, 264
 - font, 264
 - font_driver_list, 266
 - foreground, 264
 - realized_face_list, 266
 - realized_font_list, 266
 - realized_fontset_list, 266
 - rface, 265
 - space_width, 265
 - tick, 265
 - videomode, 264
- mframe
 - フレーム, 142
- mframe_default
 - フレーム, 146
- mframe_get_prop
 - フレーム, 143
- Mfreetype
 - m17n-gui.h, 415
 - フォント, 159
- Mfull_support
 - コード変換, 107
- Mgd

- フレーム, 145
- MGLYPH
 - internal-gui.h, 367
- MGlyph, 267
 - bidi_level, 268
 - category, 268
 - enabled, 268
 - g, 267
 - left_padding, 268
 - libotf_positioning_type, 269
 - rface, 268
 - right_padding, 268
 - type, 268
- MGlyphString, 269
 - anti_alias, 273
 - ascent, 271
 - control, 273
 - descent, 272
 - frame, 270
 - from, 271
 - glyphs, 271
 - head, 270
 - height, 271
 - inc, 270
 - indent, 273
 - lbearing, 272
 - line_ascent, 273
 - line_descent, 273
 - next, 273
 - physical_ascent, 272
 - physical_descent, 272
 - rbearing, 272
 - size, 270
 - text_ascent, 272
 - text_descent, 272
 - tick, 270
 - to, 271
 - top, 274
 - used, 271
 - width, 271
 - width_limit, 273
- Mhline
 - フェース, 171
- Mhook_arg
 - フェース, 172
- Mhook_func
 - フェース, 171
- MIMInputStack
 - input.h, 363
- MIMMap
 - input.h, 363
- MIMState
 - input.h, 362
- min_char
 - MCharset, 203
- min_code
 - MCharset, 203
- min_line_ascent
 - MDrawControl, 221
- min_line_descent
 - MDrawControl, 221
- Minherited
 - 入力メソッド (基本部分), 134
- minput_char_to_key
 - input.h, 363
- minput_fini
 - internal.h, 393
- minput_init
 - internal.h, 393
- minput_win_fini
 - internal-gui.h, 371
- minput_win_init
 - internal-gui.h, 371
- minput_assign_command_keys
 - 入力メソッド (基本部分), 130
- minput_callback
 - 入力メソッド (基本部分), 131
- MINPUT_CANDIDATES_CHANGED_MAX
 - 入力メソッド (基本部分), 118
- Minput_candidates_done
 - 入力メソッド (基本部分), 132
- Minput_candidates_draw
 - 入力メソッド (基本部分), 133
- MINPUT_CANDIDATES_INDEX_CHANGED
 - 入力メソッド (基本部分), 118
- MINPUT_CANDIDATES_LIST_CHANGED
 - 入力メソッド (基本部分), 118
- MINPUT_CANDIDATES_SHOW_CHANGED
 - 入力メソッド (基本部分), 118
- Minput_candidates_start
 - 入力メソッド (基本部分), 132
- minput_close_jm
 - 入力メソッド (基本部分), 119
- minput_config_command
 - 入力メソッド (基本部分), 124
- minput_config_file
 - 入力メソッド (基本部分), 127
- minput_config_variable
 - 入力メソッド (基本部分), 126
- minput_create_jc
 - 入力メソッド (基本部分), 119
- minput_default_driver
 - 入力メソッド (基本部分), 134
- Minput_delete_surrounding_text
 - 入力メソッド (基本部分), 133
- minput_destroy_jc
 - 入力メソッド (基本部分), 119
- Minput_driver
 - 入力メソッド (基本部分), 135
- minput_driver
 - 入力メソッド (基本部分), 135
- minput_event_to_key

- 入力メソッド (GUI), 188
- `minput.filter`
 - 入力メソッド (基本部分), 120
- `Minput.focus.in`
 - 入力メソッド (基本部分), 134
- `Minput.focus.move`
 - 入力メソッド (基本部分), 134
- `Minput.focus.out`
 - 入力メソッド (基本部分), 133
- `minput.get.command`
 - 入力メソッド (基本部分), 122
- `minput.get.commands`
 - 入力メソッド (基本部分), 130
- `minput.get.description`
 - 入力メソッド (基本部分), 122
- `Minput.get.surrounding.text`
 - 入力メソッド (基本部分), 133
- `minput.get.title.icon`
 - 入力メソッド (基本部分), 121
- `minput.get.variable`
 - 入力メソッド (基本部分), 125
- `minput.get.variables`
 - 入力メソッド (基本部分), 128
- `minput.gui.driver`
 - 入力メソッド (GUI), 188
- `MINPUT_KEY_ALT_MODIFIER`
 - `input.h`, 362
- `MINPUT_KEY_ALTGR_MODIFIER`
 - `input.h`, 362
- `MINPUT_KEY_CONTROL_MODIFIER`
 - `input.h`, 362
- `MINPUT_KEY_HYPER_MODIFIER`
 - `input.h`, 362
- `MINPUT_KEY_META_MODIFIER`
 - `input.h`, 362
- `MINPUT_KEY_SHIFT_MODIFIER`
 - `input.h`, 361
- `MINPUT_KEY_SUPER_MODIFIER`
 - `input.h`, 362
- `minput.list`
 - 入力メソッド (基本部分), 128
- `minput.lookup`
 - 入力メソッド (基本部分), 120
- `Minput.method`
 - 入力メソッド (基本部分), 131
- `minput.open.im`
 - 入力メソッド (基本部分), 119
- `minput.parse.im.names`
 - 入力メソッド (基本部分), 131
- `Minput.predit.done`
 - 入力メソッド (基本部分), 132
- `Minput.predit.draw`
 - 入力メソッド (基本部分), 132
- `Minput.predit.start`
 - 入力メソッド (基本部分), 131
- `Minput.reset`
 - 入力メソッド (基本部分), 133
- `minput.reset.ic`
 - 入力メソッド (基本部分), 121
- `minput.save.config`
 - 入力メソッド (基本部分), 127
- `Minput.set.spot`
 - 入力メソッド (基本部分), 133
- `minput.set.spot`
 - 入力メソッド (基本部分), 120
- `minput.set.variable`
 - 入力メソッド (基本部分), 129
- `Minput.status.done`
 - 入力メソッド (基本部分), 132
- `Minput.status.draw`
 - 入力メソッド (基本部分), 132
- `Minput.status.start`
 - 入力メソッド (基本部分), 132
- `Minput.toggle`
 - 入力メソッド (基本部分), 133
- `minput.toggle`
 - 入力メソッド (基本部分), 121
- `minput.xim.driver`
 - `m17n-X.h`, 419
- `MInputCallbackFunc`
 - 入力メソッド (基本部分), 118
- `MInputCandidatesChanged`
 - 入力メソッド (基本部分), 118
- `MInputContext`, 274
 - `active`, 276
 - `arg`, 276
 - `ascent`, 276
 - `candidate.from`, 278
 - `candidate.index`, 278
 - `candidate.list`, 278
 - `candidate.show`, 279
 - `candidate.to`, 278
 - `candidates.changed`, 279
 - `cursor.pos`, 278
 - `cursor.pos.changed`, 278
 - `descent`, 276
 - `fontsize`, 276
 - `im`, 275
 - `info`, 277
 - `mt`, 277
 - `plist`, 279
 - `pos`, 277
 - `preedit`, 277
 - `preedit.changed`, 278
 - `produced`, 275
 - `spot`, 277
 - `status`, 277
 - `status.changed`, 277
 - `x`, 276
 - `y`, 276
- `MInputContextInfo`, 280
 - `commit_key_head`, 282

- fallbacks, [284](#)
- following_text, [283](#)
- inc, [281](#)
- key_head, [282](#)
- key_unhandled, [283](#)
- keys, [282](#)
- map, [281](#)
- markers, [282](#)
- preceding_text, [283](#)
- preedit_saved, [282](#)
- prev_state, [281](#)
- pushing_or_switching, [284](#)
- size, [281](#)
- stack, [284](#)
- state, [281](#)
- state_hook, [283](#)
- state_key_head, [282](#)
- state_pos, [282](#)
- tick, [284](#)
- used, [281](#)
- vars, [283](#)
- vars_saved, [283](#)
- win_info, [283](#)
- MInputDriver, [284](#)
 - callback_list, [287](#)
 - close_im, [286](#)
 - create_ic, [286](#)
 - destroy_ic, [286](#)
 - filter, [286](#)
 - lookup, [287](#)
 - open_im, [286](#)
- MInputGUIArgIC, [288](#)
 - client, [288](#)
 - focus, [289](#)
 - frame, [288](#)
- MInputMethod, [289](#)
 - arg, [290](#)
 - driver, [290](#)
 - info, [290](#)
 - language, [290](#)
 - name, [290](#)
- MInputMethodInfo, [291](#)
 - bc_cmds, [293](#)
 - bc_vars, [293](#)
 - cmds, [292](#)
 - configured_cmds, [293](#)
 - configured_vars, [293](#)
 - description, [293](#)
 - externals, [294](#)
 - extra, [292](#)
 - language, [292](#)
 - macros, [294](#)
 - maps, [294](#)
 - mdb, [292](#)
 - name, [292](#)
 - states, [294](#)
 - tick, [294](#)
 - title, [293](#)
 - vars, [293](#)
- MInputXIMArgIC, [294](#)
 - client_win, [295](#)
 - focus_win, [295](#)
 - input_style, [295](#)
 - preedit_attrs, [295](#)
 - status_attrs, [295](#)
- MInputXIMArgIM, [296](#)
 - db, [296](#)
 - display, [296](#)
 - locale, [297](#)
 - modifier_list, [297](#)
 - res_class, [296](#)
 - res_name, [296](#)
- Minteger
 - m17n-core.h, [405](#)
 - プロパティリスト, [26](#)
- Minvocation
 - コード変換, [104](#)
- MISC API, [189](#)
- Miso10646_1
 - font.h, [356](#)
- Miso639_1
 - ロケール, [113](#)
- Miso639_2
 - m17n.h, [426](#)
 - ロケール, [113](#)
- Miso8859_1
 - font.h, [356](#)
- Miso_2022
 - コード変換, [104](#)
- Miso_6429
 - コード変換, [106](#)
- mlang_fini
 - internal.h, [393](#)
- mlang_init
 - internal.h, [393](#)
- Mlanguage
 - M-text, [58](#)
- mlanguage_code
 - ロケール, [109](#)
- mlanguage_list
 - ロケール, [109](#)
- mlanguage_name
 - language.c, [395](#)
 - m17n.h, [426](#)
- mlanguage_name_list
 - ロケール, [109](#)
- mlanguage_text
 - ロケール, [110](#)
- Mlatin
 - internal-gui.h, [371](#)
- Mlayouter
 - font.h, [356](#)

- MLIST_APPEND1
 - internal.h, 379
- MLIST_COPY1
 - internal.h, 380
- MLIST_DELETE1
 - internal.h, 380
- MLIST_FREE1
 - internal.h, 381
- MLIST_INIT1
 - internal.h, 379
- MLIST_INSERT1
 - internal.h, 380
- MLIST_PREPEND1
 - internal.h, 379
- MLIST_RESET
 - internal.h, 379
- Mlittle_endian
 - コード変換, 104
- MLocale
 - ロケール, 108
- mlocale.h, 426
 - mlocale_collate, 427
 - mlocale_ctype, 427
 - mlocale_messages, 427
 - mlocale_time, 427
- mlocale_collate
 - mlocale.h, 427
- mlocale_ctype
 - mlocale.h, 427
- mlocale_fini
 - internal.h, 393
- mlocale_init
 - internal.h, 393
- mlocale_messages
 - mlocale.h, 427
- mlocale_time
 - mlocale.h, 427
- mlocale_get_prop
 - ロケール, 111
- mlocale_set
 - ロケール, 111
- Mlocking_shift
 - コード変換, 106
- Mlong_form
 - コード変換, 105
- Mmap
 - 文字セット, 85
- Mmapfile
 - 文字セット, 84
- Mmax_advance
 - フォント, 158
- Mmax_code
 - 文字セット, 83
- Mmax_range
 - 文字セット, 83
- Mmaybe
 - コード変換, 107
- Mmethod
 - 文字セット, 82
- Mmin_char
 - 文字セット, 84
- Mmin_code
 - 文字セット, 83
- Mmin_range
 - 文字セット, 83
- Mmodifier
 - ロケール, 114
- Mname
 - 文字, 31
- Mnil
 - シンボル, 19
- Mnormal
 - フェース, 172
- modifier_list
 - MInputXIMArgIM, 297
- Moffset
 - 文字セット, 85
- Moff
 - フォント, 158
- Mparents
 - 文字セット, 84
- MPlist, 297
 - control, 298
 - func, 298
 - key, 298
 - next, 299
 - pointer, 298
 - val, 298
- Mplist
 - プロパティリスト, 26
- mplist
 - プロパティリスト, 21
- mplist_assq
 - plist.h, 441
- mplist_conc
 - plist.h, 441
- mplist_fini
 - internal.h, 391
- mplist_from_alist
 - plist.h, 440
- mplist_from_file
 - plist.h, 440
- mplist_from_plist
 - plist.h, 440
- mplist_from_string
 - plist.h, 441
- mplist_init
 - internal.h, 390
- mplist_pop_unref
 - plist.h, 441
- mplist_serialize
 - plist.h, 441

- `mplist_add`
 - プロパティリスト, 23
- `MPLIST_ADD_PLIST`
 - `plist.h`, 439
- `mplist_copy`
 - プロパティリスト, 21
- `mplist_deserialize`
 - プロパティリスト, 21
- `MPLIST_DO`
 - `plist.h`, 439
- `MPLIST_FIND`
 - `plist.h`, 439
- `mplist_find_by_key`
 - プロパティリスト, 24
- `mplist_find_by_value`
 - プロパティリスト, 25
- `MPLIST_FUNC`
 - `plist.h`, 436
- `mplist_get`
 - プロパティリスト, 22
- `mplist_get_func`
 - プロパティリスト, 23
- `MPLIST_INTEGER`
 - `plist.h`, 438
- `MPLIST_INTEGER_P`
 - `plist.h`, 437
- `MPLIST_KEY`
 - `plist.h`, 436
- `mplist_key`
 - プロパティリスト, 26
- `MPLIST_LENGTH`
 - `plist.h`, 439
- `mplist_length`
 - プロパティリスト, 25
- `MPLIST_MTEXT`
 - `plist.h`, 438
- `MPLIST_MTEXT_P`
 - `plist.h`, 437
- `MPLIST_NESTED_P`
 - `plist.h`, 437
- `MPLIST_NEXT`
 - `plist.h`, 436
- `mplist_next`
 - プロパティリスト, 25
- `MPLIST_PLIST`
 - `plist.h`, 439
- `MPLIST_PLIST_P`
 - `plist.h`, 437
- `mplist_pop`
 - プロパティリスト, 24
- `mplist_push`
 - プロパティリスト, 24
- `MPLIST_PUSH_PLIST`
 - `plist.h`, 440
- `mplist_put`
 - プロパティリスト, 22
- `mplist_put_func`
 - プロパティリスト, 23
- `MPLIST_PUT_PLIST`
 - `plist.h`, 440
- `mplist_set`
 - プロパティリスト, 25
- `MPLIST_SET_NESTED_P`
 - `plist.h`, 438
- `MPLIST_SET_VAL_FUNC_P`
 - `plist.h`, 438
- `MPLIST_STRING`
 - `plist.h`, 438
- `MPLIST_STRING_P`
 - `plist.h`, 437
- `MPLIST_SYMBOL`
 - `plist.h`, 438
- `MPLIST_SYMBOL_P`
 - `plist.h`, 437
- `MPLIST_TAIL_P`
 - `plist.h`, 437
- `MPLIST_VAL`
 - `plist.h`, 436
- `MPLIST_VAL_FUNC_P`
 - `plist.h`, 438
- `mplist_value`
 - プロパティリスト, 26
- `Mratio`
 - フェース, 170
- `MRealizedFace`, 299
 - `ascent`, 301
 - `ascii_rface`, 301
 - `average_width`, 302
 - `base_face_list`, 300
 - `box`, 301
 - `descent`, 301
 - `face`, 300
 - `font`, 300
 - `frame`, 300
 - `hline`, 301
 - `info`, 302
 - `layouter`, 301
 - `non_ascii_list`, 301
 - `rfont`, 300
 - `rfontset`, 300
 - `space_width`, 302
- `MRealizedFont`, 302
 - `ascent`, 304
 - `average_width`, 305
 - `baseline_offset`, 305
 - `descent`, 304
 - `driver`, 303
 - `encapsulating`, 304
 - `font`, 303
 - `fontp`, 305
 - `frame`, 303
 - `id`, 303

- info, 304
- layouter, 304
- max_advance, 305
- next, 305
- spec, 303
- x_ppem, 304
- y_ppem, 304
- MRealizedFontset
 - internal-gui.h, 369
- Mregistry
 - フォント, 157
- Mreset_at_cnl
 - コード変換, 105
- Mreset_at_eol
 - コード変換, 105
- Mresolution
 - フォント, 158
- Mreverse
 - フェース, 172
- Mrevision
 - 文字セット, 84
- Mrevision_number
 - コード変換, 106
- Mscreen
 - フレーム, 144
- Mscript
 - 文字, 30
- mscript_char_list
 - language.h, 396
- mscript_from_otf_tag
 - language.h, 396
- mscript_otf_tag
 - language.h, 396
- mscript_language_list
 - ロケール, 110
- mscript_list
 - ロケール, 110
- Msimple_case_folding
 - 文字, 32
- Msingle_shift
 - コード変換, 106
- Msingle_shift_7
 - コード変換, 106
- Msize
 - フォント, 158
- Msoft_dotted
 - m17n-core.h, 405
 - 文字, 32
- Mspacing
 - フォント, 157
- Mstretch
 - フォント, 157
- Mstring
 - シンボル, 19
- MSTRUCT_CALLOC
 - internal.h, 378
- MSTRUCT_CALLOC_SAFE
 - internal.h, 378
- MSTRUCT_MALLOC
 - internal.h, 377
- Mstyle
 - フォント, 157
- Msubset
 - 文字セット, 85
- Msubset_offset
 - 文字セット, 84
- Msuperset
 - 文字セット, 86
- MSymbol, 306
 - length, 307
 - managing_key, 306
 - name, 306
 - next, 307
 - plist, 307
- Msymbol
 - シンボル, 19
- msymbol
 - シンボル, 15
- msymbol_canonicalize
 - symbol.h, 445
- msymbol_deserializer
 - symbol.h, 445
- msymbol_fini
 - internal.h, 390
- msymbol_free_table
 - symbol.h, 444
- msymbol_init
 - internal.h, 390
- msymbol_list
 - symbol.h, 445
- msymbol_serializer
 - symbol.h, 445
- msymbol_with_len
 - symbol.h, 444
- msymbol_as_managing_key
 - シンボル, 15
- msymbol_exist
 - シンボル, 16
- msymbol_get
 - シンボル, 17
- msymbol_get_func
 - シンボル, 18
- msymbol_is_managing_key
 - シンボル, 16
- MSYMBOL_NAME
 - symbol.h, 444
- msymbol_name
 - シンボル, 16
- MSYMBOL_NAMELEN
 - symbol.h, 444
- msymbol_put
 - シンボル, 17

- msymbol_put_func
 - シンボル, 18
- Mt
 - シンボル, 19
- mt
 - MDrawTextItem, 233
 - MInputContext, 277
 - MTextProperty, 311
- MTABLE_ALLOCA
 - internal.h, 377
- MTABLE_CALLOC
 - internal.h, 376
- MTABLE_CALLOC_SAFE
 - internal.h, 377
- MTABLE_MALLOC
 - internal.h, 376
- MTABLE_REALLOC
 - internal.h, 377
- Mterritory
 - ロケール, 113
- MText, 307
 - allocated, 309
 - cache_byte_pos, 309
 - cache_char_pos, 309
 - control, 308
 - coverage, 308
 - data, 309
 - format, 308
 - nbytes, 308
 - nchars, 308
 - plist, 309
- Mtext
 - プロパティリスト, 26
- mtext
 - M-text, 42
- mtext-lbrk.c, 427
- mtext-wseg.c, 427
- mtext.c, 427
- mtext.h, 430
 - mtext_adjust_format, 432
 - mtext_bol, 432
 - mtext_byte_to_char, 431
 - mtext_cat_data, 432
 - mtext_char_to_byte, 431
 - mtext_enlarge, 431
 - mtext_eol, 432
 - mtext_from_data, 432
 - mtext_takein, 432
 - mtext_word_segment, 433
 - mtext_wseg_fini, 433
 - MTEXT_CAT_ASCII, 431
 - MTEXT_DATA, 431
 - POS_BYTE_TO_CHAR, 430
 - POS_CHAR_TO_BYTE, 430
- mtext_adjust_format
 - mtext.h, 432
- mtext_adjust_plist_for_change
 - textprop.h, 448
- mtext_adjust_plist_for_delete
 - textprop.h, 448
- mtext_adjust_plist_for_insert
 - textprop.h, 448
- mtext_bol
 - mtext.h, 432
- mtext_byte_to_char
 - mtext.h, 431
- mtext_cat_data
 - mtext.h, 432
- mtext_char_to_byte
 - mtext.h, 431
- mtext_copy_plist
 - textprop.h, 448
- mtext_enlarge
 - mtext.h, 431
- mtext_eol
 - mtext.h, 432
- mtext_fini
 - internal.h, 391
- mtext_free_plist
 - textprop.h, 448
- mtext_from_data
 - mtext.h, 432
- mtext_init
 - internal.h, 391
- mtext_prop_fini
 - internal.h, 391
- mtext_prop_init
 - internal.h, 391
- mtext_takein
 - mtext.h, 432
- mtext_word_segment
 - mtext.h, 433
- mtext_wseg_fini
 - mtext.h, 433
- mtext_allocated
 - internal.h, 385
- mtext_attach_property
 - テキストプロパティ, 69
- mtext_case_compare
 - M-text, 55
- mtext_cascmp
 - M-text, 55
- mtext_cat
 - M-text, 45
- MTEXT_CAT_ASCII
 - mtext.h, 431
- mtext_cat_char
 - M-text, 44
- mtext_character
 - M-text, 50
- mtext_chr
 - M-text, 50

- mtext_cmp
 - M-text, 51
- mtext_coll
 - ロケール, 113
- mtext_compare
 - M-text, 52
- mtext_copy
 - M-text, 47
- MTEXT_COVERAGE_ASCII
 - internal.h, 389
- MTEXT_COVERAGE_FULL
 - internal.h, 389
- MTEXT_COVERAGE_UNICODE
 - internal.h, 389
- mtext_cpy
 - M-text, 45
- mtext_cspn
 - M-text, 53
- MTEXT_DATA
 - mtext.h, 431
- mtext_data
 - M-text, 43
- mtext_del
 - M-text, 47
- mtext_deserialize
 - テキストプロパティ, 70
- mtext_detach_property
 - テキストプロパティ, 69
- mtext_dup
 - M-text, 44
- mtext_duplicate
 - M-text, 46
- MTEXT_FORMAT_MAX
 - M-text, 41
- MTEXT_FORMAT_US_ASCII
 - M-text, 41
- MTEXT_FORMAT_UTF_16
 - M-text, 57
- MTEXT_FORMAT_UTF_16BE
 - M-text, 41
- MTEXT_FORMAT_UTF_16LE
 - M-text, 41
- MTEXT_FORMAT_UTF_32
 - M-text, 58
- MTEXT_FORMAT_UTF_32BE
 - M-text, 41
- MTEXT_FORMAT_UTF_32LE
 - M-text, 41
- MTEXT_FORMAT_UTF_8
 - M-text, 41
- mtext_from_data
 - M-text, 42
- mtext_ftime
 - ロケール, 112
- mtext_get_prop
 - テキストプロパティ, 61
- mtext_get_prop_keys
 - テキストプロパティ, 63
- mtext_get_prop_values
 - テキストプロパティ, 62
- mtext_get_properties
 - テキストプロパティ, 68
- mtext_get_property
 - テキストプロパティ, 68
- mtext_getenv
 - ロケール, 112
- mtext_ins
 - M-text, 48
- mtext_ins_char
 - M-text, 49
- mtext_insert
 - M-text, 48
- MTEXT_LBO_AI_AS_ID
 - M-text, 42
- MTEXT_LBO_KOREAN_SP
 - M-text, 42
- MTEXT_LBO_MAX
 - M-text, 42
- MTEXT_LBO_SP_CM
 - M-text, 42
- mtext_len
 - M-text, 43
- mtext_line_break
 - M-text, 42
- mtext_lowercase
 - M-text, 56
- mtext_nbytes
 - internal.h, 385
- mtext_ncasecmp
 - M-text, 55
- mtext_ncat
 - M-text, 45
- mtext_nchars
 - internal.h, 385
- mtext_ncmp
 - M-text, 52
- mtext_ncpy
 - M-text, 46
- mtext_pbrk
 - M-text, 53
- mtext_pop_prop
 - テキストプロパティ, 65
- Mtext_prop_deserializer
 - テキストプロパティ, 71
- mtext_prop_range
 - テキストプロパティ, 66
- Mtext_prop_serializer
 - テキストプロパティ, 71
- mtext_property
 - テキストプロパティ, 67
- mtext_property_end
 - テキストプロパティ, 68

- mtext_property_key
 - テキストプロパティ, 67
- mtext_property_mtext
 - テキストプロパティ, 67
- mtext_property_start
 - テキストプロパティ, 68
- mtext_property_value
 - テキストプロパティ, 67
- mtext_push_prop
 - テキストプロパティ, 65
- mtext_push_property
 - テキストプロパティ, 69
- mtext_put_prop
 - テキストプロパティ, 63
- mtext_put_prop_values
 - テキストプロパティ, 64
- mtext_putenv
 - ロケール, 112
- mtext_rchr
 - M-text, 51
- MTEXT_READ_ONLY_P
 - internal.h, 385
- mtext_ref_char
 - M-text, 43
- mtext_replace
 - M-text, 49
- mtext_reset
 - internal.h, 386
- mtext_search
 - M-text, 54
- mtext_serialize
 - テキストプロパティ, 70
- mtext_set_char
 - M-text, 44
- mtext_spn
 - M-text, 53
- mtext_text
 - M-text, 54
- mtext_titlecase
 - M-text, 56
- mtext_tok
 - M-text, 54
- mtext_uppercase
 - M-text, 57
- MTextCoverage
 - internal.h, 389
- MTextFormat
 - M-text, 41
- MTextLineBreakOption
 - M-text, 42
- MTEXTPROP_CONTROL_MAX
 - テキストプロパティ, 61
- MTEXTPROP_END
 - textprop.h, 447
- MTEXTPROP_FRONT_STICKY
 - テキストプロパティ, 61
- MTEXTPROP_KEY
 - textprop.h, 447
- MTEXTPROP_NO_MERGE
 - テキストプロパティ, 61
- MTEXTPROP_REAR_STICKY
 - テキストプロパティ, 61
- MTEXTPROP_START
 - textprop.h, 447
- MTEXTPROP_VAL
 - textprop.h, 447
- MTEXTPROP_VOLATILE_STRONG
 - テキストプロパティ, 61
- MTEXTPROP_VOLATILE_WEAK
 - テキストプロパティ, 61
- MTextPropDeserializeFunc
 - テキストプロパティ, 60
- MTextProperty, 310
 - attach_count, 311
 - control, 311
 - end, 311
 - key, 311
 - mt, 311
 - start, 311
 - val, 311
- MTextPropertyControl
 - テキストプロパティ, 61
- MTextPropSerializeFunc
 - テキストプロパティ, 60
- Mtype
 - コード変換, 103
- multiple_sizes
 - MFont, 252
- Municode_bmp
 - font.h, 356
- Municode_full
 - font.h, 356
- Munify
 - 文字セット, 85
- Mutf
 - コード変換, 104
- Mvideomode
 - フェース, 170
- MWARNING
 - internal.h, 375
- Mweight
 - フォント, 156
- Mwidget
 - フレーム, 145
- Mx
 - フォント, 159
- Mxft
 - m17n-gui.h, 415
 - フォント, 159
- Mxim
 - 入力メソッド (GUI), 188

- name
 - M17NObjectArray, 199
 - MCharset, 202
 - MInputMethod, 290
 - MInputMethodInfo, 292
 - MSymbol, 306
- names
 - MFontPropertyTable, 261
- nbytes
 - MConverter, 211
 - MText, 308
- nchars
 - MConverter, 211
 - MText, 308
- next
 - M17NObjectArray, 199
 - MGlyphString, 273
 - MPList, 299
 - MRealizedFont, 305
 - MSymbol, 307
- next_to
 - MDrawGlyphInfo, 229
- nfeatures
 - MFontCapability, 255
- nfonts
 - MFontList, 260
- no_code_gap
 - MCharset, 203
- non_ascii_list
 - MRealizedFace, 301
- nparents
 - MCharset, 205
- object
 - MFontList, 259
- objects
 - M17NObjectArray, 199
- open
 - MFontDriver, 256
- open_jm
 - MInputDriver, 286
- orientation_reversed
 - MDrawControl, 220
- otf
 - MFontCapability, 254
- OTF_Tag
 - font.h, 349
- outer_hmargin
 - MFaceBoxProp, 237
- outer_vmargin
 - MFaceBoxProp, 237
- PACK_OTF_TAG
 - internal-flt.h, 365
- parents
 - MCharset, 205
- parse_event
 - MDeviceDriver, 218
- partial_update
 - MDrawControl, 223
- PATH_MAX
 - database.h, 337
- PATH_SEPARATOR
 - database.h, 337
- physical_ascent
 - MGlyphString, 272
- physical_descent
 - MGlyphString, 272
- plist
 - MInputContext, 279
 - MSymbol, 307
 - MText, 309
- plist.c, 433
 - mdebug_dump_plist, 434
- plist.h, 435
 - escape_mnemonic, 442
 - hex_mnemonic, 441
 - mplist_assq, 441
 - mplist_conc, 441
 - mplist_from_alist, 440
 - mplist_from_file, 440
 - mplist_from_plist, 440
 - mplist_from_string, 441
 - mplist_pop_unref, 441
 - mplist_serialize, 441
 - MPLIST_ADD_PLIST, 439
 - MPLIST_DO, 439
 - MPLIST_FIND, 439
 - MPLIST_FUNC, 436
 - MPLIST_INTEGER, 438
 - MPLIST_INTEGER_P, 437
 - MPLIST_KEY, 436
 - MPLIST_LENGTH, 439
 - MPLIST_MTEXT, 438
 - MPLIST_MTEXT_P, 437
 - MPLIST_NESTED_P, 437
 - MPLIST_NEXT, 436
 - MPLIST_PLIST, 439
 - MPLIST_PLIST_P, 437
 - MPLIST_PUSH_PLIST, 440
 - MPLIST_PUT_PLIST, 440
 - MPLIST_SET_NESTED_P, 438
 - MPLIST_SET_VAL_FUNC_P, 438
 - MPLIST_STRING, 438
 - MPLIST_STRING_P, 437
 - MPLIST_SYMBOL, 438
 - MPLIST_SYMBOL_P, 437
 - MPLIST_TAIL_P, 437
 - MPLIST_VAL, 436
 - MPLIST_VAL_FUNC_P, 438
- pointer
 - MPList, 298

- pos
 - MInputContext, 277
- POS_BYTE_TO_CHAR
 - mtext.h, 430
- POS_CHAR_TO_BYTE
 - mtext.h, 430
- preceding_text
 - MInputContextInfo, 283
- preedit
 - MInputContext, 277
- preedit_attrs
 - MInputXIMArgIC, 295
- preedit_changed
 - MInputContext, 278
- preedit_saved
 - MInputContextInfo, 282
- prev_from
 - MDrawGlyphInfo, 229
- prev_state
 - MInputContextInfo, 281
- produced
 - MInputContext, 275
- properties
 - MDatabaseInfo, 214
- property
 - MFace, 234
 - MFont, 251
 - MFontPropertyTable, 261
- ptr
 - MConverter, 211
- pushing_or_switching
 - MInputContextInfo, 284
- r2l
 - MFLTGlyphString, 248
- rbearing
 - MDrawGlyph, 226
 - MFLTGlyph, 244
 - MGlyphString, 272
- realize_face
 - MDeviceDriver, 215
- realized_face_list
 - MFrame, 266
- realized_font_list
 - MFrame, 266
- realized_fontset_list
 - MFrame, 266
- record
 - M17NObject, 198
- ref_count
 - M17NObject, 197
 - MCharset, 202
- ref_count_extended
 - M17NObject, 198
- region_add_rect
 - MDeviceDriver, 217
- region_from_rect
 - MDeviceDriver, 216
- region_to_rect
 - MDeviceDriver, 217
- render
 - MFontDriver, 257
- REPLACE_GLYPHS
 - internal-gui.h, 368
- res_class
 - MInputXIMArgIM, 296
- res_name
 - MInputXIMArgIM, 296
- result
 - MConverter, 211
- revision
 - MCharset, 204
- rface
 - MFrame, 265
 - MGlyph, 268
- rfont
 - MFLTFontForRealized, 242
 - MRealizedFace, 300
- rfontset
 - MRealizedFace, 300
- right_from
 - MDrawGlyphInfo, 229
- right_padding
 - MGlyph, 268
- right_to
 - MDrawGlyphInfo, 230
- SAFE_ALLOCA
 - internal.h, 378
- SAFE_FREE
 - internal.h, 378
- score
 - MFontScore, 262
- script
 - MFLTOfSpec, 249
 - MFontCapability, 254
- script_tag
 - MFontCapability, 254
- select
 - MFontDriver, 256
- set
 - MFLTGlyphAdjustment, 246
- simple
 - MCharset, 205
- size
 - M17NObjectArray, 199
 - M17NObjectRecord, 201
 - MCharsetISO2022Table, 206
 - MFont, 252
 - MFontPropertyTable, 260
 - MGlyphString, 270
 - MInputContextInfo, 281

- source
 - MFont, 251
- space_width
 - MFrame, 265
 - MRealizedFace, 302
- spacing
 - MFont, 251
- spec
 - MRealizedFont, 303
- spot
 - MInputContext, 277
- stack
 - MInputContextInfo, 284
- start
 - MTextProperty, 311
- state
 - MInputContextInfo, 281
- state_hook
 - MInputContextInfo, 283
- state_key_head
 - MInputContextInfo, 282
- state_pos
 - MInputContextInfo, 282
- states
 - MInputMethodInfo, 294
- status
 - MConverter, 211
 - MDatabaseInfo, 213
 - MInputContext, 277
- status_attrs
 - MInputXIMArgIC, 295
- status_changed
 - MInputContext, 277
- str
 - MFontCapability, 254
- STRING_CHAR
 - character.h, 319
- STRING_CHAR_ADVANCE
 - character.h, 320
- STRING_CHAR_ADVANCE_UTF16
 - character.h, 320
- STRING_CHAR_ADVANCE_UTF8
 - character.h, 319
- STRING_CHAR_AND_BYTES
 - character.h, 321
- STRING_CHAR_AND_UNITS
 - character.h, 321
- STRING_CHAR_AND_UNITS_UTF16
 - character.h, 320
- STRING_CHAR_AND_UNITS_UTF8
 - character.h, 320
- STRING_CHAR_UTF16
 - character.h, 319
- STRING_CHAR_UTF8
 - character.h, 318
- subset_max_code
 - MCharset, 205
- subset_min_code
 - MCharset, 205
- subset_offset
 - MCharset, 205
- SWAP_16
 - internal.h, 388
- SWAP_32
 - internal.h, 388
- sym
 - MFLTOtfSpec, 249
- symbol.c, 442
- symbol.h, 443
 - msymbol_canonicalize, 445
 - msymbol_deserializer, 445
 - msymbol_free_table, 444
 - msymbol_list, 445
 - msymbol_serializer, 445
 - msymbol_with_len, 444
 - MSYMBOL_NAME, 444
 - MSYMBOL_NAMELEN, 444
- tab_width
 - MDrawControl, 221
- tags
 - MFontCapability, 255
- text_ascent
 - MGlyphString, 272
- text_descent
 - MGlyphString, 272
- textprop.c, 446
- textprop.h, 447
 - dump_textplist, 449
 - mtext_adjust_plist_for_change, 448
 - mtext_adjust_plist_for_delete, 448
 - mtext_adjust_plist_for_insert, 448
 - mtext_copy_plist, 448
 - mtext_free_plist, 448
 - MTEXTPROP_END, 447
 - MTEXTPROP_KEY, 447
 - MTEXTPROP_START, 447
 - MTEXTPROP_VAL, 447
- tick
 - MFrame, 265
 - MGlyphString, 270
 - MInputContextInfo, 284
 - MInputMethodInfo, 294
- time
 - MDatabaseInfo, 213
- title
 - MInputMethodInfo, 293
- to
 - MDrawGlyph, 225
 - MDrawGlyphInfo, 228
 - MFLTGlyph, 243
 - MGlyphString, 271

- TOLOWER
 - character.h, [322](#)
- top
 - MGlyphString, [274](#)
- TOUPPER
 - character.h, [322](#)
- try_otf
 - MFontDriver, [258](#)
- two_dimensional
 - MDrawControl, [220](#)
- type
 - MFaceHLineProp, [238](#)
 - MFont, [251](#)
 - MGlyph, [268](#)
- u
 - M17NObject, [198](#)
- UINT_SIZE
 - character.h, [315](#)
- unified_max
 - MCharset, [204](#)
- union_rect_with_region
 - MDeviceDriver, [216](#)
- uniq_file
 - MDatabaseInfo, [213](#)
- UNIT_BYTES
 - character.h, [315](#)
- unmap_window
 - MDeviceDriver, [218](#)
- USE_SAFE_ALLOCA
 - internal.h, [378](#)
- used
 - M17NObjectArray, [199](#)
 - M17NObjectRecord, [201](#)
 - MCharsetISO2022Table, [207](#)
 - MFLTGlyphString, [248](#)
 - MFontPropertyTable, [261](#)
 - MGlyphString, [271](#)
 - MInputContextInfo, [281](#)
- USHORT_SIZE
 - character.h, [315](#)
- val
 - MPlist, [298](#)
 - MTextProperty, [311](#)
- vars
 - MInputContextInfo, [283](#)
 - MInputMethodInfo, [293](#)
- vars_saved
 - MInputContextInfo, [283](#)
- videomode
 - MFrame, [264](#)
- width
 - MDrawMetric, [231](#)
 - MFaceBoxProp, [236](#)
 - MFaceHLineProp, [239](#)
 - MGlyphString, [271](#)
 - width_limit
 - MGlyphString, [273](#)
 - win_info
 - MInputContextInfo, [283](#)
 - window_geometry
 - MDeviceDriver, [218](#)
 - with_cursor
 - MDrawControl, [222](#)
- x
 - MDrawGlyphInfo, [228](#)
 - MDrawMetric, [230](#)
 - MDrawPoint, [231](#)
 - MInputContext, [276](#)
- x_advance
 - MDrawGlyph, [225](#)
- x_off
 - MDrawGlyph, [225](#)
- x_ppem
 - MFLTFont, [240](#)
 - MRealizedFont, [304](#)
- xadv
 - MFLTGlyph, [243](#)
 - MFLTGlyphAdjustment, [246](#)
- xoff
 - MFLTGlyph, [244](#)
 - MFLTGlyphAdjustment, [246](#)
- y
 - MDrawGlyphInfo, [228](#)
 - MDrawMetric, [231](#)
 - MDrawPoint, [231](#)
 - MInputContext, [276](#)
- y_advance
 - MDrawGlyph, [225](#)
- y_off
 - MDrawGlyph, [225](#)
- y_ppem
 - MFLTFont, [240](#)
 - MRealizedFont, [304](#)
- yadv
 - MFLTGlyph, [243](#)
 - MFLTGlyphAdjustment, [246](#)
- yoff
 - MFLTGlyph, [244](#)
 - MFLTGlyphAdjustment, [246](#)
- はじめに, [5](#)
 - M17N_CORE_INITIALIZED, [9](#)
 - M17N_FINI, [8](#)
 - M17N_GUI_INITIALIZED, [9](#)
 - M17N_INIT, [8](#)
 - M17N_NOT_INITIALIZED, [9](#)
 - M17N_SHELL_INITIALIZED, [9](#)

- m17n_status, 9
- M17NLIB_MAJOR_VERSION, 7
- M17NLIB_MINOR_VERSION, 7
- M17NLIB_PATCH_LEVEL, 7
- M17NLIB_VERSION_NAME, 7
- M17NStatus, 8
- エラー処理, 189
 - m17n_memory_full_handler, 192
 - MERROR_CHAR, 191
 - MERROR_CHARSET, 191
 - MERROR_CHARTABLE, 191
 - merror_code, 192
 - MERROR_CODING, 191
 - MERROR_DB, 191
 - MERROR_DEBUG, 191
 - MERROR_DRAW, 191
 - MERROR_FACE, 191
 - MERROR_FLT, 191
 - MERROR_FONT, 191
 - MERROR_FONT_FT, 191
 - MERROR_FONT_OTF, 191
 - MERROR_FONT_X, 191
 - MERROR_FONTSET, 191
 - MERROR_FRAME, 191
 - MERROR_GD, 191
 - MERROR_IM, 191
 - MERROR_IO, 191
 - MERROR_LANGUAGE, 191
 - MERROR_LOCALE, 191
 - MERROR_MAX, 191
 - MERROR_MEMORY, 191
 - MERROR_MISC, 191
 - MERROR_MTEXT, 191
 - MERROR_NONE, 191
 - MERROR_OBJECT, 191
 - MERROR_PLIST, 191
 - MERROR_RANGE, 191
 - MERROR_SYMBOL, 191
 - MERROR_TEXTPROP, 191
 - MERROR_WIN, 191
 - MERROR_X, 191
 - MErrorCode, 191
- コア API, 9
 - M17N_FUNC, 11
 - M17NFunc, 11
- コード変換, 86
 - Mbom, 104
 - Mcharsets, 103
 - Mcode_unit, 104
 - Mcoding, 107
 - Mcoding_iso_8859_1, 101
 - MCODING_ISO_DESIGNATION_CTEXT, 92
 - MCODING_ISO_DESIGNATION_CTEXT_EXT, 92
 - MCODING_ISO_DESIGNATION_G0, 92
 - MCODING_ISO_DESIGNATION_G1, 92
 - MCODING_ISO_EIGHT_BIT, 91
 - MCODING_ISO_EUC_TW_SHIFT, 92
 - MCODING_ISO_FLAG_MAX, 92
 - MCODING_ISO_FULL_SUPPORT, 92
 - MCODING_ISO_ISO6429, 92
 - MCODING_ISO_LOCKING_SHIFT, 92
 - MCODING_ISO_LONG_FORM, 91
 - MCODING_ISO_RESET_AT_CNTL, 91
 - MCODING_ISO_RESET_AT_EOL, 91
 - MCODING_ISO_REVISION_NUMBER, 92
 - MCODING_ISO_SINGLE_SHIFT, 92
 - MCODING_ISO_SINGLE_SHIFT_7, 92
 - Mcoding_sjis, 103
 - MCODING_TYPE_CHARSET, 91
 - MCODING_TYPE_ISO_2022, 91
 - MCODING_TYPE_MISC, 91
 - MCODING_TYPE_UTF, 91
 - Mcoding_us_ascii, 101
 - Mcoding_utf_16, 102
 - Mcoding_utf_16be, 102
 - Mcoding_utf_16le, 102
 - Mcoding_utf_32, 102
 - Mcoding_utf_32be, 103
 - Mcoding_utf_32le, 103
 - Mcoding_utf_8, 101
 - Mcoding_utf_8_full, 102
 - MCodingFlagISO2022, 91
 - MCodingType, 91
 - mconv_buffer_converter, 93
 - mconv_decode, 95
 - mconv_decode_buffer, 96
 - mconv_decode_stream, 96
 - mconv_define_coding, 92
 - mconv_encode, 97
 - mconv_encode_buffer, 98
 - mconv_encode_range, 97
 - mconv_encode_stream, 98
 - mconv_free_converter, 94
 - mconv_getc, 99
 - mconv_gets, 100
 - mconv_list_codings, 92
 - mconv_putc, 100
 - mconv_rebind_buffer, 94
 - mconv_rebind_stream, 95
 - mconv_reset_converter, 94
 - mconv_resolve_coding, 92
 - mconv_stream_converter, 93
 - mconv_ungetc, 99
 - MCONVERSION_RESULT_INSUFFICIENT_DST, 90
 - MCONVERSION_RESULT_INSUFFICIENT_SRC, 90
 - MCONVERSION_RESULT_INVALID_BYTE, 90
 - MCONVERSION_RESULT_INVALID_CHAR, 90
 - MCONVERSION_RESULT_IO_ERROR, 90
 - MCONVERSION_RESULT_SUCCESS, 90
 - MConversionResult, 90

- Mdesignation, 104
- Mdesignation_cxtext, 105
- Mdesignation_cxtext_ext, 106
- Mdesignation_g0, 105
- Mdesignation_g1, 105
- Meight_bit, 105
- Meuc_tw_shift, 106
- Mflags, 103
- Mfull_support, 107
- Minvocation, 104
- Miso_2022, 104
- Miso_6429, 106
- Mlittle_endian, 104
- Mlocking_shift, 106
- Mlong_form, 105
- Mmaybe, 107
- Mreset_at_cxnti, 105
- Mreset_at_eol, 105
- Mrevision_number, 106
- Msingle_shift, 106
- Msingle_shift_7, 106
- Mtype, 103
- Mutf, 104
- シェル API, 76
- シンボル, 14
 - Mnil, 19
 - Mstring, 19
 - Msymbol, 19
 - msymbol, 15
 - msymbol_as_managing_key, 15
 - msymbol_exist, 16
 - msymbol_get, 17
 - msymbol_get_func, 18
 - msymbol_is_managing_key, 16
 - msymbol_name, 16
 - msymbol_put, 17
 - msymbol_put_func, 18
 - Mt, 19
- テキストプロパティ, 58
 - mtext_attach_property, 69
 - mtext_deserialize, 70
 - mtext_detach_property, 69
 - mtext_get_prop, 61
 - mtext_get_prop_keys, 63
 - mtext_get_prop_values, 62
 - mtext_get_properties, 68
 - mtext_get_property, 68
 - mtext_pop_prop, 65
 - Mtext_prop_deserializer, 71
 - mtext_prop_range, 66
 - Mtext_prop_serializer, 71
 - mtext_property, 67
 - mtext_property_end, 68
 - mtext_property_key, 67
 - mtext_property_mtext, 67
 - mtext_property_start, 68
 - mtext_property_value, 67
 - mtext_push_prop, 65
 - mtext_push_property, 69
 - mtext_put_prop, 63
 - mtext_put_prop_values, 64
 - mtext_serialize, 70
 - MTEXTPROP_CONTROL_MAX, 61
 - MTEXTPROP_FRONT_STICKY, 61
 - MTEXTPROP_NO_MERGE, 61
 - MTEXTPROP_REAR_STICKY, 61
 - MTEXTPROP_VOLATILE_STRONG, 61
 - MTEXTPROP_VOLATILE_WEAK, 61
 - MTextPropDeserializeFunc, 60
 - MTextPropertyControl, 61
 - MTextPropSerializeFunc, 60
- デバッグサポート, 192
 - mdebug_dump_all_symbols, 195
 - mdebug_dump_face, 194
 - mdebug_dump_im, 194
 - mdebug_dump_mtext, 194
 - mdebug_dump_symbol, 195
 - mdebug_hook, 194
- データベース, 72
 - MDatabase, 73
 - mdatabase_define, 74
 - mdatabase_dir, 76
 - mdatabase_find, 74
 - mdatabase_list, 74
 - mdatabase_load, 75
 - mdatabase_tag, 75
- フェイス, 163
 - Mbackground, 170
 - Mbox, 171
 - Mface, 177
 - mface, 167
 - mface_black, 175
 - mface_blue, 176
 - mface_bold, 173
 - mface_bold_italic, 174
 - mface_copy, 167
 - mface_cyan, 176
 - mface_equal, 167
 - mface_from_font, 168
 - mface_get_hook, 168
 - mface_get_prop, 168
 - mface_green, 176
 - mface_italic, 173
 - mface_large, 175
 - mface_magenta, 177
 - mface_medium, 173
 - mface_merge, 167
 - mface_normal_video, 172
 - mface_normalsize, 174
 - mface_put_hook, 169
 - mface_put_prop, 169
 - mface_red, 176

- mface_reverse_video, 172
- mface_small, 174
- mface_underline, 173
- mface_update, 169
- mface_white, 175
- mface_x_large, 175
- mface_x_small, 174
- mface_xx_large, 175
- mface_xx_small, 174
- mface_yellow, 176
- MFaceHookFunc, 166
- Mfontset, 171
- Mforeground, 170
- Mhline, 171
- Mhook_arg, 172
- Mhook_func, 171
- Mnormal, 172
- Mratio, 170
- Mreverse, 172
- Mvideomode, 170
- フォント, 146
 - Madstyle, 157
 - Mfamily, 156
 - mfont, 150
 - mfont_check, 155
 - mfont_close, 156
 - mfont_copy, 151
 - mfont_encapsulate, 156
 - mfont_find, 153
 - mfont_freetype_path, 159
 - mfont_from_name, 154
 - mfont_get_prop, 152
 - mfont_list, 154
 - mfont_list_family_names, 155
 - mfont_match_p, 155
 - mfont_name, 154
 - mfont_open, 155
 - mfont_parse_name, 151
 - mfont_put_prop, 152
 - mfont_resize_ratio, 154
 - mfont_selection_priority, 152
 - mfont_set_encoding, 153
 - mfont_set_selection_priority, 153
 - mfont_unparse_name, 151
 - Mfontconfig, 159
 - Mfontfile, 158
 - Mfoundry, 156
 - Mfreetype, 159
 - Mmax_advance, 158
 - Motf, 158
 - Mregistry, 157
 - Mresolution, 158
 - Msize, 158
 - Mspacing, 157
 - Mstretch, 157
 - Mstyle, 157
 - Mweight, 156
 - Mx, 159
 - Mxft, 159
- フォントセット, 160
 - mfontset, 161
 - mfontset_copy, 161
 - mfontset_lookup, 162
 - mfontset_modify_entry, 161
 - mfontset_name, 161
- フレーム, 141
 - Mcolormap, 145
 - Mdepth, 145
 - Mdevice, 144
 - Mdisplay, 144
 - Mdrawable, 145
 - Mfont, 145
 - Mfont_ascent, 146
 - Mfont_descent, 146
 - Mfont_width, 145
 - mframe, 142
 - mframe_default, 146
 - mframe_get_prop, 143
 - Mgd, 145
 - Mscreen, 144
 - Mwidget, 145
- プロパティリスト, 19
 - Minteger, 26
 - Mplist, 26
 - mplist, 21
 - mplist_add, 23
 - mplist_copy, 21
 - mplist_deserialize, 21
 - mplist_find_by_key, 24
 - mplist_find_by_value, 25
 - mplist_get, 22
 - mplist_get_func, 23
 - mplist_key, 26
 - mplist_length, 25
 - mplist_next, 25
 - mplist_pop, 24
 - mplist_push, 24
 - mplist_put, 22
 - mplist_put_func, 23
 - mplist_set, 25
 - mplist_value, 26
 - Mtext, 26
- ロケール, 107
 - Mcodeset, 114
 - Miso639_1, 113
 - Miso639_2, 113
 - mlanguage_code, 109
 - mlanguage_list, 109
 - mlanguage_name_list, 109
 - mlanguage_text, 110
 - MLocale, 108
 - mlocale_get_prop, 111

- [mlocale_set, 111](#)
 - [Mmodifier, 114](#)
 - [mscript_language_list, 110](#)
 - [mscript_list, 110](#)
 - [Mterritory, 113](#)
 - [mtext_coll, 113](#)
 - [mtext_ftime, 112](#)
 - [mtext_getenv, 112](#)
 - [mtext_putenv, 112](#)
- [入力メソッド \(GUI\), 187](#)
 - [minput_event_to_key, 188](#)
 - [minput_gui_driver, 188](#)
 - [Mxim, 188](#)
- [入力メソッド \(基本部分\), 114](#)
 - [Mconfigured, 134](#)
 - [Mcustomized, 134](#)
 - [Minherited, 134](#)
 - [minput_assign_command_keys, 130](#)
 - [minput_callback, 131](#)
 - [MINPUT_CANDIDATES_CHANGED_MAX, 118](#)
 - [Minput_candidates_done, 132](#)
 - [Minput_candidates_draw, 133](#)
 - [MINPUT_CANDIDATES_INDEX_CHANGED, 118](#)
 - [MINPUT_CANDIDATES_LIST_CHANGED, 118](#)
 - [MINPUT_CANDIDATES_SHOW_CHANGED, 118](#)
 - [Minput_candidates_start, 132](#)
 - [minput_close_im, 119](#)
 - [minput_config_command, 124](#)
 - [minput_config_file, 127](#)
 - [minput_config_variable, 126](#)
 - [minput_create_ic, 119](#)
 - [minput_default_driver, 134](#)
 - [Minput_delete_surrounding_text, 133](#)
 - [minput_destroy_ic, 119](#)
 - [Minput_driver, 135](#)
 - [minput_driver, 135](#)
 - [minput_filter, 120](#)
 - [Minput_focus_in, 134](#)
 - [Minput_focus_move, 134](#)
 - [Minput_focus_out, 133](#)
 - [minput_get_command, 122](#)
 - [minput_get_commands, 130](#)
 - [minput_get_description, 122](#)
 - [Minput_get_surrounding_text, 133](#)
 - [minput_get_title_icon, 121](#)
 - [minput_get_variable, 125](#)
 - [minput_get_variables, 128](#)
 - [minput_list, 128](#)
 - [minput_lookup, 120](#)
 - [Minput_method, 131](#)
 - [minput_open_im, 119](#)
 - [minput_parse_im_names, 131](#)
 - [Minput_preedit_done, 132](#)
 - [Minput_preedit_draw, 132](#)
 - [Minput_preedit_start, 131](#)
 - [Minput_reset, 133](#)
 - [minput_reset_ic, 121](#)
 - [minput_save_config, 127](#)
 - [Minput_set_spot, 133](#)
 - [minput_set_spot, 120](#)
 - [minput_set_variable, 129](#)
 - [Minput_status_done, 132](#)
 - [Minput_status_draw, 132](#)
 - [Minput_status_start, 132](#)
 - [Minput_toggle, 133](#)
 - [minput_toggle, 121](#)
 - [MInputCallbackFunc, 118](#)
 - [MInputCandidatesChanged, 118](#)
- [文字, 27](#)
 - [Mbidi_category, 31](#)
 - [Mblock, 33](#)
 - [Mcase_mapping, 33](#)
 - [Mcased, 32](#)
 - [Mcategory, 31](#)
 - [mchar_define_property, 29](#)
 - [mchar_get_prop, 29](#)
 - [mchar_get_prop_table, 30](#)
 - [MCHAR_MAX, 28](#)
 - [mchar_put_prop, 29](#)
 - [Mcombining_class, 31](#)
 - [Mcomplicated_case_folding, 32](#)
 - [Mname, 31](#)
 - [Mscript, 30](#)
 - [Msimple_case_folding, 32](#)
 - [Msoft_dotted, 32](#)
- [文字セット, 77](#)
 - [Maliases, 84](#)
 - [Mascii_compatible, 83](#)
 - [mchar_decode, 80](#)
 - [mchar_define_charset, 79](#)
 - [mchar_encode, 80](#)
 - [MCHAR_INVALID_CODE, 79](#)
 - [mchar_list_charset, 80](#)
 - [mchar_map_charset, 81](#)
 - [mchar_resolve_charset, 80](#)
 - [Mcharset, 86](#)
 - [Mcharset_ascii, 81](#)
 - [Mcharset_binary, 82](#)
 - [Mcharset_iso_8859_1, 82](#)
 - [Mcharset_m17n, 82](#)
 - [Mcharset_unicode, 82](#)
 - [Mdefine_coding, 84](#)
 - [Mdimension, 83](#)
 - [Mfinal_byte, 83](#)
 - [Mmap, 85](#)
 - [Mmapfile, 84](#)
 - [Mmax_code, 83](#)
 - [Mmax_range, 83](#)
 - [Mmethod, 82](#)
 - [Mmin_char, 84](#)

- Mmin_code, [83](#)
- Mmin_range, [83](#)
- Moffset, [85](#)
- Mparents, [84](#)
- Mrevision, [84](#)
- Msubset, [85](#)
- Msubset_offset, [84](#)
- Msuperset, [86](#)
- Munify, [85](#)
- 文字テーブル, [33](#)
 - Mchar_table, [38](#)
 - MCharTable, [34](#)
 - mchartable, [35](#)
 - mchartable_lookup, [35](#)
 - mchartable_map, [37](#)
 - mchartable_max_char, [35](#)
 - mchartable_min_char, [35](#)
 - mchartable_range, [37](#)
 - mchartable_set, [36](#)
 - mchartable_set_range, [36](#)
- 管理下オブジェクト, [12](#)
 - m17n_object, [12](#)
 - m17n_object_ref, [13](#)
 - m17n_object_unref, [13](#)
- 表示, [177](#)
 - mdraw_clear_cache, [186](#)
 - mdraw_coordinates_position, [183](#)
 - mdraw_default_line_break, [185](#)
 - mdraw_glyph_info, [184](#)
 - mdraw_glyph_list, [184](#)
 - mdraw_image_text, [181](#)
 - mdraw_line_break_option, [186](#)
 - mdraw_per_char_extents, [185](#)
 - mdraw_text, [179](#)
 - mdraw_text_extents, [182](#)
 - mdraw_text_items, [185](#)
 - mdraw_text_per_char_extents, [182](#)
 - mdraw_text_with_control, [181](#)
 - MDrawRegion, [179](#)
 - MDrawWindow, [179](#)