cadence®

Allegro® Front-End CPM Directive Reference Guide

Product Version 23.1
September 2023

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

1
Introduction to CPM Files 29
Local Project Files 29
Project File (.cpm) example 30
User-Specific Project Settings File 31
Site Project File 31
Creating Site Project File for All Projects 31
Custom Site Environment 32
Installation Project File 36
Locking Directives 37
Project Settings at Different Levels 37
Locking a Directive 37
Locking Support for Directives at Different Levels 38
Reading Settings from CPM Files at Different Levels 39

2
Allegro Design Entry HDL CPM Directives 43

Design Synchronization CPM Directives 48

Design Entry HDL Utilities CPM Directives 49
Allegro FPGA System Planner CPM Directives 50
Allegro Design Publisher CPM Directives 50
Rules Checker Directives 50
CREFERHDL CPM Directives 51
BOM-HDL Directives 51
Design Variance CPM Directives 51
Archiver CPM Directives 51

Packager-XL Directives 52

System Connectivity Manager CPM Directives 54

Schgen CPM Directives 55

Part Developer CPM Directives 57

Engineering Data Management Directives (EDM) 61

Start Design 62
Allegro EDM Flow Manager 62
Part Information Manager 62
Library Revision Manager 62
Cache Management 63

Allegro Design Management (Team Design) Directives 64

Allegro System Capture CPM Directives 65

Schematic Canvas CPM Directives 65
Schematic Grid CPM Directive 67
Object Formatting CPM Directives 67
Directives for the LINE Object 69
Directives for the TEXT Objects 69
Other Directives Related to Objects 69
Packaging CPM Directives 69
TOC and Table Object CPM Directives 70
Design Integrity CPM Directives 70

3
A Directives 73

ADD_AUTHOR_TO_COMMENT 74

ADD_BBOX . . . 75
ADD_BLOCK_DIAGRAM . . . 76
ADD_COMMENTS . . . 77
ADD_COMMENTS_TO_INSTANCES . . . 78
ADD_COMMENTS_TO_NETS . . . 79
ADD_COMMENTS_TO_PINS . . . 80
ADD_DATE_TO_COMMENT . . . 81
ADD_TEXT_TO_COMMENT . . . 82
ADD_TIME_TO_COMMENT . . . 83
ALIASBODY . . . 84
ALLOW_4WAY_JUNCTION . . . 85
ALLOW_COMP_OVERLAP . . . 86
ALLOW_EMPTY_LOCK_FILE . . . 87
ALLOW_FOOTPRINT_COMPATIBILITY_CHECK . . . 88
ALLOW_HFS_SWAPS . . . 89
ALLOW_IMPORT_DESIGN_AT_SITE_UNIT . . . 90
ALLOW_IMPORT_DESIGN_NUDGE_OFFGRID_OBJECTS . . . 91
ALLOW_IMPORT_LIBCELL_MISMATCH . . . 92
ALLOW_INCOMPATIBLE_JEDEC_TYPE . . . 93
ALLOW_PAGE_LOCKING . . . 94
ALLOW_PINTEXT_SWAP . . . 95
ALLOW_POWER_PINS . . . 96
ALLOW_PROP_ILLEGAL_OBJECT . . . 97
ALLOW_PROP_ON_PROP . . . 98
ALLOW_PROPERTY_LOCKING . . . 99
ALLOW_PTFVALUE_INITIAL_BLANKS . . . 100
ALLOW_SINGLE_NAVLINK_PER_PAGE . . . 101
ALLOWED_ALTERNATE_PART_PROP . . . 102
ALTERNATE_FILL_COLOR . . . 103
ANNOTATE . . . 104
ANNOTATE_ALL_PROPS . . . 106
ANNOTATE_GLOBAL_PTF_PROPS . . . 107
ANNOTATE_PART_NAME . . . 108
APPLY <OBJECT> STYLE . . . 109
APPLY_NAVLINKS_STYLE . . . 110
APPLY_NETGROUP_STYLE . . . 111

APPLY_TABLE_STYLE 112
APPLY_VARIANT_INST_STYLE 113
APPLY_VARIANT_PROPERTY_STYLE 114
ARC_COLOR 115
ASK_RENAME_SIGNAME_OPTION 116
ASSOC_PARENT_PIN_SPACING 117
ATTRIBUTEDATATOOLFORM 118
ATTRIBUTEFILTER 119
ATTRIBUTEJAVAFORM 120
ATTRIBUTES_DIR 121
AUDIT_DISALLOW_WAIVE 122
AUDIT_DISALLOW_WAIVE_RULES 123
AUDIT_ERROR 124
AUDIT_FATAL 125
AUDIT_INFO 126
AUDIT_WARNING 127
AUTO_CONNECT_DPLEG 128
auto_fix_ptf 129
auto_load_schematic_instances 130
AUTO_NAMEONBUS_BITS 131
AUTO_NETNAMEON_POWERNET 132
AUTO_SIGNAME_LOC 133
AUTO_UPDATE_DEFAULT_MODELS 134
auto_update_minor_cell 135
auto_update_minor_ptf 136
AUTO_UPDATE_PARTS_ON_START 138
auto_update_schematic 139
AUTO_XNETS_USING_GATES 140
AUTODOT 141
AUTOHEAVY 142
AUTOPAN 143
AUTOPATH 144
AUTOROUTE 145
AUTOSIZE_MARGIN 146

4
B Directives 1

B2F_OVERWRITE_CONSTRAINTS 2
BACKANNOTATE_FEEDBACK 4
BACKANNOTATE_FORWARD 5
BACKGROUND_COLOR 6
BASE_NET_OVERLAY 8
BASENET_OMIT_SYNONYM 9
BBOX_COLOR 10
BBOX_SHOWN_ON_FONT_MOVE 11
BLACKANDWHITEPDF 12
BLOCK_DIAGRAM_CELL 13
BLOCK_DIAGRAM_LIB 14
BLOCK_PIN_SHAPE_MINIMUMPINSPACING 15
BLOCK_REF_DES_PATTERN 16
BLOCK_TITLE_TOP 17
BOM_FOLDER 18
BUS_TAP_SYMBOL_ROT<angle> 19

5
C Directives 21

CAPSLOCK 22
CAPTURE_CACHE_LIBRARY_PATH 23
CAPTURE_STANDARD_LIB 24
CAS_PIN_PATTERN 25
CATPATH 26
CDS_<CATEGORY_NAME>_FONT 27
CDS_<CATEGORY_NAME>_FONTCOLOR 29
CDS_<CATEGORY_NAME>_FONTEFFECTS 31
CDS_<CATEGORY_NAME>_FONTSIZE 33
CDS_<CATEGORY_NAME>_FONTSTYLE 35
CDS_ENABLE_FONTS 37
CENTRAL_INDEX_PATH 38
CHECK_ARCS_AT_SAME_LOCATION 40

CHECK_GLOBAL_LOCAL_SHORT 41
CHECK_GRID_ON_TAP 42
CHECK_HIDDEN_WIRES 43
CHECK_IMAGE_OVERLAP_OBJECT 44
check_injected_order 45
check_local_modified 46
CHECK_MISSING_PINS 47
CHECK_MOVE_SHORT 48
CHECK_MULTIPLE_IBD_PROPS 49
CHECK_NONSYNCPROPS 50
CHECK_ON_WRITE 51
CHECK_PACK_SEC_TYPE_PROPS 52
CHECK_PACK_SYNC_ON_IMPORT 53
CHECK_PAGE_NUMBER_SYNCH 54
CHECK_PIN_WIRE_DIST_THRESH 55
CHECK_PINS_AT_ORIGIN 56
CHECK_PINS_NEAR_WIRE_ENDPT 57
CHECK_PROP_PLACE_HOLDERS 58
CHECK_SHORTED_PINS 59
CHECK_SIGNAL_NAMES 60
CHECK_SYMBOL_PLACE_HOLDERS 61
CHECK_SYMBOL_SIGNAL_NAMES 62
CHECK_SYMBOLS_AT_SAME_LOCATION 63
CHECK_TWO_WIRES_AT_PINS 64
CHECK_UNCONNECTED_WIRES 65
CHECK_VOLTAGE_ON_HDL 66
CLOCK_PIN_PATTERN 67
CLICK_TO_TYPE 68
CNAME_CELL 69
CNAME_DEPTH 70
CNAME_LIB 71
CNAME_VIEW 72
COMBINED_MARKERS_ON_HIERWRITE 73
COMMENT_COLOR 74
COMP_COLOR 75
COMP_DEF_PROP 76

COMP_INST_PROP 78
COMPONENT_BROWSER 79
ConceptSetup_Assertion_Read 80
ConceptSetup_Assertion_UseMinusInChips 81
ConceptSetup_Assertion_Write 82
ConceptSetup_SplitPart_AddSwapInfo 83
ConceptSetup_SplitPart_SymbolProp 84
CONFIRM_WRITE 85
CONNECTION_SWAP_PINS 86
CP_NO_OF_COL_TOC 87
CP_NO_OF_ROW_TOC 88
CP_TOC_COL_<column_number> 89
CP_TOC_COL_<column_number>_LABEL 90
CREATE_CACHE_PROJECT 91
CREATE_USER_PROP 92
CREF_DATA_FILE 93
CS_PIN_PATTERN 94
CTAP 95
CTRLLMB_DRAGSELECT 96
CTRLRMB_CONTEXTMENU 97
CURRENTPDFVIEWER 98
CURRENTPDFVIEWERPATH 99
CURSOR_SHAPES 100
CUSTOM_CDSLIB_SEARCH 101

6
D Directives 103

DAO_Timeout 104
DASHBOARD_SHOW_LOGICAL_HIERARCHY 105
DASHBOARD_SHOW_PHYSICAL_HIERARCHY 107
DASHBOARD_SHOW_WORKING_DESIGN 109
DataCompress 111
Datasheet_URL 112
DATATIPS_PROP_NAME 113
Default_Diffpair_Value 119

DEFAULT_PAGE_BORDER_NAME 120
DEFAULT_PAGE_BORDER_VERSION 121
DEFAULT_PHYS_DES_PREFIX 122
Default_Search_Tab 123
Default_ShoppingCart_Quantity 124
Default_Zoom_Factor 125
DELETE_ASCII 126
DELETE_BINARY 127
delete_folders_on_copyproj 128
DELETE_UNATTACHED_INVISIBLE_PROPS 129
DESIGN_TYPE 130
Detail_Tab_Order 131
DIFFPAIR_PIN_SUFFIX_N 132
DIFFPAIR_PIN_SUFFIX_P 133
DiffPair_Recognition_Rules 134
DISABLE_SMARTPDF_SMART_FEATURES 135
DISALLOW_MULTIUSER_PAGE_OVERWRITE 136
DISCONNECT_PIN_TEXT_NAME 137
DISPLAY_UNCONNECTED_PINS 138
Display_URL 139
DO_NOT_HONOR_ANNOTATIONS 140
DOC_GRID_MULTIPLE 141
DOC_GRID_SIZE 142
DOC_GRID_TOGGLE 143
DOC_GRID_TYPE 144
DONT_ALLOW_UPREV 145
DONT_FORCE_ORIGIN_ONGRID 146
DONT_SET_OFFSET_PINNUMBER 147
DONT_SHOW_CM_DLG 148
DOT_COLOR 149
DOTS 150
DPPIN_PREFIX 151
DPPIN_RULES 152
DPSIG_PREFIX 153
DPSIG_RULES 154
DRAWING_BROWSER 155

DRC_ERROR 156
DRC_INFO 158
DRC_WARN 159
dump_FileName 160

7
E Directives 161

EDIT_PHYSICAL_NET_NAME 162
EDIT_PHYSICAL_SPACING_CONSTRAINTS 163
EDITABLE_IMPORT_TABLE 164
ELECTRICAL_CONSTRAINTS 165
ELECTRICAL_LOW_STRESS_LEVEL 166
ELECTRICAL_OVER_STRESS_LEVEL 167
ENABLE_ALPHANUMERIC_CUSTOMKEYS 168
ENABLE_FONT_BASED_BBOX_COMPUTATION 169
ENABLE_SEL_LOGICAL_UPDATE_VDD 170
ERR_OK_NET_ONE_PIN_MULTI_NODES 171
ERROR_MESSAGES 172
ERROR_NO_CSB_FILES 173
ERROR_ON_PARTIAL_INSTANTIATION_OF_HSS 174
ETCH_REMOVAL 175
exclude_autoupdate_props 176
EXCLUDE_FILE_PATH 177
EXCLUDE_PPT 178
EXCLUDE_VIEW 179
EXPLICIT_BASE_NET_IDENTIFIER 180
EXPORT 181
Export_Csv_Delimeter 184
Export_Csv_Replace_<property> 185
Export_ViewLogic_Visibility_<property> 186
EXTERNAL_ALLEGRO_BOARD_FOLDER 187

8
F Directives 189

F2B_OVERWRITE_CONSTRAINTS 190

FATAL_MESSAGES 191
FEEDBACK 192
FIDUCIAL_FOOTPRINT_NAME_PATTERN 194
FIDUCIAL_MIN_NUMBER 195
FILTER_CONFLICTING_PROP 196
FILTER_PROPERTY 198
FilterZeroNet 199
FLATTEN_BLOCK_ON_ADD 200
FORCE_PTF_ENTRY 201
FORCE_SUBDESIGN 202
FORMAT_CREF_REPORTS 203

9
G Directives 205

GEN_PSTFILES 206
GEN_SUBDESIGN 207
GENERATE_FLATTENED_SCHEMATIC 208
GENERATE_SCH_METADATA 209
GENERATE_SEPARATE_CELL 210
GENERATE_TDD_NETLIST 211
GENERATE_XR_FOR_ALL_NETS 212
Global_Modify_Pin_Graphics 213
GLOBAL_NETS 214
GRAPHIC_BLOCK_FILL_COLOR 215
GRAPHIC_BLOCK_LINE_COLOR 216
GRAPHIC_CONNECTOR_FILL_COLOR 217
GRAPHIC_CONNECTOR_LINE_COLOR 218
GRID_DISPLAY_ENABLED 219
GRID_DISPLAY_MULTIPLE 220
GRID_DOC_SNAP_FRACTION 221
GRID_PIN_PITCH 223
GRID_SNAP_FRACTION 224
GRID_STYLE 225
GRID_UNIT_MEASURE 226
GND_PIN_NAME 227

GUIDE_LINES_COLOR_DARK 229
GUIDE_LINES_COLOR_LIGHT 230

10
H Directives 231

HARD_LOC_SEC 232
HARD_REFDES_CONFLICT_RESOLVE 233
HIDE_HIERARCHY_PAGES 234
HIDE_INSTANCE_NAME 235
HIDE_SHEET_NUMBER 236
HIGHLIGHT_COLOR 237
HOLE_FOOTPRINT_NAME_PATTERN 238
HONOR_PPTOPTION_ON_ADD_OR_REPLACE 239
HPF_BATCH 240
HPF_BUS_SCALEFACTOR 241
HPF_FONT 242
HPF_PAGESIZE 243
HPF_PLOT_PAGESIZE 244
HPF_PLOTTER 245
HPF_SCALEFACTOR 246
HPF_SCALETYPE 247
HPF_SPEC_PLOT_PAGESIZE 248
HPF_WIRE_SCALEFACTOR 249
HYPERLINK_HIGHLIGHTED_DARK_THEME_COLOR 250
HYPERLINK_HIGHLIGHTED_LIGHT_THEME_COLOR 251
HYPERLINK_VISTED_COLOR 252
HYPERLINKS 253

11
I Directives 1

IGNORE_BLOCK_WITHOUT_SCHEMATIC 2
IGNORE_BUNDLED_CONSTRAINTS_ERROR_ON_ZERO_NODE_NETS 3
IGNORE_FIXED 4
IGNORE_INSTANCE_WITH_ERRORS 5
IGNORE_VAR_STATUS_COL 6

ILLEGAL_NET_NAME_CHAR 7
IMAGE* Directives 8
IMAGE_ARC_COLOR 9
IMAGE_BACKGROUND_COLOR 10
IMAGE_COLOR_MODE_PRINT 11
IMAGE_DEFAULT_DPI 12
IMAGE_DOT_COLOR 13
IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING 14
IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_PASTING 15
IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_NAMES_WHEN_PASTING 16
IMAGE_NOTE_COLOR 17
IMAGE_OCCPROP_COLOR 18
IMAGE_PROP_COLOR 19
IMAGE_SYMBOL_COLOR 20
IMAGE_WIRE_COLOR 21
Import_AllegroFtprint_DefaultPinType 22
Import_APD_PwrGndNCPinsInGlobalSection 23
Import_APD_strippinnum 24
Import_Csv_ApplyVectorConversion 25
Import_Csv_Delimeter 26
Import_CSV_GenerateSymbolForNoLocation 27
Import_Csv_LowAssertFlag 28
Import_Csv_Replace_assertion 29
Import_Csv_Replace_assertionchar 30
Import_Csv_Replace_DIFFPAIRPINSNEG 31
Import_Csv_Replace_DIFFPAIRPINSPOS 32
Import_Csv_Replace_jedectype 33
Import_Csv_Replace_loadsetupfile 34
Import_Csv_Replace_packagename 35
Import_Csv_Replace_pinlocation 36
Import_Csv_Replace_pinname 37
Import_Csv_Replace_pinnumber 38
Import_Csv_Replace_pinposition 39
Import_Csv_Replace_pinshape 40

Import_Csv_Replace_pintype 41
Import_Csv_Replace_symbol 42
IMPORT_DEHDL_PNS_CONSTRAINTS 43
Import_DML_Braces_TreatedAs_Vector 44
IMPORT_DEHDL_SKIP_ASCII 45
Import_DML_DefaultPinType 46
Import_DML_Pins_DefaultVector 47
Import_DML_PwrGndNCPinsInGlobalSection 48
Import_FPGA_DefaultPinType 49
Import_FPGA_PwrGndNCPinsInGlobalSection 50
Import_FPGA_Standard_PartNameAsCellName 51
Import_FPGA_Xilinx_CSVSeparationChar 52
Import_FPGA_Xilinx_SeparationChar 53
Import_IBIS_With_Ibischk4 54
Import_IBIS_With_Unchanged_ModelName 55
IMPORT_SOURCE_AT_SITE_UNIT 56
Import_Text_Braces_TreatedAs_Vector 57
Import_Text_DefaultPinType 58
Import_Text_Pins_DefaultVector 59
Import_Text_PwrGndNCPinsInGlobalSection 60
Import_ViewLogic_gridratio 61
IN_PORT_PIN_SIDE 62
INCLUDE_PPT 63
INCREMENTAL_RUN 65
INFORMATIONAL_MESSAGES 66
INOUT_PORT_PIN_SIDE 67
INPORT 68
INPUT_SCRIPT 69
INST_BLOCKAGE_MARGIN 70
INST_DEFAULT_ALIGNMENT 71
INST_DEFAULT_PROP_SIZE 72
INST_DEFAULT_VISIBILITY 73
Instantiation_Packaging_Validation_Type 74
IOPORT 75
Import_IBIS_With_Dmlcheck 76
ITEM_OPACITY 77

12
L Directives . . . 79
LAST_BOARD_FILE . . . 80
LAST_OUTPUT_FILE . . . 81
LAST_TEMPLATE_FILE . . . 82
LAST_VARIANT_FILE . . . 83
When Operated Under Variant BOM Report . . . 83
When Operated Under Variant Editor . . . 83
LastFlow . . . 85
LAUNCH_OPTION . . . 86
LEFT_IN_OFFPAGE . . . 87
LEFT_IN_ROT . . . 88
LEFT_IO_OFFPAGE . . . 89
LEFT_IO_ROT . . . 90
LEFT_OUT_OFFPAGE . . . 91
LEFT_OUT_ROT . . . 92
LIBRARY_BROWSER . . . 93
LINE_CAP_STYLE . . . 94
LINE_COLOR . . . 95
LINE_JOIN_STYLE . . . 96
LINE_STYLE . . . 97
LINE_WIDTH . . . 98
LIST_VALID_VERSIONS_METADATA . . . 99
LOCK_FILE_PERM . . . 100
LOGIC_DOT_RADIUS . . . 101
LOGIC_GRID_MULTIPLE . . . 102
LOGIC_GRID_SIZE . . . 103
LOGIC_GRID_TOGGLE . . . 104
LOGIC_GRID_TYPE . . . 105
lrm_logfile . . . 106
LOWVOLTAGE_CLASS_PATTERN . . . 107
LOWVOLTAGE_THRESHOLD . . . 108

13
M Directives 109
MAKE_PAGE_TITLE_INVISIBLE 110
MAX_COL_IMPORT_TABLE 111
MAX_DRAWINGS 112
MAX_ERRORS 113
MAX_PINS_IN_NET 114
MAX_ROW_IMPORT_TABLE 115
Max_Search_Rows 116
Minimize_On_Add 117
MISO_NET_PATTERN 118
MISO_PIN_PATTERN 120
MONOCHROME_PRINTING_THRESHOLD 122
MOSI_NET_PATTERN 123
MOSI_PIN_PATTERN 125
MOVE 127
MULTI_FORMAT 128
MULTIPAGEBODERCREF 129

14
N Directives 131
NAVIGATION_OPTION 132
NAVLINKS_TEXT_FONT_BOLD 133
NAVLINKS_TEXT_FONT_COLOR 134
NAVLINKS_TEXT_FONT_ITALIC 135
NAVLINKS_TEXT_FONT_NAME 136
NAVLINKS_TEXT_FONT_SIZE 137
NAVLINKS_TEXT_FONT_UNDERLINE 138
NAVLINKS_TEXT_MARGIN 139
NET_NAME_CHARS 140
NET_NAME_LENGTH 141
NETGROUP_FILL_COLOR 142
NETGROUP_FILL_STYLE 143
NETGROUP_LINE_CAP_STYLE 144

NETGROUP_LINE_COLOR 145
NETGROUP_LINE_JOIN_STYLE 146
NETGROUP_LINE_STYLE 147
NETGROUP_LINE_WIDTH 148
NETGROUP_TEXT_FONT_BOLD 149
NETGROUP_TEXT_FONT_COLOR 150
NETGROUP_TEXT_FONT_ITALIC 151
NETGROUP_TEXT_FONT_NAME 152
NETGROUP_TEXT_FONT_SIZE 153
NETGROUP_TEXT_FONT_UNDERLINE 154
NETGROUP_TEXT_MARGIN 155
NETSPLIT_SUFFIX 156
NEW_PROPERTY_VISIBILITY 157
NO_CONNECT 158
NO_FEEDBACK 159
NON_GRAPHIC_MODE_FOR_CM 160
NOTE_COLOR 161
NUM_OLD_VERSIONS 162

15
O Directives 163

Text Object Types 164
<OBJECT>_FILL_STYLE 165
<OBJECT>_LINE_CAP_STYLE 166
<OBJECT>_LINE_COLOR 167
<OBJECT>_LINE_JOIN_STYLE 168
<OBJECT>_LINE_STYLE 169
<OBJECT>_LINE_WIDTH 171
<OBJECT>_TEXT_FONT_BOLD 172
<OBJECT>_TEXT_FONT_COLOR 173
<OBJECT>_TEXT_FONT_ITALIC 174
<OBJECT>_TEXT_FONT_NAME 175
<OBJECT>_TEXT_FONT_SIZE 176
<OBJECT>_TEXT_FONT_UNDERLINE 177
OCCPROP_COLOR 178

OFFPAGE 179
OFFPAGE_INPUT 180
OFFPAGE_IO 181
OFFPAGE_OUTPUT 182
old_versions_count 183
OMIT_CELL_FROM_CREF_PARTS 184
OMIT_CREFPARTS_HIERARCHY 185
OMIT_DOWN_HIERARCHY 186
OMIT_ZONE_INFO 187
Online_Mode 188
OPEN_ONLY_ACTIVE_TAB 189
OPTIMIZE 190
ORIENT_NET_NAME_DISPLAY 191
OUT_PORT_PIN_SIDE 192
OUTPORT 193
OUTPUT 194
OUTPUT_ASCII 195
OUTPUT_BINARY 196
OUTPUT_DEPENDENCY 197
OUTPUT_VERILOG 198
OUTPUT_VHDL 199
OVERWRITE_CONSTRAINTS 200

16
P Directives 201

Package_Class 202
Package_JedecType 203
Package_PinDelayUnit 204
PACKAGE_PROP 205
Package_RefDesPrefix 207
PACKAGED_FOLDER 208
PackagePin_AbsentChar 209
PackagePin_Property_ANALOG_Assert 210
PackagePin_Property_ANALOG_Connect 211
PackagePin_Property_ANALOG_Dir 212

PackagePin_Property_ANALOG_IO 213
PackagePin_Property_ANALOG_Load 214
PackagePin_Property_ANALOG_UnknownLoading 215
PackagePin_Property_BIDIR_Assert 216
PackagePin_Property_BIDIR_Connect 217
PackagePin_Property_BIDIR_Dir 218
PackagePin_Property_BIDIR_IO 219
PackagePin_Property_BIDIR_Load 220
PackagePin_Property_BIDIR_UnknownLoading 221
PackagePin_Property_GROUND_Assert 222
PackagePin_Property_GROUND_Connect 223
PackagePin_Property_GROUND_Dir 224
PackagePin_Property_GROUND_IO 225
PackagePin_Property_GROUND_Load 226
PackagePin_Property_GROUND_UnknownLoading 227
PackagePin_Property_INPUT_Assert 228
PackagePin_Property_INPUT_Connect 229
PackagePin_Property_INPUT_Dir 230
PackagePin_Property_INPUT_IO 231
PackagePin_Property_INPUT_Load 232
PackagePin_Property_INPUT_UnknownLoading 233
PackagePin_Property_NC_Assert 234
PackagePin_Property_NC_Connect 235
PackagePin_Property_NC_Dir 236
PackagePin_Property_NC_IO 237
PackagePin_Property_NC_Load 238
PackagePin_Property_NC_UnknownLoading 239
PackagePin_Property_OC_Assert 240
PackagePin_Property_OC_Connect 241
PackagePin_Property_OC_Dir 242
PackagePin_Property_OC_IO 243
PackagePin_Property_OC_Load 244
PackagePin_Property_OC_UnknownLoading 245
PackagePin_Property_OCBIDIR_Assert 246
PackagePin_Property_OCBIDIR_Connect 247
PackagePin_Property_OCBIDIR_Dir 248

PackagePin_Property_OCBIDIR_IO 249
PackagePin_Property_OCBIDIR_Load 250
PackagePin_Property_OCBIDIR_UnknownLoading 251
PackagePin_Property_OE_Assert 252
PackagePin_Property_OE_Connect 253
PackagePin_Property_OE_Dir 254
PackagePin_Property_OE_IO 255
PackagePin_Property_OE_Load 256
PackagePin_Property_OE_UnknownLoading 257
PackagePin_Property_OEBIDIR_Assert 258
PackagePin_Property_OEBIDIR_Connect 259
PackagePin_Property_OEBIDIR_Dir 260
PackagePin_Property_OEBIDIR_IO 261
PackagePin_Property_OEBIDIR_Load 262
PackagePin_Property_OEBIDIR_UnknownLoading 263
PackagePin_Property_OUTPUT_Assert 264
PackagePin_Property_OUTPUT_Connect 265
PackagePin_Property_OUTPUT_Dir 266
PackagePin_Property_OUTPUT_IO 267
PackagePin_Property_OUTPUT_Load 268
PackagePin_Property_OUTPUT_UnknownLoading 269
PackagePin_Property_POWER_Assert 270
PackagePin_Property_POWER_Connect 271
PackagePin_Property_POWER_Dir 272
PackagePin_Property_POWER_IO 273
PackagePin_Property_POWER_Load 274
PackagePin_Property_POWER_UnknownLoading 275
PackagePin_Property_UNSPEC_Assert 276
PackagePin_Property_UNSPEC_Connect 277
PackagePin_Property_UNSPEC_Dir 278
PackagePin_Property_UNSPEC_IO 279
PackagePin_Property_UNSPEC_Load 280
PackagePin_Property_UNSPEC_UnknownLoading 281
PAGE_BORDER 282
PAGE_DEFAULT_SIZE 283
PAGE_DOUBLE_WIDTH 284

PAGE_HEIGHT 285
PAGE_MARGIN_BOTTOM 286
PAGE_MARGIN_LEFT 287
PAGE_MARGIN_RIGHT 288
PAGE_MARGIN_TOP 289
PAGE_NAME_CASE 290
PAGE_NAME_PROP 291
PAGE_ORIENTATION 292
PAGE_SCALE 293
PAGE_SINGLE_WIDTH 294
PAGE_UNIT 295
PAGE_WIDTH 296
PAPER_ORIENTATION 297
PAPER_SIZE 298
PAPER_SOURCE 299
PART_TYPE_LENGTH 300
PASS_PROPERTY 302
PASTE_REPEATEDLY 303
PATHPROP_INVISIBLE 304
PDFA 305
PDFFont 306
PHYSICAL_FOLDER 307
PHYSICAL_PATH 308
PINALIAS_<PIN_TYPE> 309
PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH 311
PIN_ASSIGNMENT_DIALOG_SHOW_BLOCK_NET_NAME 312
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 313
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 314
PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT 315
PINNUMBER_ROTATION 316
PINNUMBER_SIZE 317
PINPROP_VISIBILITY 318
PinType 319
PLACEMENT_WITHIN_GROUP_USING_ORDER_IN_DESIGN 320
PLOT_COLOR 321
PLOT_DOUBLE_WIDTH 322

PLOT_FILE_NAME . . . 323
PLOT_FIT_TO_PAGE . . . 324
PLOT_FONT . . . 325
PLOT_SCALE . . . 326
PLOT_SCREEN . . . 327
PLOT_SINGLE_WIDTH . . . 328
PLOT_THICK_WIDTH . . . 329
PLOT_THIN_WIDTH . . . 330
PLOT_TO_FILE . . . 331
POWER_SYMBOLS . . . 333
POWERPROP_VIS . . . 335
Ppl_Only . . . 336
PPT . . . 337
PPT_BROWSER . . . 338
PPT_OPTIONSET_PATH . . . 339
PRESELECT_FLAG . . . 340
PRESELECT_FLAG ALLOW_USER_CPM . . . 341
PRESERVE_BYPASS . . . 342
PRESERVE_CONN . . . 343
PRESERVE_DAT . . . 344
PRESERVE_DAT 'ON'PRESERVE_PIN_TEXT_ROTATION . . . 345
PRESERVE_PINPAIR . . . 346
PRESERVE_PWRGRP . . . 347
PRESERVE_REFDES . . . 348
PRESERVE_TERMINATION . . . 349
PRESERVE_USERPROP . . . 350
PRESERVE_ZOOM_INFO . . . 351
PRINT_EXCLUSION . . . 352
PRINTLAYER . . . 353
PRINTLAYERENABLE . . . 354
PROCESS_PIN_SHORT_PROP . . . 355
Project_Ppl . . . 356
PROP_COLOR . . . 357
PROP_JUSTIFICATION . . . 358
PROP_OFFSET . . . 360
PROP_PLACEMENT_DEFAULT . . . 361

PROP_STACKING 363
PROP_VISIBILITY 365
PTF_MISMATCH_EXCLUDE_INJ_PROP 366
PTF_VIEW 367
PULLDOWN_MAX 368
PULLDOWN_MIN 369
PULLUP_MAX 370
PULLUP_MIN 371
PWR_PIN_NAME 372

17
R Directives 375

RAS_PIN_PATTERN 376
RAT_ON_REPLACE 377
REF_DES_LENGTH 378
REF_DES_PATTERN 379
REF_DES_PATTERN_FIX 382
REFDES_ALPHA_NUM 384
REFDES_PAGE_PADDING 386
REFDES_PREFIX_VISIBILITY_CHECK 387
REGENERATE_PHYSICAL_NET_NAME 388
REMOVE_FROM_STATE 389
rename_folders_on_copyproj 390
REPACKAGE 391
REPLACE_ACT_AS_MODIFY 392
REPLACE_PTF_PROPS 393
REPORT_COL_PAD 394
REPORT_COL_SEP 395
REPORT_CURRENCY_CHAR 396
REPORT_DIR 397
REPORT_FILES 399
REPORT_FONT_FACE 400
REPORT_FONT_SIZE 401
REPORT_FONT_STYLE 402
REPORT_FORMAT 403

REPORT_HEADER_SEP 404
REPORT_HIDE_LINENO 405
REPORT_ROW_SEP 406
REPORT_SORT_ORDER 407
REPORT_STRING_SEP 408
RESTRICTIVE_WIRE_MOVE 409
RETAIN_EXISTING_XNETS_AND_DIFFPAIRS 410
RETAIN_FONT_SETTINGS_ON_SYMBOL_ADD 411
RETAIN_HARD_LOCATION_ON_REPLACE 412
RETAIN_HARDLOCATION_ON_COPY 413
RETAIN_INSTANCE_PROP_ON_VERSION 414
RETAIN_LOCATION_ON_COPYALL 415
RETAIN_PATH_ON_REPLACE 416
RETAIN_PREVIOUS_HILITE 417
RETAIN_VERSION_ON_REPLACE 418
RETAIN_ZERONODE_NET 419
REUSE_REFDES 420
RIGHT_IN_OFFPAGE 422
RIGHT_IN_ROT 423
RIGHT_IO_OFFPAGE 424
RIGHT_IO_ROT 425
RIGHT_OUT_OFFPAGE 426
RIGHT_OUT_ROT 427
RUN_FEEDBACK 428
RUN_GENFEEDFORMAT 429
RUN_HIERWR_BEFORE_PXL 430
RUN_NETREV 431
RUN_PACKAGER 432

18
S Directives 433

SCALAR_PARENTHESIS 434
SAVEHIER_READONLY_MSG_AS_INFO 435
SAVE_WORKSPACE 436
SHOW_ALL_BLOCKS_FOR_IMPORT 437

SCH_POWER_GROUP_WINS_OVER_PPT 438
SCL_PIN_PATTERN 439
SDA_PIN_PATTERN 440
SHOW_PNN_SIGNAME 441
SHOW_POWER_SIGNAL_NAMES_FOR_NEW_CONNECTIONS 442
SHOW_PROPERTIES 443
SHOW_VARIANT_COLORS 444
SHOW_XNET_STATUS 445
SMART_PDF_LINE_WIDTH_FACTOR 446
SORT_ROWS_ON_ATTRFORM_LAUNCH 447
STICKY 448
SYMBOL_COLOR 449
SYMBOL_DOT_RADIUS 450
SYMBOL_GRID_MULTIPLE 451
SYMBOL_GRID_SIZE 452
SYMBOL_GRID_TOGGLE 453
SYMBOL_GRID_TYPE 454
SYNC_ON_PAGE_EDIT 455
SYNC_ON_STARTUP 456
SETCONCEPTFONT 457
SCH_XR_FORMAT_IN_REPORTS 458
SPREADSHEET 459
SD_SUFFIX_SEPARATOR 460
STATE_WINS_OVER_DESIGN 462
STATE_WINS_OVER_LAYOUT 464
STOP_PACKAGE_ON_SCHEMATIC_ERROR 465
STOP_PST_GEN_ON_PTF_MISMATCH 466
STRICT_PACKAGE_PROP 467
SUPPRESS 468
SD_PREFIX_SEPARATOR 469
SHOW_ANALYZE_DIALOG 471
SUPPORTLIBRARYDEFINEDDIFFPAIR 472
Symbol_Length 473
Symbol_OutLine 474
Symbol_PinShape_Dot_Size 475
Symbol_Pintext_LeftRight_XOffset 476

Symbol_Pintext_LeftRight_YOffset 477
Symbol_Pintext_TopBottom_XOffset 478
Symbol_Pintext_TopBottom_YOffset 479
Symbol_PN_LeftRight_XOffset 480
Symbol_PN_LeftRight_YOffset 481
Symbol_PN_TopBottom_XOffset 482
Symbol_PN_TopBottom_YOffset 483
Symbol_SymSheetSize 484
Symbol_Units 485
Symbol_Width 486
Search_Attr_Name 487
Search_Attr_Type 488
Search_Result_Type 489
Search_Tab_Order 490
Search_Tolerance 491
Search_Tree_Order 492
Show_Cell 493
Show_Library 494
Single_Click_Details 495
sync_properties 496
SAVE_TCL_IN_PROJECT 497
SDA_CAPTURE_SPECIAL_LIB 498
STUB_LENGTH 499
SHOW_UNCONNECTED_PINS 500
SHOW_NET_HOTSPOTS 501
SHOW_NET_NAME_DIALOG 502
SHOW_PART_MANAGER_ON_START 503

19
T Directives 505

TAP_SYMBOL 506
TESTPAD_FOOTPRINT_NAME_PATTERN 507
TEXT_EDITOR 508
TEXT_JUSTIFICATION 510
TEXT_SIZE 511

TOC_AUTO_SAVE 512
TOC_DISPLAY_SHEET_RANGE 513
TOC_ROW_SPACING_MULTIPLIER 514
TOC_PAGE_DEFAULT_SIZE 515
TABLE_ALTERNATE_FILL_COLOR 516
TABLE_FILL_COLOR 517
TABLE_FILL_STYLE 518
TABLE_HEADER_FILL_COLOR 519
TABLE_LINE_CAP_STYLE 520
TABLE_LINE_COLOR 521
TABLE_LINE_JOIN_STYLE 522
TABLE_LINE_STYLE 523
TABLE_LINE_WIDTH 524
TABLE_TEXT_FONT_BOLD 525
TABLE_TEXT_FONT_COLOR 526
TABLE_TEXT_FONT_ITALIC 527
TABLE_TEXT_FONT_NAME 528
TABLE_TEXT_FONT_SIZE 529
TABLE_TEXT_FONT_UNDERLINE 530
TABLE_TEXT_MARGIN 531
TEXT_FONT_NAME 532
TEXT_FONT_SIZE 533
TEXT_FONT_ITALIC 534
TEXT_FONT_BOLD 535
TEXT_FONT_UNDERLINE 536
TEXT_FONT_COLOR 537
TEXT_MARGIN 538
TEXT_WORD_WRAP 539

20
U Directives 541

UNNAMED_NET_GEN 542
UNNAMED_NET_USING_FULL_PINNAME 543
UNIQUE_FEATURE 544
USE_VECTOR_NOT_ATION 545

USE_LIBRARY_PPT 546
USE_SUBDESIGN 547
USE_OFFPAGE 548
USE_POWER_SYMBOLS 549
UPPERCASE_SIGNAL_NAMES 550

21
V Directives 551

VAR_OVERLAY_PROPS_VISIBLE 552
VAR_COMP_BOM_PROPS 554
VAR_REPLACE_BY_PROP 556
VAR_REPLACE_PROP 557
VISIBLE 558
VIEW_PCB 559
Viewers 560
VARIANT_DNI_CROSS 561
VARIANT_DNI_CROSS_COLOR 562
VARIANT_DNI_CROSS_LINE_WIDTH 563
VARIANT_EDITOR_DISPLAY_PROP_NAME 564
VARIANT_INST_FILL_COLOR 565
VARIANT_INST_FILL_STYLE 566
VARIANT_INST_LINE_CAP_STYLE 567
VARIANT_INST_LINE_COLOR 568
VARIANT_INST_LINE_JOIN_STYLE 569
VARIANT_INST_LINE_STYLE 570
VARIANT_INST_LINE_WIDTH 571
VARIANT_INST_ITEM_OPACITY 572
VARIANT_PROPERTY_ITEM_OPACITY 573
VARIANT_PROPERTY_TEXT_FONT_BOLD 574
VARIANT_PROPERTY_TEXT_FONT_COLOR 575
VARIANT_PROPERTY_TEXT_FONT_ITALIC 576
VARIANT_PROPERTY_TEXT_FONT_NAME 577
VARIANT_PROPERTY_TEXT_FONT_SIZE 578
VARIANT_PROPERTY_TEXT_FONT_UNDERLINE 579

22
W Directives 581
WARNING_MESSAGES 582
WINDOWSMODE_FLAG 583
WINDOWSMODE_FLAG ALLOW_USER_CPM 584
WIRE 585
WIRE_COLOR 586
WM_COLOR 587
WM_FONT 588
WM_FONTSIZE 589
WM_HORIZONTAL_ALIGNMENT 590
WM_OPACITY 591
WM_ROTATION 592
WM_SCALE 594
WM_SHOW_ONPRINT 595
WM_SHOW_ONSCREEN 596
WM_TEXT 597
WM_TYPE 598
WM_VERTICAL_ALIGNMENT 599
WARNINGS 600

23
X Directives 601
XNET_ABSENT_COLOR 602
XNET_EXISTS_COLOR 603

1

Introduction to CPM Files

The SPB design capture and library creation and maintenance tools— Allegro Design Entry HDL, System Connectivity Manager, PCB Librarian, and Allegro Design Workbench— manage all the information about a project through project files (`.cpm`). This information includes default values for tools, libraries, physical part table files, log files, and property files.

There are four types of project (cpm) files:

- Local Project Files
- User-Specific Project Settings File
- Site Project File
- Installation Project File

Local Project Files

When you create a new project, the Project Manager creates a project file called <*projectname*>.cpm in the project directory. Each project has one project file. The <*projectname*>.cpm file contains all the setup information that you specified for your project. It has the following:

- The name of the top-level design and the library in which it is located
- The list of project libraries
- The physical part tables selection
- Changes to the default view names
- The name and location of the text editor for editing text files from Cadence tools
- The name and location of the property file
- The name and location of the log file

- The name and location of the application temp directory, which is the directory in which applications such as Design Entry HDL store temporary files.
- Setup directives for individual tools such as Design Entry HDL, Packager-XL, and the Project Manager.
- Directives for customizing the Project Manager (a customized Tools menu or customized flows).

The default setup information is maintained in an installation project file (`cds.cpm`) shipped by Cadence. The defaults in the `cds.cpm` file apply to all your projects. If you want to change these defaults, create a site project file (`site.cpm`) for your site.

When you open a project, the Project Manager gets the setup directives you specified for that project from the `<projectname>.cpm` file and the defaults for the others from the `site.cpm` and `cds.cpm` files. Your setup directives always have precedence over the `site.cpm` directives, which in turn have precedence over the `cds.cpm` directives.

You can view the project settings for a project with the *View – Project Settings* command.

Project File (.cpm) example

```
( Machine generated file created by SPI )
( Last modified was 11:38:31 Thursday, October 09, 1997 )
( NOTE: Do not modify the contents of this file. If this is regenerated by )
( SPI, your modifications will be overwritten. )

START_GLOBAL
use library_ppt 'ON'
design_name 'poa'
design_library 'poa'
library 'poa' 'standard' 'pic' 'poa_lib' 'element'
temp_dir 'temp'
cpm_version '@'
session_name 'ProjectMgr12919'
cdsprop_file ''
ppt './ptf/poa.ppt'
EXCLUDE_PPT
INCLUDE_PPT
END_GLOBAL

START_PKGRXL
state_wins_over_design 'ALL'
END_PKGRXL
```

User-Specific Project Settings File

You define user-specific project settings in the user.cpm file. The project settings in this file apply to all the projects that a user opens. The user.cpm file is located at $HOME/cdssetup/projmgr/, provided the environment variable $HOME is set.

Site Project File

You can create a site project file, called `site.cpm`, in the `<your_inst_dir>/share/local/cdssetup/projmgr` directory if you want to specify default setup options for all the projects at your site. The directives in this file have precedence over the installation project file (`cds.cpm`) and the local project file (`<projectname>.cpm`) has precedence over the `site.cpm` file.

You can customize the default settings for all your projects by creating the `site.cpm` file. To create a `site.cpm` file, either use a copy of an existing project file, or create a dummy project and use its project file to define your site settings.

Creating Site Project File for All Projects

To create a site project file for all the projects at your site,

1. Choose *Tools – Setup*.
2. In each tab of the *Project Setup* dialog box, specify the default setup information you want for all projects. For information about the setup options, click the *Help* button in the dialog box.
3. Click *Apply* to save your changes.
4. Close *Project Setup* by clicking *OK*.
5. Choose *File – Export*.
6. In the *Export Project* dialog box,
   - Type `site.cpm` in the *File Name* box.
   - In the *Folders* list, select *`<your_inst_dir>/share/local/cdssetup/projmgr`*, where *`<your_inst_dir>`* is the directory in which you have installed Cadence tools.
   - Ensure that the *Save File as Type* box displays Project Files (`*.cpm`).
   - Ensure that the *Full Settings* option is not selected.

7. Click *OK* in the *Export Project Setup* dialog box.

Custom Site Environment

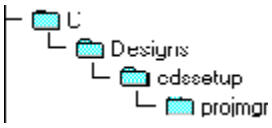
The site.cpm File

If you do not place the `site.cpm` file in the *`<your_inst_dir>/share/local/cdssetup/projmgr`* directory, you must set a `CDS_SITE = location` environment variable that specifies the location of the site project file. The site location must have the following directory structure:

```
cdssetup/projmgr/site.cpm
```

For example, if you want to set your `CDS_SITE = C:\Designs`, you must create the following directory structure and place the `site.cpm` file in the `projmgr` directory:

.



The concepthdl.scr File

If you have set the `CDS_SITE` environment variable to another location, such as `/hm/common/`, you need to ensure that the `concepthdl.scr` file is at `/hm/common/cdssetup/concept/`. Otherwise, backannotation from Variant Editor will not work.

Flows

If you have any custom Project Manager flows, maintain them at *`$CDS_SITE/cdssetup/projmgr/flows`* using the same directory structure as at *`<your_inst_dir>/share/cdssetup/projmgr/flows/`*.

Other Customized Files

If you have customized any of the following files and want the changed version to be available for all projects at your site, copy them into the location mentioned in the table below. This will ensure that the customized information is available even when you install a newer version of Cadence SPB software.

Table 1-1 Customizations for Project Manager and Point Tools

Files and Descriptions	Location
`cds.lib` (lists libraries used in the project)	Place at `$CDS_SITE/cdssetup`.
`bom.callouts` (mechanical parts to be added in the BOM reports)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cdsinfo.tag` (project-specific information, including the name of the data management system, if any, used in the project)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cdsprop.paf` (information about properties)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cdsprop.tmf` (information about text macros)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`cjedectype.txt` (compatible JEDEC types in the Variant Editor tool)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`propflow.txt` (the default property flow setup in Packager Setup)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`template.bom` (the default template for BOM reports)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`xilfam.dat` (mapping information between a Xilinx family and a specific library and architecture)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.

`xmodules.dat` (modules that have to be excluded for cross-referencing and plotting)	Copy from `<your_inst_dir>/share/cdssetup/` to `$CDS_SITE/cdssetup/`.
`concepthdl_key.txt` (Design Entry HDL shortcut keys)	Copy from `<your_inst_dir>/share/cdssetup/concept/` to `$CDS_SITE/cdssetup/concept/`.
`concepthdl_menu.txt` (Design Entry HDL menus)	Copy from `<your_inst_dir>/share/cdssetup/concept/` to `$CDS_SITE/cdssetup/concept/`.
`template.tsg` (information related to the graphical attributes of the symbols and additional pin and symbol properties)	Copy from `<your_inst_dir>/share/cdssetup/concept/genview/` to `$CDS_SITE/cdssetup//concept/genview/`.
`cref.dat` (template options of CRefer)	Copy from `<your_inst_dir>/share/cdssetup/creferhdl/` to `$CDS_SITE/cdssetup/creferhdl/`.

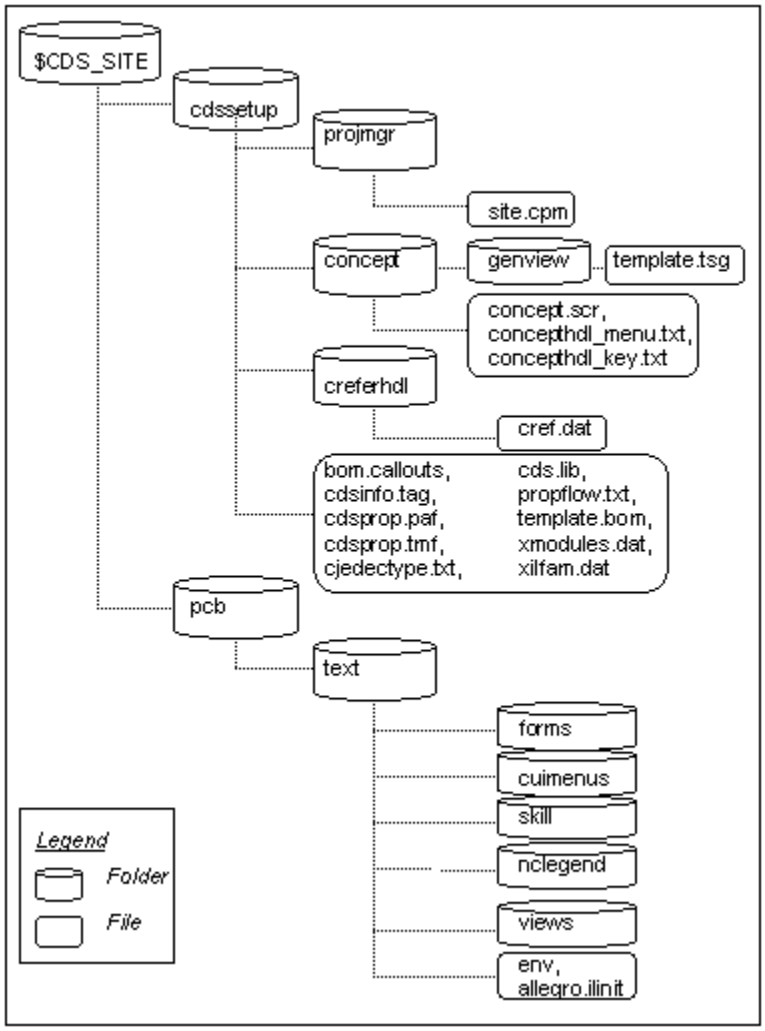
Note: The `cdsprop.txt` file need not be copied as you should not modify this file.

Table 1-2 Customizations for Allegro PCB Editor

Files and Descriptions	Location
`.ilinit` file (PCB Editor SKILL initialization file)	Copy file from `<your_inst_dir>/share/pcb/text/` to `$ALLEGRO_SITE/skill/` where `ALLEGRO_SITE = <your_inst_dir>/share/local/pcb`
`allegro.men` (updated PCB Editor menus)	Copy file(s) from `<your_inst_dir>/share/pcb/text/cuimenus` to `$ALLEGRO_SITE/menus/` where `ALLEGRO_SITE = <your_inst_dir>/share/local/pcb`

`env` (paths to PCB Editor libraries and other site settings for PCB Editor)	Copy file(s) from `<your_inst_dir>/pcb/text/` to `ALLEGRO_SITE/pcb/`
`forms` folder (all forms called from Allegro SKILL code)	Copy folder from `<your_inst_dir>/share/pcb/text/` to `ALLEGRO_SITE/pcb/`
`nclegend` folder (`.dlt` files) (PCB Editor templates from NCDRIII legend)	Copy files from `<your_inst_dir>/share/pcb/text/nclegend` to `ALLEGRO_SITE/pcb/nclegend`
`skill` folder (custom Allegro SKILL code)	Place the folder at `$ALLEGRO_SITE/`
`views` files (Allegro extract command files)	Copy from `<your_inst_dir>/share/pcb/text/` to `$ALLEGRO_SITE/views/` where `ALLEGRO_SITE = <your_inst_dir>/share/local/pcb`

The directory structure for a custom site environment must be as depicted below:



Installation Project File

The installation project file, called `cds.cpm`, is shipped by Cadence and is in the *`<your_inst_dir>/share/cdssetup/projmgr`* directory. The `cds.cpm` file contains default setup directives for all projects and tools. The Project Manager obtains defaults from this file for setup options that are not defined in the `<projectname>.cpm` or `site.cpm` files. Do not modify this file. If you want to change the defaults for a set of projects, create a site project file (`site.cpm`).

The setup directives you specify (that is, the directives in the `<projectname>.cpm` file) always have precedence over the `site.cpm` directives, which in turn have precedence over the `cds.cpm` directives. When you open a project, the Project Manager gets the setup directives you specified for that project from the `<projectname>.cpm` file and the default values for the others from the `site.cpm` file. If they are not defined in the `site.cpm` file either, the Project Manager obtains the default values from the `cds.cpm` file.

Locking Directives

Locking project (cpm) file directives provide a mechanism by which you can control user access and modification permissions on project settings. You can also configure settings that are reflected in all the projects you open, irrespective of the settings in the project's `.cpm` file.

Project Settings at Different Levels

Project settings are configured in the installation project file (`cds.cpm`), the site project file (`site.cpm`), and the local project file (`<project>.cpm`).

However, you might want to define user-specific settings, which you can customize according to your needs and retain the same settings for any project you open, irrespective of the local project settings. Some examples of user-specific settings include: default printer, text editor, and panning.

Directive locking is allowed at any of the four levels, including the `user.cpm` level. Locking directives at user level provides control over the list of directives that you can configure at user level and will reflect in all the projects, irrespective of the project settings. This can be done by defining user-specific settings in the `user.cpm` file. If the environment variable `$HOME` is set, the ***user.cpm*** file is located at *`$HOME/cdssetup/projmgr/`*.

Locking a Directive

A locked directive is defined with the keyword `LOCK` in the `.cpm` file. Locking implies that the directive is locked for all levels down from the level at which it is locked.

For example, locking a directive at `project.cpm` implies that the directive will be honored at the `project.cpm` level if the directive is in `project.cpm`. If it is not in `project.cpm`, the directive will be honored from `site.cpm` or `cds.cpm` as the case may be. However, the directive, if in `user.cpm`, will not be honored.

Example

To lock the PINNUMBER_SIZE directive, the following two sections are required in the .cpm file:

The PINNUMBER_SIZE directive will need to be in the following section:

```
START_CONCEPTHDL
PINNUMBER_SIZE 0.090
END_CONCEPTHDL
```

And it will need to be in this section too in the cpm file:

```
START_CONCEPTHDL_CONTROL_SETTINGS
PINNUMBER_SIZE   LOCK
END_CONCEPTHDL_CONTROL_SETTINGS
```

When a project is loaded, the directive in the user.cpm file will be honored only if the installation project file (cds.cpm) or the site-level cpm file (site.cpm) allows the directive to be read and set at the user level.

You need to specifically allow user-level settings for a given directive in the install or site-level cpm files with the ALLOW_USER_CPM keyword as illustrated:

```
START_CONCEPTHDL_CONTROL_SETTINGS
PINNUMBER_SIZE ALLOW_USER_CPM
END_CONCEPTHDL_CONTROL_SETTINGS
```

Locking Support for Directives at Different Levels

The locking mechanism for directives allows better control over configuring settings, such as part table settings, PXL property settings, CHECK command rules, grid settings, and so on. You can lock any directive in the cpm files at different levels. The directive locking feature uses the following precedence to check the locking status of directives:

- $CDSROOT: The CDSROOT project file (cds.cpm) is the first cpm file to be read.
- $CDS_SITE: The next cpm file in the load process is from CDS_SITE (site.cpm). You can lock the directives in this file also, but this setting overrides the directive settings in the user's HOME account and the local <project>.cpm file.
- Project: The <project>.cpm file is the next file to be read.
- $HOME: After the CDSROOT and CDS_SITE project files, the HOME cpm file (user.cpm) is loaded. Directives in user.cpm will be honored only when the ALLOW_USER_CPM setting is provided at the install or CDS_SITE level cpm for those directives.

The `$HOME` settings will be honored only when the directives in `user.cpm` are not locked at the install, site, or project levels. Locks found in the HOME account can be used to keep local project settings from masking your preferences. This is how preferences are honored in each project that you open.

Note: You cannot edit the directives locked in the `CDSROOT`, `CDS_SITE`, or `HOME` areas in a local project environment.

Reading Settings from CPM Files at Different Levels

The following table describes how settings in the cpm files at different levels impact your project:

CPM File	Description
`user.cpm`	Contains the user-level settings. Directives defined in the `user.cpm` file are honored if: ■ They are not locked at the `<project>.cpm`, `site.cpm`, or `cds.cpm` levels ■ They are specified in the `ALLOW_USER_CPM` keyword in `site.cpm` or `cds.cpm` Modified directives honored in `user.cpm` are written to `user.cpm` and not to `<project>.cpm`.

CPM File	Description
`<project>.cpm`	Contains settings that are local to a project. Directives defined in the `<project>.cpm` file are honored if: ■ They are not honored in `user.cpm`. ■ They are not locked in `site.cpm` or `cds.cpm`. Changes to a directive locked at the `<project>.cpm` level are allowed and the new value is written in the `<project>.cpm`. The modified directive in the `<project>.cpm` file does not have the `LOCK` keyword associated with it.
`site.cpm`	Contains the site-level settings. Directives defined in the `site.cpm` file are honored if: ■ They are not honored in `user.cpm` or `<project>.cpm`. ■ They are not locked in `cds.cpm`. Changes to the directives are not honored if they are locked in `site.cpm` itself. Otherwise, all modifications to the directives are written to the `<project>.cpm` file. A modified directive in the `<project>.cpm` file does not have the `LOCK` keyword associated with it.

CPM File	Description
cds.cpm	Contains the install-level settings. Directives defined in the cds.cpm file are honored if they are not honored in user.cpm, <project>.cpm, or site.cpm. Modifications to the directives are not honored if they are locked in the cds.cpm itself. Otherwise, all the modifications to the directives are written to the <project>.cpm file. The modified directive in the <project>.cpm file does not have the LOCK keyword associated with it.

Example

Consider the following example of the PINNUMBER_SIZE directive defined at the four levels of cpm files:

CPM File	Directive Definition
user.cpm	START_CONCEPTHDL ... PINNUMBER_SIZE 0.075 ... END_CONCEPTHDL ...
<project.cpm>	START_CONCEPTHDL ... PINNUMBER_SIZE 0.072 ... END_CONCEPTHDL

CPM File	Directive Definition

`site.cpm`	`START_CONCEPTHDL`
	`..`
	`PINNUMBER_SIZE 0.090`
	`...`
	`END_CONCEPTHDL`
	`...`
	`START_CONCEPTHDL_CONTROL_SETTINGS`
	`PINNUMBER_SIZE LOCK`
	`END_CONCEPTHDL_CONTROL_SETTINGS`

cds.cpm	`START_CONCEPTHDL`
	`...`
	`PINNUMBER_SIZE 0.082`
	`...`
	`END_CONCEPTHDL`

Here, the directive `PINNUMBER_SIZE` with a value of *`0.090`* (in `site.cpm`) will be honored. Modification to this value is not allowed.

2

Allegro Design Entry HDL CPM Directives

This chapter lists the CPM directives for Design Entry HDL. These directives are specified in the `START_CONCEPTHDL...END_CONCEPTHDL` section of the project's `.cpm` file.

ALLOW_EMPTY_LOCK_FILE
ALLOW_FOOTPRINT_COMPATIBILITY_CHECK
ALLOW_PAGE_LOCKING
ALLOW_PINTEXT_SWAP
ALLOW_POWER_PINS
ALLOW_PROP_ILLEGAL_OBJECT
ALLOW_PROP_ON_PROP
ALLOW_PROPERTY_LOCKING
ALLOW_PTFVALUE_INITIAL_BLANKS
ALLOWED_ALTERNATE_PART_PROP
ANNOTATE_PART_NAME
ARC_COLOR
ASK_RENAME_SIGNAME_OPTION
ATTRIBUTES_DIR
AUTO_UPDATE_DEFAULT_MODELS
AUTODOT
AUTOHEAVY
AUTOPAN
AUTOPATH
AUTOROUTE
BACKGROUND_COLOR
BBOX_SHOWN_ON_FONT_MOVE
BLOCK_TITLE_TOP
BLOCK_PIN_SHAPE_MINIMUMPINSPACING
CATPATH
CAPSLOCK
CDS_<CATEGORY_NAME>_FONTCOLOR
CDS_<CATEGORY_NAME>_FONT
CDS_<CATEGORY_NAME>_FONTSIZE
CDS_<CATEGORY_NAME>_FONTEFFECTS
CDS_ENABLE_FONTS
CDS_<CATEGORY_NAME>_FONTSTYLE
CHECK_GLOBAL_LOCAL_SHORT
CHECK_ARCS_AT_SAME_LOCATION
CHECK_HIDDEN_WIRES
CHECK_GRID_ON_TAP
CHECK_MISSING_PINS

CHECK_IMAGE_OVERLAP_OBJECT
CHECK_MOVE_SHORT
CHECK_NONSYNCPROPS
CHECK_PACK_SEC_TYPE_PROPS
CHECK_PAGE_NUMBER_SYNCH
CHECK_PINS_AT_ORIGIN
CHECK_PROP_PLACE_HOLDERS
CHECK_SIGNAL_NAMES
CHECK_SYMBOL_SIGNAL_NAMES
CHECK_TWO_WIRES_AT_PINS
CHECK_VOLTAGE_ON_HDL
CNAME_CELL
CNAME_LIB
COMBINED_MARKERS_ON_HIERWRITE
CONFIRM_WRITE
CTRLRMB_CONTEXTMENU
CUSTOM_CDSLIB_SEARCH
DEFAULT_PAGE_BORDER_NAME
DELETE_ASCII
DELETE_UNATTACHED_INVISIBLE_PROPS
DISABLE_SMARTPDF_SMART_FEATURES
DOC_GRID_SIZE
DOC_GRID_TYPE
DONT_FORCE_ORIGIN_ONGRID
DONT_SHOW_CM_DLG
DOTS
EDIT_PHYSICAL_SPACING_CONSTRAINTS
CHECK_MULTIPLE_IBD_PROPS
CHECK_ON_WRITE
CHECK_PACK_SYNC_ON_IMPORT
CHECK_PIN_WIRE_DIST_THRESH
CHECK_PINS_NEAR_WIRE_ENDPT
CHECK_SHORTED_PINS
CHECK_SYMBOL_PLACE_HOLDERS
CHECK_SYMBOLS_AT_SAME_LOCATION
CHECK_UNCONNECTED_WIRES
CLICK_TO_TYPE
CNAME_DEPTH
CNAME_VIEW
COMPONENT_BROWSER
CTRLLMB_DRAGSELECT
CURSOR_SHAPES
DATATIPS_PROP_NAME
DEFAULT_PAGE_BORDER_VERSION
DELETE_BINARY
DESIGN_TYPE
DOC_GRID_MULTIPLE
DOC_GRID_TOGGLE
DONT_ALLOW_UPREV
DONT_SET_OFFSET_PINNUMBER
DOT_COLOR
DRAWING_BROWSER
ENABLE_ALPHANUMERIC_CUSTOMKEYS
ERROR_MESSAGES

ERR_OK_NET_ONE_PIN_MULTI_NODES
FLATTEN_BLOCK_ON_ADD
FATAL_MESSAGES
GENERATE_TDD_NETLIST
GENERATE_SCH_METADATA
HIDE_INSTANCE_NAME
HIDE_HIERARCHY_PAGES
HIGHLIGHT_COLOR
HIDE_SHEET_NUMBER
HPF_BATCH
HONOR_PPTOPTION_ON_ADD_OR_REPLACE
HPF_FONT
HPF_BUS_SCALEFACTOR
HPF_PLOT_PAGESIZE
HPF_PAGESIZE
HPF_SCALEFACTOR
HPF_PLOTTER
HPF_SPEC_PLOT_PAGESIZE
HPF_SCALETYPE
HYPERLINKS
HPF_WIRE_SCALEFACTOR
IMAGE* Directives
IGNORE_BUNDLED_CONSTRAINTS_ERROR_ON_ZERO_NODE_NETS
IMAGE_BACKGROUND_COLOR
IMAGE_ARC_COLOR
IMAGE_DOT_COLOR
IMAGE_DEFAULT_DPI
IMAGE_OCCPROP_COLOR
IMAGE_NOTE_COLOR
IMAGE_SYMBOL_COLOR
IMAGE_PROP_COLOR
INFORMATIONAL_MESSAGES
IMAGE_WIRE_COLOR
LIBRARY_BROWSER
INPUT_SCRIPT
LOCK_FILE_PERM
LIST_VALID_VERSIONS_METADATA
LOGIC_GRID_MULTIPLE
LOGIC_DOT_RADIUS
LOGIC_GRID_TOGGLE
LOGIC_GRID_SIZE
MAX_DRAWINGS
LOGIC_GRID_TYPE
MAX_PINS_IN_NET
MOVE
MULTI_FORMAT
NAVIGATION_OPTION
NOTE_COLOR
OCCPROP_COLOR
OUTPUT_ASCII
OUTPUT_BINARY
OUTPUT_DEPENDENCY
OUTPUT_VERILOG
OUTPUT_VHDL

PAGE_NAME_CASE
PAGE_NAME_PROP
PAPER_ORIENTATION
PAPER_SIZE
PAPER_SOURCE
PATHPROP_INVISIBLE
PINNUMBER_ROTATION
PINNUMBER_SIZE
PINPROP_VISIBILITY
PLOT_COLOR
PLOT_DOUBLE_WIDTH
PLOT_FILE_NAME
PLOT_FIT_TO_PAGE
PLOT_FONT
PLOT_SCALE
PLOT_SCREEN
PLOT_SINGLE_WIDTH
PLOT_THICK_WIDTH
PLOT_THIN_WIDTH
PLOT_TO_FILE
POWERPROP_VIS
PPT_BROWSER
PPT_OPTIONSET_PATH
PRESELECT_FLAG
PRESELECT_FLAG ALLOW_USER_CPM
PRESERVE_DAT 'ON'PRESERVE_PIN_TEXT_ROTATION
PRESERVE_ZOOM_INFO
PROP_COLOR
PROP_JUSTIFICATION
PROP_OFFSET
PROP_PLACEMENT_DEFAULT
PROP_STACKING
PROP_VISIBILITY
REPLACE_ACT_AS_MODIFY
REPLACE_PTF_PROPS
RETAIN_EXISTING_XNETS_AND_DIFF PAIRS
RETAIN_FONT_SETTINGS_ON_SYMBOL_ADD
RETAIN_HARDLOCATION_ON_COPY
RETAIN_HARD_LOCATION_ON_REPLACE
RETAIN_LOCATION_ON_COPYALL
RETAIN_INSTANCE_PROP_ON_VERSION
RETAIN_PREVIOUS_HILITE
RETAIN_PATH_ON_REPLACE
SAVE_WORKSPACE
RETAIN_VERSION_ON_REPLACE
SCH_POWER_GROUP_WINS_OVER_PPT
SAVEHIER_READONLY_MSG_AS_INFO
SHOW_PROPERTIES
SHOW_PNN_SIGNAME
SORT_ROWS_ON_ATTRFORM_LAUNCH

SHOW_VARIANT_COLORS
SHOW_XNET_STATUS
STICKY
SYMBOL_COLOR
SYMBOL_DOT_RADIUS
SYMBOL_GRID_MULTIPLE
SYMBOL_GRID_SIZE
SYMBOL_GRID_TOGGLE
SYMBOL_GRID_TYPE
SYNC_ON_PAGE_EDIT
SYNC_ON_STARTUP
TAP_SYMBOL
TEXT_EDITOR
TEXT_JUSTIFICATION
TEXT_SIZE
TOC_DISPLAY_SHEET_RANGE
TOC_ROW_SPACING_MULTIPLIER
UNNAMED_NET_GEN
UNNAMED_NET_USING_FULL_PINNAME
VAR_OVERLAY_PROPS_VISIBLE
WARNING_MESSAGES
WINDOWSMODE_FLAG
WINDOWSMODE_FLAG ALLOW_USER_CPM
WIRE
WIRE_COLOR
XNET_ABSENT_COLOR
XNET_EXISTS_COLOR

Design Synchronization CPM Directives

This section lists the CPM directives for the Design Synchronization toolset, which includes the following utilities:

- Packager Utilities
- Design Differences
- Design Association
- Netrev
- Genfeedformat

BACKANNOTATE_FEEDBACK
BACKANNOTATE_FORWARD
CREATE_USER_PROP
ETCH_REMOVAL
IGNORE_FIXED
LAST_BOARD_FILE
RUN_FEEDBACK
RUN_GENFEEDFORMAT
RUN_HIERWR_BEFORE_PXL
RUN_NETREV
RUN_PACKAGER

Design Entry HDL Utilities CPM Directives

This chapter lists the CPM directives for Design Entry HDL tools, such as Allegro Design Publisher, Cross-Referencer, and BOM-HDL. These directives are specified in the setup dialog and are stored in the `.cpm` file.

Allegro FPGA System Planner CPM Directives

NON_GRAPHIC_MODE_FOR_CM

Allegro Design Publisher CPM Directives

ATTRIBUTEDATATOOLFORM
ATTRIBUTEFILTER
ATTRIBUTEJAVAFORM
AUTOSIZE_MARGIN
BLACKANDWHITEPDF
CURRENTPDFVIEWER
CURRENTPDFVIEWERPATH
EXPORT
MULTIPAGEBODERCREF
PAGE_DOUBLE_WIDTH
PAGE_HEIGHT
PAGE_MARGIN_BOTTOM
PAGE_MARGIN_LEFT
PAGE_MARGIN_RIGHT
PAGE_MARGIN_TOP
PAGE_ORIENTATION
PAGE_SCALE
PAGE_SINGLE_WIDTH
PAGE_UNIT
PAGE_WIDTH
PDFA
PDFFont
PRINTLAYER
PRINTLAYERENABLE
SETCONCEPTFONT
VISIBLE
WM_COLOR
WM_FONT
WM_FONTSIZE
WM_HORIZONTAL_ALIGNMENT
WM_OPACITY
WM_ROTATION
WM_SCALE
WM_SHOW_ONPRINT
WM_SHOW_ONSCREEN
WM_TEXT
WM_TYPE
WM_VERTICAL_ALIGNMENT

Rules Checker Directives

ENABLE_FONT_BASED_BBOX_COMPUTATION

CREFERHDL CPM Directives

BASENET_OMIT_SYNONYM
ERROR_NO_CSB_FILES
FORMAT_CREF_REPORTS
GENERATE_FLATTENED_SCHEMATIC
GENERATE_SEPARATE_CELL
GENERATE_XR_FOR_ALL_NETS
MAKE_PAGE_TITLE_INVISIBLE
OMIT_CELL_FROM_CREF_PARTS
OMIT_CREFPARTS_HIERARCHY
OMIT_DOWN_HIERARCHY
OMIT_ZONE_INFO
SCH_XR_FORMAT_IN_REPORTS

BOM-HDL Directives

LAST_OUTPUT_FILE
LAST_TEMPLATE_FILE
LAST_VARIANT_FILE
SPREADSHEET
UNIQUE_FEATURE
VAR_COMP_BOM_PROPS

Design Variance CPM Directives

ALLOW_INCOMPATIBLE_JEDEC_TYPE
LAST_VARIANT_FILE

Archiver CPM Directives

EXCLUDE_FILE_PATH
EXCLUDE_VIEW

Packager-XL Directives

This chapter lists the Packager-XL directives. These directives are specified in the setup form and are stored in the project file.

Caution

You should not edit the project file yourself. Use Packager Setup to change Packager-XL directives.

ANNOTATE
B2F_OVERWRITE_CONSTRAINTS
COMP_DEF_PROP
COMP_INST_PROP
DEFAULT_PHYS_DES_PREFIX
ELECTRICAL_CONSTRAINTS
ERROR_ON_PARTIAL_INSTANTIATION_OF_HSS
EXCLUDE_PPT
F2B_OVERWRITE_CONSTRAINTS
FEEDBACK
FILTER_CONFLICTING_PROP
FILTER_PROPERTY
FORCE_PTF_ENTRY
FORCE_SUBDESIGN
GEN_SUBDESIGN
HARD_LOC_SEC
IGNORE_VAR_STATUS_COL
INCLUDE_PPT
MAX_ERRORS
NET_NAME_CHARS
NET_NAME_LENGTH
NO_FEEDBACK
NUM_OLD_VERSIONS
OPTIMIZE
OUTPUT
PACKAGE_PROP
PART_TYPE_LENGTH
PASS_PROPERTY
PHYSICAL_PATH
PPT
PROCESS_PIN_SHORT_PROP
PTF_MISMATCH_EXCLUDE_INJ_PROP
PTF_VIEW
REF_DES_LENGTH
REF_DES_PATTERN
REF_DES_PATTERN_FIX

REGENERATE_PHYSICAL_NET_NAME
REMOVE_FROM_STATE
REPACKAGE
REUSE_REFDES
SD_SUFFIX_SEPARATOR
STATE_WINS_OVER_DESIGN
STATE_WINS_OVER_LAYOUT
STOP_PACKAGE_ON_SCHEMATIC_ERROR
STOP_PST_GEN_ON_PTF_MISMATCH
STRICT_PACKAGE_PROP
SUPPRESS
USE_LIBRARY_PPT
USE_SUBDESIGN
USE_VECTOR_NOTATION
VIEW_PCB
WARNINGS

System Connectivity Manager CPM Directives

This section lists the CPM directives for System Connectivity Manager. These directives are specified in the `START_DESIGNSTUDIO` section of a `.cpm` file.

ALLOW_POWER_PINS
ASSOC_PARENT_PIN_SPACING
AUTO_CONNECT_DPLEG
CONNECTION_SWAP_PINS
DPPIN_PREFIX
DPPIN_RULES
DPSIG_PREFIX
DPSIG_RULES
ENABLE_SEL_LOGICAL_UPDATE_VDD
GEN_PSTFILES
LAUNCH_OPTION
OVERWRITE_CONSTRAINTS
PACKAGE_PROP
PRESERVE_BYPASS
PRESERVE_CONN
PRESERVE_PINPAIR
PRESERVE_PWRGRP
PRESERVE_REFDES
PRESERVE_TERMINATION
PRESERVE_USERPROP
REPORT_COL_PAD
REPORT_COL_SEP
REPORT_CURRENCY_CHAR
REPORT_DIR
REPORT_FILES
REPORT_FONT_FACE
REPORT_FONT_SIZE
REPORT_FONT_STYLE
REPORT_FORMAT
REPORT_HEADER_SEP
REPORT_HIDE_LINENO
REPORT_ROW_SEP
REPORT_SORT_ORDER
REPORT_STRING_SEP
SD_PREFIX_SEPARATOR
SD_SUFFIX_SEPARATOR
SHOW_ANALYZE_DIALOG
SUPPORTLIBRARYDEFINEDDIFFPAIR

Schgen CPM Directives

This chapter lists the CPM directives for System Connectivity Manager. These directives are specified in the `START_DSSCHGEN` section of a `.cpm` file.

ADD_AUTHOR_TO_COMMENT
ADD_BBOX
ADD_BLOCK_DIAGRAM
ADD_COMMENTS
ADD_COMMENTS_TO_INSTANCES
ADD_COMMENTS_TO_NETS
ADD_COMMENTS_TO_PINS
ADD_DATE_TO_COMMENT
ADD_TEXT_TO_COMMENT
ADD_TIME_TO_COMMENT
BBOX_COLOR
BLOCK_DIAGRAM_CELL
BLOCK_DIAGRAM_LIB
COMMENT_COLOR
COMP_COLOR
CTAP
IGNORE_BLOCK_WITHOUT_SCHEMATIC
IGNORE_INSTANCE_WITH_ERRORS
INPORT
INST_DEFAULT_ALIGNMENT
INST_DEFAULT_PROP_SIZE
INST_DEFAULT_VISIBILITY
IOPORT
LEFT_IN_OFFPAGE
LEFT_IN_ROT
LEFT_IO_OFFPAGE
LEFT_IO_ROT
LEFT_OUT_OFFPAGE
LEFT_OUT_ROT
OFFPAGE
OUTPORT
PAGE_BORDER
PLACEMENT_WITHIN_GROUP_USING_ORDER_IN_DESIGN
POWER_SYMBOLS
PROP_COLOR
RIGHT_IN_OFFPAGE
RIGHT_IN_ROT
RIGHT_IO_OFFPAGE
RIGHT_IO_ROT
RIGHT_OUT_OFFPAGE

RIGHT_OUT_ROT
USE_POWER_SYMBOLS
USE_OFFPAGE
WIRE_COLOR

Part Developer CPM Directives

This chapter lists the CPM directives for Part Developer. These directives are specified in the `START_PDV` section of a `.cpm` file.

ConceptSetup_Assertion_Read
ConceptSetup_Assertion_UseMinusInChips
ConceptSetup_Assertion_Write
ConceptSetup_SplitPart_AddSwapInfo
ConceptSetup_SplitPart_SymbolProp
Default_Diffpair_Value
Default_Zoom_Factor
DiffPair_Recognition_Rules
DISCONNECT_PIN_TEXT_NAME
Export_Csv_Replace_<property>
Export_Csv_Delimeter
Global_Modify_Pin_Graphics
Export_ViewLogic_Visibility_<property>
Import_APD_PwrGndNCPinsInGlobalSection
Import_AllegroFtprint_DefaultPinType
Import_Csv_ApplyVectorConversion
Import_APD_strippinnum
Import_CSV_GenerateSymbolForNoLocation
Import_Csv_Delimeter
Import_Csv_Replace_assertion
Import_Csv_LowAssertFlag
Import_Csv_Replace_DIFFPAIRPINSNEG
Import_Csv_Replace_assertionchar
Import_Csv_Replace_jedectype
Import_Csv_Replace_DIFFPAIRPINSPOS
Import_Csv_Replace_packagename
Import_Csv_Replace_loadsetupfile
Import_Csv_Replace_pinname
Import_Csv_Replace_pinlocation
Import_Csv_Replace_pinposition
Import_Csv_Replace_pinnumber
Import_Csv_Replace_pintype
Import_Csv_Replace_pinshape
Import_DML_Braces_TreatedAs_Vector
Import_Csv_Replace_symbol
Import_DML_Pins_DefaultVector
Import_DML_DefaultPinType
Import_FPGA_DefaultPinType

Import_DML_PwrGndNCPinsInGlobalSection
Import_FPGA_Standard_PartNameAsCellName
Import_FPGA_PwrGndNCPinsInGlobalSection
Import_FPGA_Xilinx_SeparationChar
Import_FPGA_Xilinx_CSVSeparationChar
Import_IBIS_With_Ibischk4
Import_IBIS_With_Dmlcheck
Import_Text_Braces_TreatedAs_Vector
Import_IBIS_With_Unchanged_ModelName
Import_Text_Pins_DefaultVector
Import_Text_DefaultPinType
Import_ViewLogic_gridratio
Import_Text_PwrGndNCPinsInGlobalSection
Package_Class
Instantiation_Packaging_Validation_Type
Package_PinDelayUnit
Package_JedecType
PackagePin_AbsentChar
Package_RefDesPrefix
PackagePin_Property_ANALOG_Connect
PackagePin_Property_ANALOG_Assert
PackagePin_Property_ANALOG_IO
PackagePin_Property_ANALOG_Dir
PackagePin_Property_ANALOG_UnknownLoading
PackagePin_Property_ANALOG_Load
PackagePin_Property_BIDIR_Connect
PackagePin_Property_BIDIR_Assert
PackagePin_Property_BIDIR_IO
PackagePin_Property_BIDIR_Dir
PackagePin_Property_BIDIR_UnknownLoading
PackagePin_Property_BIDIR_Load
PackagePin_Property_GROUND_Connect
PackagePin_Property_GROUND_Assert
PackagePin_Property_GROUND_IO
PackagePin_Property_GROUND_Dir
PackagePin_Property_GROUND_UnknownLoading
PackagePin_Property_GROUND_Load
PackagePin_Property_INPUT_Connect
PackagePin_Property_INPUT_Assert
PackagePin_Property_INPUT_IO
PackagePin_Property_INPUT_Dir
PackagePin_Property_INPUT_UnknownLoading
PackagePin_Property_INPUT_Load
PackagePin_Property_NC_Connect
PackagePin_Property_NC_Assert
PackagePin_Property_NC_IO

PackagePin_Property_NC_Dir
PackagePin_Property_NC_UnknownLoading
PackagePin_Property_NC_Load
PackagePin_Property_OC_Connect
PackagePin_Property_OC_Assert
PackagePin_Property_OC_IO
PackagePin_Property_OC_Dir
PackagePin_Property_OC_UnknownLoading
PackagePin_Property_OC_Load
PackagePin_Property_OCBIDIR_Connect
PackagePin_Property_OCBIDIR_Assert
PackagePin_Property_OCBIDIR_IO
PackagePin_Property_OCBIDIR_Dir
PackagePin_Property_OCBIDIR_UnknownLoading
PackagePin_Property_OCBIDIR_Load
PackagePin_Property_OE_Connect
PackagePin_Property_OE_Assert
PackagePin_Property_OE_IO
PackagePin_Property_OE_Dir
PackagePin_Property_OE_UnknownLoading
PackagePin_Property_OE_Load
PackagePin_Property_OEBIDIR_Connect
PackagePin_Property_OEBIDIR_Assert
PackagePin_Property_OEBIDIR_IO
PackagePin_Property_OEBIDIR_Dir
PackagePin_Property_OEBIDIR_UnknownLoading
PackagePin_Property_OEBIDIR_Load
PackagePin_Property_OUTPUT_Connect
PackagePin_Property_OUTPUT_Assert
PackagePin_Property_OUTPUT_IO
PackagePin_Property_OUTPUT_Dir
PackagePin_Property_OUTPUT_UnknownLoading
PackagePin_Property_OUTPUT_Load
PackagePin_Property_POWER_Connect
PackagePin_Property_POWER_Assert
PackagePin_Property_POWER_IO
PackagePin_Property_POWER_Dir
PackagePin_Property_POWER_UnknownLoading
PackagePin_Property_POWER_Load
PackagePin_Property_UNSPEC_Connect
PackagePin_Property_UNSPEC_Assert
PackagePin_Property_UNSPEC_IO
PackagePin_Property_UNSPEC_Dir
PackagePin_Property_UNSPEC_UnknownLoading
PackagePin_Property_UNSPEC_Load
PINALIAS_<PIN_TYPE>

PinType
Symbol_OutLine
Symbol_Pintext_LeftRight_XOffset
Symbol_Pintext_TopBottom_XOffset
Symbol_PN_LeftRight_XOffset
Symbol_PN_TopBottom_XOffset
Symbol_SymSheetSize
Symbol_Width
Symbol_Length
Symbol_PinShape_Dot_Size
Symbol_Pintext_LeftRight_YOffset
Symbol_Pintext_TopBottom_YOffset
Symbol_PN_LeftRight_YOffset
Symbol_PN_TopBottom_YOffset
Symbol_Units

Engineering Data Management Directives (EDM)

This section lists the Engineering Data Management (EDM) directives that are available in the *project*.cpm file. These directives control the behavior of EDM or store information about tasks and processes in EDM.

Start Design

Project_Ppl

Allegro EDM Flow Manager

LastFlow

Part Information Manager

CENTRAL_INDEX_PATH
DAO_Timeout
DataCompress
Datasheet_URL
Default_Search_Tab
Default_ShoppingCart_Quantity
Detail_Tab_Order
Display_URL
Max_Search_Rows
Minimize_On_Add
Online_Mode
Ppl_Only
Search_Attr_Name
Search_Attr_Type
Search_Result_Type
Search_Tab_Order
Search_Tolerance
Search_Tree_Order
Show_Cell
Show_Library
Single_Click_Details
Viewers

Library Revision Manager

sync_properties
auto_fix_ptf
auto_load_schematic_instances
auto_update_minor_cell
auto_update_minor_ptf
check_injected_order
auto_update_schematic
exclude_autoupdate_props
check_local_modified
dump_FileName
lrm_logfile

Cache Management

old_versions_count

Allegro Design Management (Team Design) Directives

This section lists the CPM directives for Allegro Design Management (team design). These directives are specified in the `START_SDM...END_SDM` section of the `.cpm` file.

DASHBOARD_SHOW_LOGICAL_HIERARCHY
DASHBOARD_SHOW_PHYSICAL_HIERARCHY
DASHBOARD_SHOW_WORKING_DESIGN

Allegro System Capture CPM Directives

This section lists the CPM directives for Allegro System Capture. These directives are specified in the `START_CANVAS...END_CANVAS` section of the `.cpm` file, unless otherwise specified in the descriptions of the commands.

Schematic Canvas CPM Directives

ALLOW_4WAY_JUNCTION
ALLOWED_ALTERNATE_PART_PROP
AUTO_NAMEONBUS_BITS
AUTO_NETNAMEON_POWERNET
AUTO_SIGNAME_LOC
AUTO_UPDATE_PARTS_ON_START
B2F_OVERWRITE_CONSTRAINTS
CAPTURE_CACHE_LIBRARY_PATH
BUS_TAP_SYMBOL_ROT<angle>
CREATE_CACHE_PROJECT
CAPTURE_STANDARD_LIB
DISCONNECT_PIN_TEXT_NAME
CTAP
DRC_ERROR
DISPLAY_UNCONNECTED_PINS
DRC_WARN
DO_NOT_HONOR_ANNOTATIONS
HYPERLINK_HIGHLIGHTED_DARK_THEME_COLOR
DRC_INFO
HYPERLINK_VISTED_COLOR
EXTERNAL_ALLEGRO_BOARD_FOLDER
INOUT_PORT_PIN_SIDE
HYPERLINK_HIGHLIGHTED_LIGHT_THEME_COLOR
NEW_PROPERTY_VISIBILITY
IN_PORT_PIN_SIDE
ORIENT_NET_NAME_DISPLAY
NETSPLIT_SUFFIX
PAGE_DEFAULT_SIZE
OPEN_ONLY_ACTIVE_TAB
RESTRICTIVE_WIRE_MOVE
OUT_PORT_PIN_SIDE
SAVE_TCL_IN_PROJECT
REPORT_DIR

SHOW_NET_HOTSPOTS
SHOW_PART_MANAGER_ON_START
STUB_LENGTH
UPPERCASE_SIGNAL_NAMES
TEXT_EDITOR
VARIANT_EDITOR_DISPLAY_PROP_NAME
RETAIN_ZERONODE_NET
SDA_CAPTURE_SPECIAL_LIB
SHOW_NET_NAME_DIALOG
SHOW_UNCONNECTED_PINS
TOC_PAGE_DEFAULT_SIZE

Schematic Grid CPM Directive

GRID_DISPLAY_ENABLED
GRID_STYLE
GRID_UNIT_MEASURE
GRID_PIN_PITCH
GRID_SNAP_FRACTION
GRID_DOC_SNAP_FRACTION
GRID_DISPLAY_MULTIPLE
GUIDE_LINES_COLOR_DARK
GUIDE_LINES_COLOR_LIGHT

Object Formatting CPM Directives

APPLY_<OBJECT>_STYLE
APPLY_NAVLINKS_STYLE
NAVLINKS_TEXT_FONT_BOLD
NAVLINKS_TEXT_FONT_COLOR
NAVLINKS_TEXT_FONT_ITALIC
NAVLINKS_TEXT_FONT_NAME
NAVLINKS_TEXT_FONT_SIZE
NAVLINKS_TEXT_FONT_UNDERLINE
NAVLINKS_TEXT_MARGIN
APPLY_TABLE_STYLE
TABLE_ALTERNATE_FILL_COLOR
TABLE_FILL_COLOR
TABLE_FILL_STYLE
TABLE_HEADER_FILL_COLOR
TABLE_LINE_CAP_STYLE
TABLE_LINE_COLOR
TABLE_LINE_JOIN_STYLE
TABLE_LINE_STYLE
TABLE_LINE_WIDTH
TABLE_TEXT_FONT_BOLD
TABLE_TEXT_FONT_COLOR
TABLE_TEXT_FONT_ITALIC
TABLE_TEXT_FONT_NAME
TABLE_TEXT_FONT_SIZE
TABLE_TEXT_FONT_UNDERLINE
TABLE_TEXT_MARGIN
APPLY_NETGROUP_STYLE
NETGROUP_FILL_COLOR
NETGROUP_FILL_STYLE
NETGROUP_LINE_CAP_STYLE
NETGROUP_LINE_COLOR
NETGROUP_LINE_JOIN_STYLE
NETGROUP_LINE_STYLE
NETGROUP_LINE_WIDTH
NETGROUP_TEXT_FONT_BOLD
NETGROUP_TEXT_FONT_COLOR
NETGROUP_TEXT_FONT_ITALIC
NETGROUP_TEXT_FONT_NAME
NETGROUP_TEXT_FONT_SIZE
NETGROUP_TEXT_FONT_UNDERLINE

NETGROUP_TEXT_MARGIN
<OBJECT>_LINE_WIDTH
<OBJECT>_LINE_STYLE
<OBJECT>_LINE_CAP_STYLE
<OBJECT>_LINE_JOIN_STYLE
<OBJECT>_LINE_COLOR
ALTERNATE_FILL_COLOR
<OBJECT>_FILL_STYLE
<OBJECT>_TEXT_FONT_NAME
<OBJECT>_TEXT_FONT_SIZE
<OBJECT>_TEXT_FONT_ITALIC
<OBJECT>_TEXT_FONT_BOLD
<OBJECT>_TEXT_FONT_UNDERLINE
<OBJECT>_TEXT_FONT_COLOR
<OBJECT>_LINE_COLOR
GRAPHIC_BLOCK_LINE_COLOR
GRAPHIC_BLOCK_FILL_COLOR
GRAPHIC_CONNECTOR_LINE_COLOR
GRAPHIC_CONNECTOR_FILL_COLOR
APPLY_VARIANT_INST_STYLE
VARIANT_INST_FILL_COLOR
VARIANT_INST_FILL_STYLE
VARIANT_INST_LINE_CAP_STYLE
VARIANT_INST_LINE_COLOR
VARIANT_INST_LINE_JOIN_STYLE
VARIANT_INST_LINE_STYLE
VARIANT_INST_LINE_WIDTH
VARIANT_INST_ITEM_OPACITY
APPLY_VARIANT_PROPERTY_STYLE
VARIANT_PROPERTY_TEXT_FONT_BOLD
VARIANT_PROPERTY_TEXT_FONT_COLOR
VARIANT_PROPERTY_TEXT_FONT_ITALIC
VARIANT_PROPERTY_TEXT_FONT_NAME
VARIANT_PROPERTY_TEXT_FONT_SIZE
VARIANT_PROPERTY_TEXT_FONT_UNDERLINE
VARIANT_PROPERTY_ITEM_OPACITY
VARIANT_DNI_CROSS
VARIANT_DNI_CROSS_COLOR
VARIANT_DNI_CROSS_LINE_WIDTH

Directives for the LINE Object

LINE_CAP_STYLE
LINE_COLOR
LINE_JOIN_STYLE
LINE_STYLE
LINE_WIDTH
SMART_PDF_LINE_WIDTH_FACTOR

Directives for the TEXT Objects

ITEM_OPACITY
TEXT_FONT_BOLD
TEXT_FONT_COLOR
TEXT_FONT_ITALIC
TEXT_FONT_NAME
TEXT_FONT_SIZE
TEXT_FONT_UNDERLINE
TEXT_MARGIN
TEXT_WORD_WRAP

Other Directives Related to Objects

PASTE_REPEATEDLY

Packaging CPM Directives

ALLOW_PINTEXT_SWAP
AUTO_XNETS_USING_GATES
ANNOTATE_GLOBAL_PTF_PROPS
COMP_DEF_PROP
ANNOTATE_ALL_PROPS
DEFAULT_PHYS_DES_PREFIX
BLOCK_REF_DES_PATTERN
HARD_REFDES_CONFLICT_RESOLVE
COMP_INST_PROP
NET_NAME_LENGTH
FilterZeroNet
PART_TYPE_LENGTH
NET_NAME_CHARS
REF_DES_PATTERN
PACKAGE_PROP
REFDES_ALPHA_NUM
REF_DES_LENGTH
REUSE_REFDES
REF_DES_PATTERN_FIX
SD_SUFFIX_SEPARATOR

REFDES_PAGE_PADDING
STRICT_PACKAGE_PROP
SD_PREFIX_SEPARATOR

TOC and Table Object CPM Directives

ALIASBODY
BOM_FOLDER
CP_NO_OF_COL_TOC
CP_NO_OF_ROW_TOC
CP_TOC_COL_<column_number>
CP_TOC_COL_<column_number>_LABEL
CREF_DATA_FILE
EDIT_PHYSICAL_NET_NAME
EDITABLE_IMPORT_TABLE
INPORT
INST_BLOCKAGE_MARGIN
IOPORT
MAX_COL_IMPORT_TABLE
MAX_ROW_IMPORT_TABLE
NO_CONNECT
OFFPAGE_INPUT
OFFPAGE_IO
OFFPAGE_OUTPUT
OUTPORT
PACKAGED_FOLDER
PHYSICAL_FOLDER
PIN_ASSIGNMENT_DIALOG_SHOW_BLOCK_NET_NAME
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER
POWER_SYMBOLS
PPT_OPTIONSET_PATH
PRESERVE_DAT
RAT_ON_REPLACE
TOC_AUTO_SAVE

Design Integrity CPM Directives

This section lists the CPM directives for Design Integrity. These directives are specified in the `START_RELIABILITY...END_RELIABILITY` section of the `.cpm` file.

AUDIT_DISALLOW_WAIVE
ILLEGAL_NET_NAME_CHAR
AUDIT_DISALLOW_WAIVE_RULES
INCREMENTAL_RUN
AUDIT_ERROR
LOWVOLTAGE_CLASS_PATTERN
AUDIT_FATAL
LOWVOLTAGE_THRESHOLD

AUDIT_INFO
AUDIT_WARNING
CAS_PIN_PATTERN
CLOCK_PIN_PATTERN
CS_PIN_PATTERN
DIFFPAIR_PIN_SUFFIX_N
DIFFPAIR_PIN_SUFFIX_P
ELECTRICAL_LOW_STRESS_LEVEL
ELECTRICAL_OVER_STRESS_LEVEL
FIDUCIAL_FOOTPRINT_NAME_PATTERN
FIDUCIAL_MIN_NUMBER
GLOBAL_NETS
GND_PIN_NAME
HOLE_FOOTPRINT_NAME_PATTERN
MISO_NET_PATTERN
MISO_PIN_PATTERN
MOSI_NET_PATTERN
MOSI_PIN_PATTERN
PULLDOWN_MAX
PULLDOWN_MIN
PULLUP_MAX
PULLUP_MIN
PWR_PIN_NAME
RAS_PIN_PATTERN
REFDES_PREFIX_VISIBILITY_CHECK
SCL_PIN_PATTERN
SDA_PIN_PATTERN
TESTPAD_FOOTPRINT_NAME_PATTERN

3

A Directives

This chapter lists the CPM directives that start with `A` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

ADD_AUTHOR_TO_COMMENT

The name of the user who added a comment to a design/instance/pin is added as a note in the generated document schematic.

Note: This directive only works if the `add_comments` directive and either the `add_comments_to_instances` or `add_comments_to_pins` directive is already specified in the `.cpm` file

Syntax

```
add_author_to_comment <0 or 1>
```

Example

```
add_author_to_comment '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Text

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_BBOX

Use this directive to add a bounding box to the generated schematic.

Syntax

```
add_bbox <0 or 1>
```

Example

```
add_bbox '0'
```

Corresponding UI Option for System Connectivity Manager

None

See Also

BBOX_COLOR

ADD_BLOCK_DIAGRAM

Use this directive to include the block diagrams in the specified lib:cell:sch_1 as initial pages of the document schematic. The library name and the cell name containing the block diagram are specified using the block_diagram_library and block_diagram_cell directives respectively.

Syntax

```
add_block_diagram <0 or 1>
```

Example

```
add_block_diagram '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — General — Include Block Diagram.

See Also

BLOCK_DIAGRAM_LIB

BLOCK_DIAGRAM_CELL

ADD_COMMENTS

Use this directive to add design comments as notes in the generated document schematic.

Syntax

```
add_comments <0 or 1>
```

Example

```
add_comments '1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Add Comments as Notes.

See Also

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_DATE_TO_COMMENT

ADD_TEXT_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_COMMENTS_TO_INSTANCES

Use this directive to indicates that comments added to component instances will get included as notes in the generated document schematic.

Note: This option works only if the add_comments directive is already added.

Syntax

```
add_comments_to_instances <0 or 1>
```

Example

```
add_comments_to_instances '1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Instance Comments.

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_DATE_TO_COMMENT

ADD_TEXT_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_COMMENTS_TO_NETS

Use this directive to include the comments added on the design signals as notes in the generated document schematic.

Note: This option works only if the add_comments directive is already added

Syntax

```
add_comments_to_nets <0>
```

Example

```
add_comments_to_nets '0'
```

Corresponding UI Option for System Connectivity Manager

None

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_DATE_TO_COMMENT

ADD_TEXT_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_COMMENTS_TO_PINS

Use this directive to indicate that comments added to component pins will get included as notes in the generated document schematic.

Note: This option works only if the add_comments directive is already added.

Syntax

```
add_comments_to_pins <0 or 1>
```

Example

```
add_comments_to_pins '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Pin Comments.

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_DATE_TO_COMMENT

ADD_TEXT_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_DATE_TO_COMMENT

Use this directive to add the date on which a particular comment was added to a design/instance/pin as a note in the generated document schematic.

Note: This option works only if the `add_comments` directive and either the `add_comments_to_instances` or `add_comments_to_pins` directive is already specified in the `.cpm` file.

Syntax

```
add_date_to_comment <0 or 1>
```

Example

```
add_date_to_comment '1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Date.

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_TEXT_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_TEXT_TO_COMMENT

Use this directive to indicate that the actual comment added to a design/instance/pin is added as a note in the generated document schematic.

Note: This option works only if the `add_comments` directive and either the `add_comments_to_instances` or `add_comments_to_pins` directive is already specified in the `.cpm` file.

Syntax

```
add_text_to_comment <0 or 1>
```

Example

```
add_text_to_comment '1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Text

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_DATE_TO_COMMENT

ADD_TIME_TO_COMMENT

ADD_TIME_TO_COMMENT

Use this directive to indicate that the time at which a particular comment was added to design/instance/pin is added as a note in the generated document schematic.

Note: This option works only if the `add_comments` directive and either the `add_comments_to_instances` or `add_comments_to_pins` directive is already specified in the `.cpm` file.

Syntax

```
add_time_to_comment <0 or 1>
```

Example

```
add_time_to_comment '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Comments — Time

See Also

ADD_COMMENTS

ADD_COMMENTS_TO_NETS

ADD_COMMENTS_TO_PINS

ADD_DATE_TO_COMMENT

ADD_TEXT_TO_COMMENT

ALIASBODY

Using this directive, you can add a symbol to the *Alias* section of the *Special Symbols* bucket.

Syntax

```
ALIASBODY '<library:cell:view>'
```

where

library	Enter the library name in which you want the symbol to be added.
cell	Enter the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	Enter the view name in which you want the symbol to be added.

Example

```
ALIASBODY 'standard.alias:sym_1'
```

Corresponding UI Option

None

ALLOW_4WAY_JUNCTION

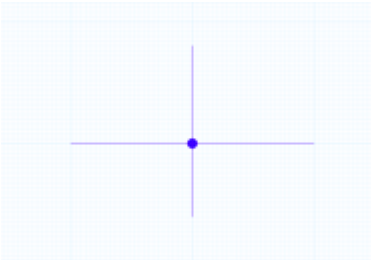
Controls the creation of a four-way wire intersection.

Syntax

```
ALLOW_4WAY_JUNCTION '1'|'0'
```

where

1 Creation of a four-way junction is allowed. This is the default value.



0 Creation of a four-way junction is not allowed.

Example

```
ALLOW_4WAY_JUNCTION '1'
ALLOW_4WAY_JUNCTION '0'
```

Corresponding UI Option for Allegro System Capture

None

ALLOW_COMP_OVERLAP

When set to ON, this directive allows you to add, move, or copy a component that overlaps another component's origin. The directive also disables the error message that would indicate an overlap.

Syntax

```
ALLOW_COMP_OVERLAP 'ON'|'OFF'
```

Example

```
ALLOW_COMP_OVERLAP 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_EMPTY_LOCK_FILE

Specifies whether Design Entry HDL should continue when a design has empty lock (.lck) files owned by the design owner. If this directive is ON, empty, design-level `.dcf` and `.xcon` lock files are deleted after you perform various operations (e.g., page management) and save your design or save the hierarchy.

Note: If DE-HDL exits unexpectedly, empty `.dcf` or `.xcon` lock files may remain despite this directive being ON. To remove these stale lock files, use the `run_rstlock` command from within DE-HDL.

Syntax

```
ALLOW_EMPTY_LOCK_FILE 'ON'|'OFF'
```

Example

```
ALLOW_EMPTY_LOCK_FILE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_FOOTPRINT_COMPATIBILITY_CHECK

When on, defines whether DE-HDL should check for footprint compatibility between components when components are modified or replaced.

You can define three values for this directive:

- `always` - DE-HDL will always check for footprint compatibility for all components in the design
- `inst` - define the `JEDEC_TYPE_CHECK` property for symbols or instances that should be checked for footprint compatibility. The value of this property can be set as 1, ON, or TRUE. When modifying or replacing components, only those instances that have this property will be checked for compatible footprints. You can also define compatible footprints using a file named `cjedectype.txt`. See Design Entry HDL User Guide for details on this file.
- `disable` - DE-HDL will not check whether components have matching footprints

When searching for compatible footprints, DE-HDL finds compatible footprints by matching the source component and target component JEDEC types and ATL_SYMBOLS.

DE-HDL first searches the instance property for JEDEC and ALT_SYMBOLS, then PTF properties and then the properties in `chips.prt`. If you have a `cjedectype.txt` file in the SITE area, DE-HDL also checks this file for compatible footprints.

Syntax

```
ALLOW_FOOTPRINT_COMPATIBILITY_CHECK 'ALWAYS'|'INST'|'DISABLE'
```

Example

```
ALLOW_FOOTPRINT_COMPATIBILITY_CHECK 'INST'
```

Corresponding UI Option

None

ALLOW_HFS_SWAPS

Set this directive to enable System Capture to swap sections for ‘Has fixed size’ parts in the back-to-front flow, that is when updating the schematic with the layout changes.

Syntax

```
ALLOW_HFS_SWAPS 'ON' | 'OFF'
```

By default, this directive is set to `'NO'`.

Example

```
ALLOW_HFS_SWAPS 'ON'
```

Corresponding UI Option

None

ALLOW_IMPORT_DESIGN_AT_SITE_UNIT

This directive controls the grid settings and pin-to-pin spacing for the designs being imported into System Capture. By default, this directive is set to `NO` and the System Capture designs continue to use the same settings as the DE-HDL source designs.

Set this variable to `YES` to force the import process to follow the settings specified in the `site.cpm` file.

Syntax

```
ALLOW_IMPORT_DESIGN_AT_SITE_UNIT = ‘NO’
```

By default, this directive is set to `'YES'`.

Example

```
ALLOW_IMPORT_DESIGN_AT_SITE_UNIT = ‘YES’
```

Corresponding UI Option

None

ALLOW_IMPORT_DESIGN_NUDGE_OFFGRID_OBJECTS

This directive controls the import of DE-HDL designs into Allegro System Capture. By default, this directive is set to `NO`. This means that when the import process finds the design being imported as a block or sheet with different grid settings and pin-to-pin spacing, the import stops. Set this variable to `YES` to force the import process to adjust or nudge the components or blocks to the destination design's grid settings. When enabled, System Capture:

- Adjusts the placement of components to match the destination design's grid settings.
- Nudges hierarchical block pins
- If a design has symbols with offgrid pins, it is not imported.

Syntax

```
ALLOW_IMPORT_DESIGN_NUDGE_OFFGRID_OBJECTS = 'NO'
```

By default, this directive is set to `'YES'`.

Example

```
ALLOW_IMPORT_DESIGN_NUDGE_OFFGRID_OBJECTS = 'YES'
```

Corresponding UI Option

None

ALLOW_IMPORT_LIBCELL_MISMATCH

By default, design or sheet import or paste operations across System Capture projects require the version of the cells in the schematic being pasted or imported to be the same across designs. When circuitry is copied from read-only designs into a working design and the parts are out-of-date in the read-only Part Manager needs to be first run to sync the parts.

However, it is possible to bypass this requirement and paste or import even when the cells across projects do not match.

Note: In case of mismatch, the cell in the target is retained and the content being imported is refreshed based on the cell version in the target design.

Syntax

```
ALLOW_IMPORT_LIBCELL_MISMATCH 'TRUE'|'FALSE'
```

Example

```
ALLOW_IMPORT_LIBCELL_MISMATCH 'TRUE'
```

Corresponding UI Option in Allegro System Capture

Preferences window – Schematic – Component – Part Manager – Allow import or paste even if component cells do not match

ALLOW_INCOMPATIBLE_JEDEC_TYPE

Set this directive to FALSE to ensure that only parts with compatible footprints can replace an instantiated part. Parts that have matching JEDEC_TYPE and ATL_SYMBOLS are referred to as parts with compatible footprints. In this case, if you try to replace or modify a part with another part that has a different footprint, an error message similar to the following is displayed with the relevant PART_NUMBER and JEDEC_TYPE:

```
Error: Cannot replace part ’2N4339’ (footprint ’SOR23’) with part
’RES’ (footprint ’1206_T’) because the footprints are incompatible.
Only parts with compatible footprints can replace each other. Select
a part with a compatible footprint then replace the part.
```

Set this directive to TRUE to allow modification or replacement of parts that have incompatible footprints. In this case, a warning message similar to the following is displayed with the relevant PART_NUMBER and JEDEC_TYPE prompting you for confirmation:

```
Warning: Part ’2N4339’(footprint ’SOR23’) is being replaced with
part ’RES’ (footprint ’1206_T’). These footprints are incompatible,
do you still want to replace.
```

For more information about replacing components with different JEDEC_TYPEs, refer to the *Using Compatible JEDEC_TYPEs* section of *Design Variance Tutorial*.

Syntax

```
ALLOW_INCOMPATIBLE_JEDEC_TYPE 'TRUE'|'FALSE'
```

By default, this directive is set to 'TRUE'.

Example

```
ALLOW_INCOMPATIBLE_JEDEC_TYPE 'TRUE'
```

Corresponding UI Option

None

ALLOW_PAGE_LOCKING

Controls the locking of a page. When a user who has write permissions is editing a page in a design, Design Entry HDL locks the page. If a second user opens the same page for editing, Design Entry HDL displays a message that the page is locked by the first user and that the second user cannot save any changes made in the page. Design Entry HDL creates a lock file called `page<n>_csb.lck` in the schematic view when you open a schematic page.

Syntax

```
ALLOW_PAGE_LOCKING 'ON'|'OFF'
```

Example

```
ALLOW_PAGE_LOCKING 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_PINTEXT_SWAP

By default, pin swaps received from back-end or performed on the schematic swap only pin numbers assigned to the pins. When this directive set to ON, both pin names and pin numbers assigned to the pins are swapped. This directive applies to DE-HDL and Allegro System Capture.

Syntax

```
ALLOW_PINTEXT_SWAP 'ON|OFF'
```

Example

```
ALLOW_PINTEXT_SWAP 'ON'
```

Corresponding UI Option

None

ALLOW_POWER_PINS

Controls the overwriting of existing POWER_PINS property on an instance. Use this directive if you want the properties in the chips.prt file to take priority.

The ALLOW_POWER_PINS directive can also be used in System Connectivity Manager to display power pins in the Component Connectivity pane.

Syntax

```
ALLOW_POWER_PINS 'ON'|'OFF'
```

where

ON	The POWER_PINS properties can be edited from the Assign Power Pins dialog box. This is the default value.
OFF	You cannot edit the POWER_PINS property from the Assign Power Pins dialog box. Design Entry HDL reads the POWER_PINS properties only from the chips.prt file and POWER_GROUP property from the instance. If POWER_PINS and NC_PINS properties are present on the instance, an error message is flagged that, because the ALLOW_POWER_PINS directive is set to OFF, POWER_PINS, NC_PINS, MERGE_POWER_PINS, and MERGE_NC_PINS will not be read from or assigned to the instance.

Example

```
ALLOW_POWER_PINS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Design Entry HDL — Text — Assign Power Pins

Corresponding UI Option for System Connectivity Manager

None

ALLOW_PROP_ILLEGAL_OBJECT

This directive allows any default property definitions to be added to any object in the schematic regardless of the validity of the property. For example, you can add the `PACK_TYPE` property to a wire. The directive is `ON` by default and is in the `START_NETLIST` section of a `.cpm` file.

If you want to restrict properties from being incorrectly added to an object, set this directive to `OFF`. Doing so will generate information message in the *Command Console* window whenever you add an incorrect property to any object and save the design. The message type cannot be changed.

When this directive is set to `OFF`, incorrect properties added to objects are not added to the netlist.

Note: Even if you set this directive to `OFF`, user-defined properties that are incorrectly added to objects will be ignored. For example, if you add a property such as `USER_PIN_PROP` to a wire, it will not be flagged.

Syntax

```
ALLOW_PROP_ILLEGAL_OBJECT 'ON'|'OFF'
```

Example

```
ALLOW_PROP_ILLEGAL_OBJECT 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_PROP_ON_PROP

When this directive is set to ON, it allows you to add a property to another property (for example, SIG_NAME).

Syntax

```
ALLOW_PROP_ON_PROP 'ON'|'OFF'
```

Example

```
ALLOW_PROP_ON_PROP 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_PROPERTY_LOCKING

Controls the locking and unlocking of the key properties in the schematic.

Syntax

```
ALLOW_PROPERTY_LOCKING 'ON'|'OFF'
```

Example

```
ALLOW_PROPERTY_LOCKING 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_PTFVALUE_INITIAL_BLANKS

Specifies whether the *initial blanks* in the ptf value in a ptf row will be preserved while loading/comparing ptf properties.

Syntax

```
ALLOW_PTFVALUE_INITIAL_BLANKS 'ON'|'OFF'
```

Example

```
ALLOW_PTFVALUE_INITIAL_BLANKS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOW_SINGLE_NAVLINK_PER_PAGE

Consolidates the duplicate values of the navigation links and shows a single entry for the same.

Syntax

```
ALLOW_SINGLE_NAVLINK_PER_PAGE 'TRUE'|'FALSE'
```

Example

```
ALLOW_SINGLE_NAVLINK_PER_PAGE 'TRUE'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALLOWED_ALTERNATE_PART_PROP

Set this directive if you want a part to be replaced or modified only with specific components from the library.

The value of this directive must be a property, such as `PART_NUMBER` or `VALUE`, on the basis of which a part can be replaced. The value of the property is specified in the `ALLOWED_ALT_PARTS` attribute, which is added to the part to be replaced. In the *Value* column of the *Attributes* dialog, for the `ALLOWED_ALT_PARTS` attribute, you need to specify one or more of the values of the property you defined in the `ALLOWED_ALTERNATE_PART_PROP` directive.

The values of the property you define in the `ALLOWED_ALTERNATE_PART_PROP` directive can be a comma-separated list of values. You can use an asterisk (*) and question mark (?) as characters in the list. For more information on `ALLOWED_ALT_PARTS`, refer to the *ALLOWED_ALT_PARTS* section of *Allegro Platform Properties Reference*.

For example, if you want a part with part number `QD-000045-00` to be replaced or modified only with parts `QD-000295-30`, `QD-000077-00`, or `QD-000372-00`, set the value of this directive to `PART_NUMBER`. You must also add the `ALLOWED_ALT_PARTS` attribute to part `QD-000045-00` with its value as `QD-000295-30`, `QD-000077-00`, `QD-000372-00`.

Design Entry HDL checks and replaces a part only if its property value matches with any one of the property values of parts specified in the `ALLOWED_ALT_PARTS` attribute.

Syntax

```
ALLOWED_ALTERNATE_PART_PROP '<part_property_name>'
```

where

part_property_name	Any part property name

Example

```
ALLOWED_ALTERNATE_PART_PROP 'PART_NUMBER'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ALTERNATE_FILL_COLOR

Sets the alternate fill color to be used for block arrows.

Syntax

```
ALTERNATE_FILL_COLOR '<fill_color>'
```

fill_color	The default fill color for block arrows. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FFFFFF'` or `'WHITE'` to represent white. The default value is `'#000000'` (Black).

Examples

```
ALTERNATE_FILL_COLOR 'BLACK'
ALTERNATE_FILL_COLOR '#000000'
```

ANNOTATE

The ANNOTATE directive controls the type of property information backannotated to the schematic by using the `pstback.dat` file.

Syntax

```
ANNOTATE on|off | option [,option]...;
```

on	Lets you backannotate body, pin, and net information.
off	Does not generate the backannotation file, `pstback.dat`. If ANNOTATE directive is set to OFF, the backannotation files will not be generated even if the *Backannotate Packaging Properties to Schematic Canvas* option is selected in the Export Physical dialog box.
option	Backannotates the following types of physical information: ■ *body* – Reference designators and body properties ■ *pin* – Pin numbers and pin properties ■ *net* – Net properties This option is for scalar nets only. We do not recommend backannotating net properties. Packager-XL has no way of determining the source of a net property; therefore, backannotated properties are placed on every occurrence of a net.
phys_net_name	Represents the physical net names used in the board.

The default value for the ANNOTATE property is both body and pin options. Body, net, and pin properties are backannotated only when their value in the state file is different from their value in the schematic. Structured parts and hierarchical modules that are used more than once are not included in the backannotation file.

Example

```
ANNOTATE pin;
```

Corresponding UI Option

None

ANNOTATE_ALL_PROPS

Set this directive if you want to annotate all Library Part properties (PTF properties) on the part instance in the design. When the value of the property is ON, all the library properties are annotated on the instance The properties if included in ppt option set are added to the design based on the annotation settings in the PPT option set. Any properties not included in PPT Option set are added as invisible properties on the instance.

When this directive is set to ON/TRUE/1, Physical Netlist is generated based on the annotated property set and the parts in the design are not checked to the reference library parts. This might result in a situation where the design has Part Manager mismatches but Export to PCB Layout still works without reporting any errors. To avoid such a scenario, run Part manager on the schematic design whenever ANNOTATE_ALL_PROPS is set to ON before running *Export to PCB Layout* command.

Syntax

```
ANNOTATE_ALL_PROPS'<BOOL>'
```

where

`BOOL`	On | True | 1
	Off | False | 0

Example

```
ANNOTATE_ALL_PROPS'ON'
```

Corresponding UI Option for Allegro System Capture

None

ANNOTATE_GLOBAL_PTF_PROPS

Set this directive to import global PTF properties from DE-HDL designs to System Capture designs. When this directive is enabled, properties are imported and any potential loss of discretes, XNets, or constraints is avoided.

By default, this directive is set to false. For designs that depend on global PTF properties for identifying discrete, this might result in loss of XNets and related constraints.

Note: When enabled, parts may get flagged as auto-sync in Part Manager because adding the annotations is treated as a library change in System Capture.

Syntax

```
ANNOTATE_GLOBAL_PTF_PROPS '<BOOL>'
```

where,

`BOOL`	On | True | 1
	Off | False | 0

Example

```
ANNOTATE_GLOBAL_PTF_PROPS 'true'
```

ANNOTATE_PART_NAME

Annotates single primitive parts with the PART_NAME property. Add this directive to the project or site level `.cpm` file and all the parts will annotate the PART_NAME value in all cases.

Caution

Selectively turning off this directive is not recommended as there could be problems for existing parts that are modified or changed. The changed parts may not update with the PART_NAME and cause the packager to fail. Use this directive only if you always intend to annotate the PART_NAME in all cases.

Syntax

```
ANNOTATE_PART_NAME 'TRUE'|'FALSE'
```

Example

```
ANNOTATE_PART_NAME 'TRUE'
```

Corresponding UI Option for Allegro Design Entry HDL

None

APPLY_<OBJECT>_STYLE

Controls whether to apply formatting styles defined by various CPM directives on the specified object type.

Syntax

```
APPLY_<OBJECT>_STYLE 'TRUE'|'FALSE'
```

OBJECT	Represents the object type for which the directive is set.

Example

```
APPLY_INST_STYLE 'TRUE'
APPLY_PIN_STYLE 'TRUE'
APPLY_BUS_STYLE 'TRUE'
APPLY_RAT_STYLE 'FALSE'
APPLY_ROUTE_STYLE 'TRUE'
APPLY_NOTE_STYLE 'FALSE'
APPLY_PROPERTY_STYLE 'TRUE'
APPLY_OCC_PROPERTY_STYLE 'FALSE'
APPLY_RICH_NOTE_STYLE 'TRUE'
APPLY_SIMPLE_NOTE_STYLE 'FALSE'
APPLY_GRAPHIC_BLOCK_STYLE 'TRUE'
APPLY_GRAPHIC_CONNECTOR_STYLE 'FALSE'
```

APPLY_NAVLINKS_STYLE

Controls whether to apply formatting styles defined by various CPM directives on navigation links.

Syntax

```
APPLY_NAVLINKS_STYLE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
APPLY_NAVLINKS_STYLE 'TRUE'
```

APPLY_NETGROUP_STYLE

Controls whether to apply formatting styles defined by various CPM directives on netgroups.

Syntax

```
APPLY_NETGROUP_STYLE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
APPLY_NETGROUP_STYLE 'TRUE'
```

APPLY_TABLE_STYLE

Controls whether to apply formatting styles defined by various CPM directives on tables.

Syntax

```
APPLY_TABLE_STYLE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
APPLY_TABLE_STYLE 'TRUE'
```

APPLY_VARIANT_INST_STYLE

Controls whether to apply formatting styles defined by various CPM directives on variant instances.

Syntax

```
APPLY_VARIANT_INST_STYLE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
APPLY_VARIANT_INST_STYLE 'TRUE'
```

APPLY_VARIANT_PROPERTY_STYLE

Controls whether to apply formatting styles defined by various CPM directives on variant properties.

Syntax

```
APPLY_VARIANT_PROPERTY_STYLE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
APPLY_VARIANT_PROPERTY_STYLE 'TRUE'
```

ARC_COLOR

Changes the default arc color to the specified value.

Syntax

```
ARC_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
ARC_COLOR 'yellow'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Arc

See Also

- BACKGROUND_COLOR
- DOT_COLOR
- NOTE_COLOR
- HIGHLIGHT_COLOR
- PROP_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

ASK_RENAME_SIGNAME_OPTION

When this directive is set to ON, and a user tries to change a net name, Design Entry HDL prompts users to confirm whether they want to change the name only for the instance, or rename the net across the design.

You can change a net name for one instance or change it across a design. To change a net name for one instance only, select the net text and choose Text — Change, or right-click and choose Change, or use the Attributes form. When you change a net name for one instance, it is equivalent to deleting the net name and re-adding it, which deletes the constraints on the net. The constraints on that the net whose name you changed are reset to the default.

When you rename a net across a design, the changed net name is propagated across the design and the constraint information is preserved. Choose the *Rename Signal* option to change a net name across a design.

Use the `ASK_RENAME_SIGNAME_OPTION` directive if you often use methods other than the *Rename Signal* option to rename a net, even when you wanted to rename a net across a design.

Syntax

```
ASK_RENAME_SIGNAME_OPTION 'ON'|'OFF'
```

Example

```
ASK_RENAME_SIGNAME_OPTION 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

ASSOC_PARENT_PIN_SPACING

Use this directive to specify the spacing between the pins of associated components and the parent pin.

Syntax

```
assoc_parent_pin_spacing '<spacing distance>'
```

Example

```
assoc_parent_pin_spacing '200mil'
```

Corresponding UI Option for System Connectivity Manager

None

ATTRIBUTEDATATOOLFORM

Displays the Model Tree for components and nets in the published PDF document. The Model Tree provides you with an easier way of viewing object properties. You need to add decimal values corresponding to the component (decimal value 2) and or the net (decimal value 4) to set the `ATTRIBUTEDATATOOLFORM` directive.

Syntax

```
ATTRIBUTEDATATOOLFORM '<decimal_value>'
```

Example

```
ATTRIBUTEDATATOOLFORM '6'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Export Attribute form in PDF File — Attribute Form — Data Tool check box

See Also

ATTRIBUTEJAVAFORM

ATTRIBUTEFILTER

Using this directive, you can specify property names which you do not want to export to the published PDF document. This way you can filter out unwanted properties from being exported to the published PDF document.

Filtered attributes are not visible in the Attribute dialog box of a part, net, or pin in a published PDF document. For example, if you decide to exclude the `SIG_NAME`, and `SIGNAL_MODEL` properties from the published PDF document, the following line is added:

```
START_PDF

ATTRIBUTEFILTER 'SIG_NAME' 'SIGNAL_MODEL'

END_PDF
```

Syntax

`ATTRIBUTEFILTER'<Attribute>'|'<Attribute>'|'<Attribute>'|'<Attribute>'`

Example

```
ATTRIBUTEFILTER 'SIG_NAME''SIGNAL_MODEL'
```

Corresponding UI Option in Design Entry HDL

- *File — Publish PDF — Setup — PDF — General — Attribute Filter*
- *Tools — Options — PDF — General — Attribute Filter*

ATTRIBUTEJAVAFORM

Displays the Attribute form for various schematic objects in the published PDF document. You need to add decimal values corresponding to the component (decimal value 2), net (decimal value 4), and pin (decimal value 8) to set the `ATTRIBUTEJAVAFORM` directive.

Syntax

```
ATTRIBUTEJAVAFORM '<decimal_value>'
```

Example

```
ATTRIBUTEJAVAFORM '14'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Export Attribute form in PDF File — Attribute Form — Java check box

See Also

ATTRIBUTEDATATOOLFORM

ATTRIBUTES_DIR

Sets the path for the location of attribute files. The default attribute file, *allegro_net.att* is located at `$CDS_INST_DIR/tools/fet/concept/attributes`.

Syntax

```
ATTRIBUTES_DIR '<path>'
```

where

path	is the path to the directory containing the attributes file (.att).

Example

```
ATTRIBUTES_DIR '$CDS_INST_DIR/tools/fet/concept/attributes'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Paths page — Input Paths — Attribute Directory

See Also

- CATPATH
- INPUT_SCRIPT
- PPT_OPTIONSET_PATH

AUDIT_DISALLOW_WAIVE

Defines the report type of the violations that cannot be waived during an audit analysis.

Syntax

```
AUDIT_DISALLOW_WAIVE '<Report Type>'
```

Example

```
AUDIT_DISALLOW_WAIVE 'FATAL'
```

Corresponding UI Option in Allegro System Capture

None

AUDIT_DISALLOW_WAIVE_RULES

Defines the rules that cannot be waived during schematic audit analysis.

Syntax

```
AUDIT_DISALLOW_WAIVE_RULES '<Audit Rule Name>'
```

Example

```
AUDIT_DISALLOW_WAIVE_RULES 'floatingResistor'
```

Corresponding UI Option in Allegro System Capture

None

AUDIT_ERROR

Defines the report type of the specified rules as ERROR.

Syntax

```
AUDIT_ERROR '<Audit Rule Name>'
```

Example

```
AUDIT_ERROR 'DPPinsPolarityMismatch' 'DPNetPinPolarityMismatch'
'DPNetUnconnected' 'DPNetSingleNodeNet' 'floatingBJT'
```

Corresponding UI Option in Allegro System Capture

None

AUDIT_FATAL

Defines the report type of the specified rules as FATAL. In shared designs, project owners and designers with *Edit* permission for the design can modify the report type.

Syntax

```
AUDIT_FATAL '<Audit Rule Name>'
```

Example

```
AUDIT_FATAL 'Differential pair net polarity mismatch'
```

Corresponding UI Option in Allegro System Capture

None

AUDIT_INFO

Defines the report type of the specified rules as INFO.

Syntax

```
AUDIT_INFO '<Audit Rule Name>'
```

Example

```
AUDIT_INFO 'asymmFunctionMissing' 'jedecMissing' 'unconnectedNet'
'singleNodeNet' 'powerNetMissingBypassCapacitor'
'ICOutputPinMissingVohVol'
'netWithBothPulledUpAndPulledDownResistors'
```

For details on audit rules, refer to *List of Rules for Schematic Audit*.

Corresponding UI Option in Allegro System Capture

None

AUDIT_WARNING

Defines the report type of the specified rules as WARNING.

Syntax

```
AUDIT_WARNING '<Audit Rule Name>'
```

Example

```
AUDIT_WARNING 'highPullUpValue' 'highPullDownValue' 'lowPullUpValue'
'lowPullDownValue' 'unrecognisedDevice'
```

Corresponding UI Option in Allegro System Capture

None

AUTO_CONNECT_DPLEG

Use this directive to ensure that when you connect one member net of a differential pair signal to a differential pair pin, the second member net automatically gets connected to the unconnected pin of a differential pair pin.

Syntax

```
auto_connect_dpleg 'ON|OFF'
```

Example

```
auto_connect_dpleg 'ON'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Differential Pairs — Auto Connect differential pairs in connectivity panes

auto_fix_ptf

Fixes the autofixable PTF row (corresponding to the cell selected in the *Cell/Block Details* pane) if the *Update* button is clicked.

This directive can be used with the following directives:

- `sync_properties`
- `auto_update_minor_ptf` when set to `TRUE`

Syntax

```
auto_fix_ptf 'TRUE' | 'FALSE'
```

Example

```
auto_fix_ptf 'TRUE'
```

Corresponding UI Option

None

See Also

sync_properties

auto_update_minor_ptf

auto_load_schematic_instances

Loads all the schematic instances and their reference designators (when set to *TRUE*) in the Schematic Instances area.

Syntax

```
auto_load_schematic_instances 'TRUE' | 'FALSE'
```

Example

```
auto_load_schematic_instances 'TRUE'
```

Corresponding UI Option

None

AUTO_NAMEONBUS_BITS

Controls automatic naming of bus bits.

Syntax

```
AUTO_NETNAMEONBUS_BITS 'YES'|'NO'
```

where,

YES	Netnames automatically appear on bus bits. This is the default value.
NO	Bus bits are not named automatically.

Example

```
AUTO_NETNAMEONBUS_BITS 'YES'
AUTO_NETNAMEONBUS_BITS 'NO'
```

AUTO_NETNAMEON_POWERNET

Controls automatic naming of a power net.

Syntax

```
AUTO_NETNAMEON_POWERNET 'YES'|'NO'
```

where

YES	Netnames automatically appears on power nets. This is the default value.
NO	Power nets are not shown automatically.

Example

```
AUTO_NETNAMEON_POWERNET 'YES'
AUTO_NETNAMEON_POWERNET 'NO'
```

AUTO_SIGNAME_LOC

Sets the location of the net name after it is displayed on a wire/bus connected between pins.

Syntax

```
AUTO_SIGNAME_LOC 'DRIVER'|'RECEIVER'
```

where,

DRIVER	Indicates that the signal name will be displayed near the out-pin.
RECEIVER	Indicates that the signal name will be displayed near the in-pin.

Example

```
AUTO_SIGNAME_LOC 'DRIVER'
AUTO_SIGNAME_LOC 'RECEIVER'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - Default Net Name Location

AUTO_UPDATE_DEFAULT_MODELS

When replacing or modifying a component in DE HDL, the default signal models associated with the component are automatically updated if this directive is ON.

Syntax

```
AUTO_UPDATE_DEFAULT_MODELS 'ON'|'OFF'
```

Example

```
AUTO_UPDATE_DEFAULT_MODELS 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

auto_update_minor_cell

Automatically updates the cells that have minor differences when the *Update* button is clicked.

Syntax

```
auto_update_minor_cell 'TRUE' | 'FALSE'
```

Example

```
auto_update_minor_cell 'TRUE'
```

Corresponding UI Option

None

auto_update_minor_ptf

If instantiated parts in a design project/cache have been modified by the librarian in the reference library, when you load the project in Flow Manager, LRM indicates the status of these parts as *Need Manual Update* and *Injected Header Mismatch*.

Rather than update these parts by selecting each required part table row in LRM, you can use the `auto_update_minor_ptf` directive.

- When set to TRUE, LRM auto-matches reference library parts whose key properties match parts in the design/cache but whose injected property values do not match. You can select the part rows using the check box, and click *Update*. If the `auto_fix_ptf` directive is also set to TRUE, LRM updates the part table rows only if there are injected property value mismatches. LRM will not update injected header or key differences.
- When the directive is set to FALSE, you will need to manually update the parts by right-clicking and selecting *Replace with* to update the cached parts with the ones from the reference library.
- When set to AUTO, LRM automatically updates the injected property and injected header differences. However, if any part has key and injected property differences, LRM will not auto-update the mismatch.

Note the following:

- The `auto_update_minor_ptf` directive can also be used when you do not want to auto-update part table rows but want to use the auto-update feature with the `sync_properties` directive.
- If both `auto_update_minor_ptf` and `check_injected_order` are set to TRUE, LRM considers both directives when updating the mismatches.
- If you do not want LRM to auto-update certain injected property mismatches, you can specify those properties using the `exclude_autoupdate_props` directive.

Syntax

```
auto_update_minor_ptf 'TRUE' | 'FALSE'| 'AUTO'
```

Example

```
auto_update_minor_ptf 'TRUE'
```

Corresponding UI Option

None

See Also

auto_fix_ptf

sync_properties

AUTO_UPDATE_PARTS_ON_START

Set this directive to TRUE, if you want Part Manager to automatically synchronize the design cache and reference libraries when a design is loaded.

Syntax

```
AUTO_UPDATE_PARTS_ON_START 'TRUE'|'FALSE'
```

The default value is FALSE.

Example

```
AUTO_UPDATE_PARTS_ON_START 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Component - Auto Update Parts

auto_update_schematic

Updates schematic instances (when set to *TRUE*) with the changes made using Library Revision Manager if the *Update* button is clicked.

Syntax

```
auto_update_schematic 'TRUE' | 'FALSE'
```

Example

```
auto_update_schematic 'TRUE'
```

Corresponding UI Option

None

AUTO_XNETS_USING_GATES

Using this directive you can define XNet creation mode.

Syntax

```
AUTO_XNETS_USING_GATES 'ON'|'OFF'
```

`ON`	Automatic XNets mode is enabled. In this mode, XNets are created automatically between pins of a discrete device. This is the default value.
`OFF`	Manual XNets mode is enabled. In this mode, XNets are not automatically created between the pins of a discrete device.

Example

```
AUTO_XNETS_USING_GATES 'ON'
```

Corresponding UI Option

None

AUTODOT

Automatically displays dots at wire connections.

Syntax

```
AUTODOT 'ON'|'OFF'
```

Example

```
AUTODOT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Dots — Auto Dot At Intersection

AUTOHEAVY

Automatically thickens a wire when you attach a bus signal name to it.

Syntax

```
AUTOHEAVY 'ON'|'OFF'
```

Example

```
AUTOHEAVY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Wires — Auto Heavy If Busname

See Also

AUTOROUTE

AUTOPAN

Enables panning behavior that lets you move the window over the drawing, rather than move the drawing inside the window.

Syntax

```
AUTOPAN 'ON'|'OFF'
```

Example

```
AUTOPAN 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Window Autopan

AUTOPATH

Automatically attaches the PATH property to an added part.

Syntax

```
AUTOPATH 'ON'|'OFF'
```

Example

```
AUTOPATH 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Text: Properties — Autopath Properties On Components

AUTOROUTE

Automatically routes a wire around objects when you move a component in the drawing.

Syntax

```
AUTOROUTE 'ON'|'OFF'
```

Example

```
AUTOROUTE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Wires — Auto Route On Move

See Also

AUTOHEAVY

AUTOSIZE_MARGIN

Using this directive extends the PDF page as needed, which helps you print schematics that are larger than one printed page. If you specify this directive, the width and height of the page, and the left, right, top, and bottom margins, as also the scaling factor are calculated by default.

The printer paper size is not changed when you use Auto Size, but there will be page breaks when you print the diagram.

Syntax

```
AUTOSIZE_MARGIN '<ON|OFF>'
```

Example

```
AUTOSIZE_MARGIN 'ON'
```

Corresponding UI Option for Design Entry HDL

File —Publish PDF — Setup — PDF — PageSetup — Paper Size

4

B Directives

This chapter lists the CPM directives that start with `B` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

B2F_OVERWRITE_CONSTRAINTS

The B2F_OVERWRITE_CONSTRAINTS directive is used in Allegro System Capture and Packager-XL.This directive controls how constraints are synchronized during the back to front flow.

If this directive is set to `ON`, all the constraints in the logic design are overwritten by the constraints propagated from the physical layout.

If this directive is set to `OFF`, constraints from the physical layout are merged into the logic design using the Changes Only mode. This means that only those constraints that have been modified in the physical layout since the last synchronization between the layout and the logic design are transferred from the layout to the logic design. Constraints that have been updated in the logic design since the last synchronization are not updated in the back to front flow. This allows users to capture constraints in the layout and the logic design concurrently with no loss of data.

If this directive is set in the `.cpm` file, the `OVERWRITE_CONSTRAINTS` directive is ignored during the back to front flow.

You can also lock the directive at the site level. This ensures that if changes are being made simultaneously in the layout and the logic design, then the logic design changes are not overwritten.

```
START_PKGRXL_CONTROL_SETTINGS

B2F_OVERWRITE_CONSTRAINTS 'LOCK'

END_PKGRXL_CONTROL_SETTINGS
```

Syntax

```
B2F_OVERWRITE_CONSTRAINTS 'ON'|'OFF'
```

Example

```
START_PKGRXL

B2F_OVERWRITE_CONSTRAINTS 'ON'

END_PKGRXL
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

BACKANNOTATE_FEEDBACK

This directive corresponds to the *Backannotate Packaging Properties to Schematic Canvas* check box in the Import Physical dialog.

When the value of this directive is set to '`YES`', the packaging properties of the design are backannotated to the schematic while running Import Physical (Feedback mode).

Syntax

```
BACKANNOTATE_FEEDBACK 'YES'|'NO'|'ON'|'OFF'|'1'|'0'
```

Example

```
BACKANNOTATE_FEEDBACK 'YES'
```

Corresponding UI Option for Design Entry HDL

File —Import Physical — Import Physical dialog — Backannotate Packaging Properties to Schematic Canvas check box

BACKANNOTATE_FORWARD

This directive corresponds to the *BackAnnotate Packaging Properties to Schematic Canvas* check box in the Export Physical dialog.

When the value of this directive is set to '`YES`', the packaging properties of the design are backannotated to the schematic while running Export Physical (Forward mode).

Syntax

```
BACKANNOTATE_FORWARD 'YES'|'NO'|'ON'|'OFF'|'1'|'0'
```

Example

```
BACKANNOTATE_FORWARD 'YES'
```

Corresponding UI Option for Design Entry HDL

File —Export Physical — Export Physical dialog — Backannotate Packaging Properties to Schematic Canvas check box

BACKGROUND_COLOR

Using this directive, you can change the default color for the drawing area in Allegro Design Entry HDL and in Allegro System Capture.

Syntax

```
BACKGROUND_COLOR '<color>'
```

where

color	white	black	gray
	dark_gray	yellow	red
	blue	magenta	cyan
	salmon	lime_green	brown
	peach	brick_red	pink
	red_violet	steel_blue	blue_violet
	blue_green	green_blue	violet
	violet_blue	violet_red	green
	yellow_green	mustard	yellow_orange
	orange	orange_red	orange_yellow
	red_orange		

Example

```
BACKGROUND_COLOR 'BLACK'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Background Color

See Also

- ARC_COLOR

- HIGHLIGHT_COLOR
- DOT_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

BASE_NET_OVERLAY

This directive controls the display of additional text on the base net names in a System Capture design. The suffix for the winning net name is called a text overlay and can be customized using the `EXPLICIT_BASE_NET_IDENTIFIER` directive.

Syntax

```
BASE_NET_OVERLAY 'ON'|'OFF'
```

Example

```
BASE_NET_OVERLAY 'ON''
```

Corresponding UI Option for Allegro System Capture

None

See Also

EXPLICIT_BASE_NET_IDENTIFIER

BASENET_OMIT_SYNONYM

Omits the synonym column in the base net report and schematic reports. Depending upon number of synonyms attached to a base signal, multiple rows appear in the base net report. the synonym columns are not displayed and all location values are listed in the same row as the base signal name. This formatting is especially beneficial in schematic reports as it saves space.

By default, the BASENET_OMIT_SYNONYM directive is set to OFF, which results in displaying of synonyms information corresponding to base signals.

Value

```
BASENET_OMIT_SYNONYM 'ON'|'OFF'
```

Example

```
BASENET_OMIT_SYNONYM 'ON'
```

Corresponding UI Option

None

BBOX_COLOR

Use this directive to specify the color of the bounding box in the generated document schematic.

Syntax

```
bbox_color <color>
```

The valid values are: Red, Blue, Green, Yellow, Orange, Salmon, Violet, Brown, Skyblue, White, Peach, Pink, Purple, Aqua, Gray, and Mono.

Example

```
bbox_color 'Pink'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Colors — Bounding Box Color

See Also

ADD_BBOX

BBOX_SHOWN_ON_FONT_MOVE

The Design Entry HDL environment, by design, provides support for vector fonts. You can use TrueType fonts to display and distinguish between different types of text objects in a design.

When you move text objects while using True Type fonts, the text is attached to the cursor and is displayed in a vector font. As the width of TrueType and vector fonts is different, it can sometimes be difficult to read.

Using this directive ensures that when moving the text, the bounding box displayed is according to the width of the True Type font. This provides you with an accurate idea of the text width and ensures that there is no overlap after the text is placed on the schematic.

Syntax

```
BBOX_SHOWN_ON_FONT_MOVE 'ON'|'OFF' 'TRUE'|'FALSE'
```

Example

```
BBOX_SHOWN_ON_FONT_MOVE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

BLACKANDWHITEPDF

Generates a black and white PDF with Data tool and JavaScript support. On printing, the PDF will print in black and white as well.

Syntax

```
BLACKANDWHITEPDF '0'|'1'
```

Example

```
BLACKANDWHITEPDF '0'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — Advanced — Publish in — Black&White

See Also

PRINTLAYER

BLOCK_DIAGRAM_CELL

Use this directive to specify the cell in which the schematic of the block diagram to be included in the generated document schematic is saved.

Note: This option works only if the add_block_diagram directive is set to 1 or TRUE.

Syntax

```
block_diagram_cell <cell name>
```

Example

```
block_diagram_cell 'ABLK'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — General — Include Block Diagram — Cell.

See Also

ADD_BLOCK_DIAGRAM

BLOCK_DIAGRAM_LIB

BLOCK_DIAGRAM_LIB

Use this directive to specify the library in which the schematic of the block diagram to be included in the generated document schematic is saved.

Note: This option works only if the add_block_diagram directive is set to 1 or TRUE.

Syntax

```
block_diagram_lib <library>
```

Example

```
block_diagram_lib 'block'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — General — Include Block Diagram — Library.

See Also

ADD_BLOCK_DIAGRAM

BLOCK_DIAGRAM_CELL

BLOCK_PIN_SHAPE_MINIMUMPINSPACING

Use this directive to specify the pin-to-pin spacing for hierarchical block symbols in Allegro System Capture. This directive value can only take integer values. The actual grid spacing is twice of the directive value set. For example, when `BLOCK_PIN_SHAPE_MINIMUMPINSPACING` is set to *2*, the actual pin-to-pin grid spacing is 4 for newly added pins after symbol regeneration.

Syntax

```
BLOCK_PIN_SHAPE_MINIMUMPINSPACING '<spacing distance>'
```

Example

```
BLOCK_PIN_SHAPE_MINIMUMPINSPACING '2'
```

Corresponding UI Option

None

BLOCK_REF_DES_PATTERN

Using this directive you can define custom pattern for reference designator in the *Packaging Options for Block* dialog when adding a block instance on the schematic.

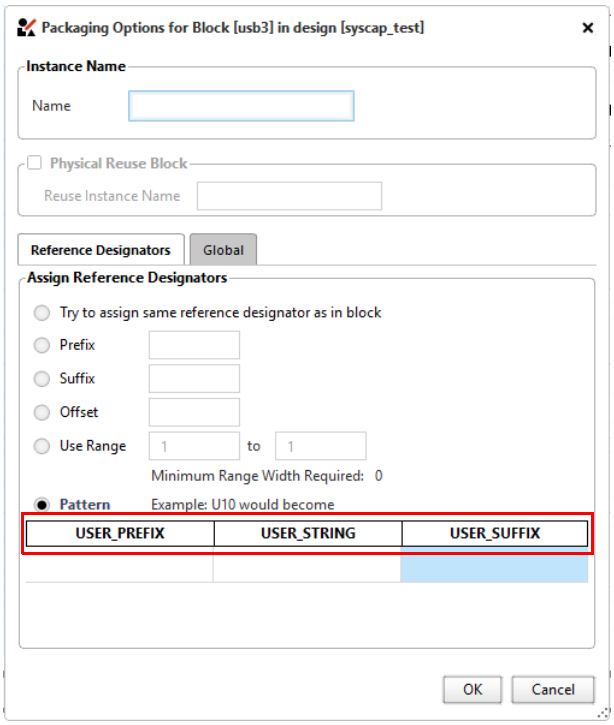
Syntax

```
BLOCK_REF_DES_PATTERN <'($pattern1)($pattern2)($pattern3)'>
```

Example

```
BLOCK_REF_DES_PATTERN '($USER_PREFIX)($USER_STRING)($USER_SUFFIX)'
```

If the above value is specified, the *Pattern* section of the *Packaging Options for Block* dialog appears as shown in the following figure:



If `USER_STRING` is specified as `2`, `USER_PREFIX` is `A_` and `USER_SUFFIX` is `_B`, the RefDes changes from `U10` to `A_U210_B`.

Corresponding UI Option

None

BLOCK_TITLE_TOP

When a new block is created, by default, Design Entry HDL places its default name at the top of the block then draws a line below it. Use this directive in the `START_CONCEPTHDL` section to stop the line under the block title section from being created when you create a new block. The directive will not apply to existing blocks and only works for new blocks.

When working with blocks generated by Genview, there are times when moving block pins or resizing a block by stretching it deletes the line below the block name. Other issues might occur such as the line separating the title section from the rest of the block shorting two pins. For such issues, this directive can be used as a workaround.

Syntax

```
BLOCK_TITLE_TOP 'ON'|'OFF'
```

Example

```
BLOCK_TITLE_TOP 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

BOM_FOLDER

Using this directive, you can specify the location where you want the BOM files to be generated.

Syntax

```
BOM_FOLDER '<directory>'
```

where

<directory>	Specify the path where you want the BOM files to be generated.

Example

```
BOM_FOLDER 'C:\Designs'
```

Corresponding UI Option

None

BUS_TAP_SYMBOL_ROT<angle>

Controls the rotation of the given bus tap symbol by a specified angle.

Syntax

```
BUS_TAP_SYMBOL_ROT<angle> '<symbol_name>'
```

where,

angle	Indicates the angle by which the given bus tap symbol is rotated. The valid values are 90, 180, 270.
symbol name	Indicates the bus tap symbol.

Example

```
BUS_TAP_SYMBOL_ROT90 'standard.tap:sym_7'
BUS_TAP_SYMBOL_ROT180 'standard.tap:sym_1'
BUS_TAP_SYMBOL_ROT270 'standard.tap:sym_3'
```

Corresponding UI Option

None

5

C Directives

This chapter lists the CPM directives that start with `C` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

CAPSLOCK

Displays text in capital letters in the schematic.

Syntax

```
CAPSLOCK 'ON'|'OFF'
```

Example

```
CAPSLOCK 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Text — Upper-case Input

CAPTURE_CACHE_LIBRARY_PATH

Specify the path of the folder that contains the default parts to be used for the OrCAD Capture Library flow. These OrCAD Capture parts are converted to DE-HDL library format.

Syntax

```
CAPTURE_CACHE_LIBRARY_PATH '<path>'
```

where,

path	Specify the path the folder that contains parts.

Example
```
CAPTURE_CACHE_LIBRARY_PATH '$CONCEPT_INST_DIR/share/cdssetup/canvas/
resources/capture/library/orcadlib
```

CAPTURE_STANDARD_LIB

Specify the name of the library that contains standard parts to be used for the OrCAD Capture Library flow.

Syntax

```
CAPTURE_STANDARD_LIB '<library_name>'
```

where,

`library_name`	Specify the name of the library.

Example

```
CAPTURE_STANDARD_LIB 'capsym'
```

CAS_PIN_PATTERN

Defines the patterns used for identifying pins as Column Address Strobe (CAS) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `RAS and CAS pins of ICs incorrectly connected to non-RAS or non-CAS pins`
- `RAS and CAS pins of ICs not connected to the same set of ICs or connectors`
- `RAS and CAS pins of IC incorrectly connected`
- `Unconnected RAS and CAS pins of ICs`

Syntax

```
CAS_PIN_PATTERN '<pin pattern>'
```

Example

```
CAS_PIN_PATTERN 'CAS'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *CAS Pin Patterns*

CATPATH

Specifies the directory that contains the category (.cat) files used to organize components by category.

Syntax

```
CATPATH '<path>'
```

where

path	is the path to the directory containing the category file.

Example

```
CATPATH 'D:/PROJECT/TEMP/cat_file'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Paths page — Input Paths — Category Path

See Also

- ATTRIBUTES_DIR
- INPUT_SCRIPT
- PPT_OPTIONSET_PATH

CDS_<CATEGORY_NAME>_FONT

Sets the value of the font name to be used for a specific category of text objects.

Syntax

```
CDS_<CATEGORY_NAME>_FONT '<font_name>'
```

where

CATEGORY_NAME is the schematic text object for which you want to specify the font name. You can set the font attributes for the following categories:

- SYMBOLTEXT
- SYMBOL
- REFDES
- NETNAME
- NETPROP
- CROSSREF
- PINNAME
- PINTEXT
- PINPROP
- CUSTOMTEXT
- TEXTNOTES

font_name is the font you want to use to display a specific category of text objects. For example, you can specify Courier font to display all the net names in the design. You can specify any font installed on the local system.

Example

```
CDS_SYMBOLTEXT_FONT 'BOOK ANTIQUA'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Name

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONTSIZE
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTCOLOR
- CDS_<CATEGORY_NAME>_FONTEFFECTS

CDS_<CATEGORY_NAME>_FONTCOLOR

Sets the value of the font color to be used for a specific category of text objects.

Syntax

```
CDS_<CATEGORY_NAME>_FONTCOLOR '<font_color>'
```

where

CATEGORY_NAME is the schematic text object for which you want to specify the font color. You can set the font color for the following categories:

- SYMBOLTEXT
- SYMBOL
- REFDES
- NETNAME
- NETPROP
- CROSSREF
- PINNAME
- PINTEXT
- PINPROP
- CUSTOMTEXT
- TEXTNOTES

font_color is the color in which all the newly-created text objects of this category are to be displayed. You can specify a color from the 16 basic colors: Black, Dark Red, Dark Green, Dark Yellow, Dark Blue, Dark Purple, Dark Cyan, Pale Gray, Mid Gray, Red, Green, Yellow, Blue, Magenta, Cyan, and White.

Note: The text color, like text size, is currently stored in the database. Therefore, you can specify the font color for individual objects. All the objects which are already on the canvas have a font color specified on them and the same font color is honored.

Example

```
CDS_TEXTNOTES_FONTCOLOR 'RED'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Color

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONT
- CDS_<CATEGORY_NAME>_FONTSIZE
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTEFFECTS

CDS_<CATEGORY_NAME>_FONTEFFECTS

Sets the value of the effect to *Underline* effect to display the text as underlined.

Syntax

```
CDS_<CATEGORY_NAME>_FONTEFFECTS '<font_effect>'
```

where

CATEGORY_NAME is the schematic text object for which you want to specify the font style. You can set the font effects for the following categories:

- SYMBOLTEXT
- SYMBOL
- REFDES
- NETNAME
- NETPROP
- CROSSREF
- PINNAME
- PINTEXT
- PINPROP
- CUSTOMTEXT
- TEXTNOTES

font_effect sets the Underline effect to display the text as underlined. By default, all text objects display regular text.

Example

```
CDS_CROSSREF_EFFECTS 'Underline'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Effects

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONT
- CDS_<CATEGORY_NAME>_FONTSIZE
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTCOLOR

CDS_<CATEGORY_NAME>_FONTSIZE

Sets the value of the font size to be used for a specific category of text objects.

Syntax

```
CDS_<CATEGORY_NAME>_FONTSIZE '<font_size>'
```

where

CATEGORY_NAME is the schematic text object for which you want to specify the font size. You can set the font size for the following categories:

- SYMBOLTEXT
- SYMBOL
- REFDES
- NETNAME
- NETPROP
- CROSSREF
- PINNAME
- PINTEXT
- PINPROP
- CUSTOMTEXT
- TEXTNOTES

font_size is the font size with which all the newly added text objects for the category are to be displayed. This size is also known as point size, where one point size equals 1/72 of an inch.

Note: The text size is currently stored in the database. Therefore, you can specify the font size for individual objects. All the objects which are already on the canvas have a font size specified on them and the same font size is honored.

Example

```
CDS_REFDES_FONTSIZE '8'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Size

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONT
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTCOLOR
- CDS_<CATEGORY_NAME>_FONTEFFECTS

CDS_<CATEGORY_NAME>_FONTSTYLE

Sets the value of the font style to be used for a specific category of text objects.

Syntax

```
CDS_<CATEGORY_NAME>_FONTSTYLE '<font_style>'
```

where

CATEGORY_NAME is the schematic text object for which you want to specify the font style. You can set the font style for the following categories:

- SYMBOLTEXT
- SYMBOL
- REFDES
- NETNAME
- NETPROP
- CROSSREF
- PINNAME
- PINTEXT
- PINPROP
- CUSTOMTEXT
- TEXTNOTES

font_style is the font style with which all the text objects for the category are to be displayed. You can specify one of the four font styles: Regular, **Bold**, ***Bold Italic***, and *Italic*. All fonts do not support all the styles. Therefore, you can specify only those styles which are supported for a specific font. For example, you can specify all the four styles for the Arial font, while only Regular style is supported for Arial Black.

Example

```
CDS_NETNAME_FONTSTYLE 'Bold Italic'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Style

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONT
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTCOLOR
- CDS_<CATEGORY_NAME>_FONTEFFECTS

CDS_ENABLE_FONTS

Enables support for fonts in Design Entry HDL.

Syntax

```
CDS_ENABLE_FONTS 'ON'|'OFF'
```

Example

```
CDS_ENABLE_FONTS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Font page — Enable Font Support check box

See Also

- CDS_ENABLE_FONTS
- CDS_<CATEGORY_NAME>_FONTSIZE
- CDS_<CATEGORY_NAME>_FONTSTYLE
- CDS_<CATEGORY_NAME>_FONTCOLOR
- CDS_<CATEGORY_NAME>_FONTEFFECTS

CENTRAL_INDEX_PATH

Allegro System Capture and Allegro EDM work with Part Information Manager, which provides quick search capabilities for library parts using an indexed database. If you work with very large libraries frequently, you might want to be able to access libraries more quickly using the indexed database. This database can be at the project or central library level.

The `central_index_path` directive provides the location of the indexed database. If the directive is not set, Part Information Manager looks for the indexed database in the project itself. The directive needs to be in the `START_COMPBROWSER...END_COMPBROWSER` section, and can be configured at the site level in the `site.cpm` file.

The indexed database can be created for all the libraries included in the `cds.lib` file by doing the following:

1. Set the directive in the `site.cpm` file.
2. Make sure all your libraries are included.
3. In a command prompt, run the application indexer to create the indexed database using `cds.lib` as input. The application indexer is available at:

   `<Cadence installation hierarchy>/tools/pcbdw/bin/indexer`

This will generate or update an indexed database of all the libraries in the `cds.lib` file at the location defined in `central_index_path`.

When you now launch Part Information Manager from Allegro System Capture, libraries will be read from the indexed database defined in `central_index_path`.

If you add or remove libraries by manually editing `$CDS_SITE/cdssetup/cds.lib` or through the user interface (*Edit – Project Preferences*), you will need to re-index the database by running `indexer.bat` again. This can be configured as a Cron job or manually run whenever libraries are updated. Changes will be automatically reflected in Part Information Manager in Allegro System Capture designs that read the indexed database.

Memory Consumption

Scenario	Memory Consumption
System Capture with CENTRAL_INDEX_PATH	Low

System Capture indexes customer libraries for each project on each designer's machine	Increases with library size
If you are using System Capture with CENTRAL_INDEX_PATH + Allegro EDM server	Low In this case, because System Capture fetches only required data from the EDM server and the designer's machine has no library data loaded on it, this consumes the least memory.

Syntax

```
central_index_path '<path to indexed database>'
```

Example

```
central_index_path '${CDS_SITE}/central_indexed_database'
```

```
central_index_path 'D:/central_indexed_database'
```

Corresponding UI Option

None

CHECK_ARCS_AT_SAME_LOCATION

Checks for overlaid arcs.

Syntax

```
CHECK_ARCS_AT_SAME_LOCATION 'ON'|'OFF'
```

Example

```
CHECK_ARCS_AT_SAME_LOCATION 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Check— Arcs At Same Location

See Also

- CHECK_GRID_ON_TAP
- CHECK_IMAGE_OVERLAP_OBJECT

CHECK_GLOBAL_LOCAL_SHORT

Checks for local signals connected to power symbols whose names are different from the value of the HDL_POWER property of the power symbol.

Syntax

```
CHECK_GLOBAL_LOCAL_SHORT 'ON'|'OFF'
```

Example

```
CHECK_GLOBAL_LOCAL_SHORT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Electrical Checks — Power-Local Signal Short

See Also

- CHECK_MISSING_PINS
- CHECK_UNCONNECTED_WIRES
- CHECK_VOLTAGE_ON_HDL

CHECK_GRID_ON_TAP

Checks for any bus tap error that occurs if the end coordinates are located at the end pin grid on the ctap symbol. The directive is set to 'ON' by default.

Syntax

```
CHECK_GRID_ON_TAP
```

Example

```
CHECK_GRID_ON_TAP
```

Corresponding UI Option for Allegro Design Entry HDL

None

CHECK_HIDDEN_WIRES

Checks for wire segments hidden by portions of components.

Syntax

```
CHECK_HIDDEN_WIRES 'ON'|'OFF'
```

Example

```
CHECK_HIDDEN_WIRES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Check — Hidden Wires

See Also

- CHECK_ARCS_AT_SAME_LOCATION
- CHECK_IMAGE_OVERLAP_OBJECT

CHECK_IMAGE_OVERLAP_OBJECT

Checks whether the images pasted on the schematic overlap any schematic component. If found, an error is flagged: *ERROR (SPCOCN-2023): Image overlapping with objects*.

Syntax

```
CHECK_IMAGE_OVERLAP_OBJECT 'ON'|'OFF'
```

Example

```
CHECK_IMAGE_OVERLAP_OBJECT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Checks — Image Overlapping Object

See Also

- CHECK_ARCS_AT_SAME_LOCATION
- CHECK_GRID_ON_TAP

check_injected_order

Verifies (when set to `TRUE`) whether the order of any of the injected property headers has changed and indicates the change in LRM:

Status
Injected Property Order Mismatch
Injected Property Order Mismatch
Not in Reference
Injected Property Order Mismatch

Syntax

```
check_injected_order 'TRUE' | 'FALSE'
```

Example

```
check_injected_order 'TRUE'
```

Corresponding UI Option

None

check_local_modified

When set to `TRUE`, this directive verifies whether any cell has been modified directly on the disk without using Library Revision Manager (LRM). The default is `TRUE`.

In LRM, the rows in the *Cell/Block data for design* `<design name>` table report what is out of sync in the design. When the status of a component is *Incorrect Metadata,* it indicates that the cell has been modified manually (possibly without using Part Developer) resulting in inconsistencies in the cell metadata. If you often modify cells manually, you might want to set this directive to `FALSE`.

Updating the cell using LRM will replace the modified cell in the cache with the reference library cell.

Note: Even after an LRM update, the cell will be marked as *Incorrect Metadata* when you launch LRM again. It is recommended that instead of making changes manually, the librarian should modify and distribute the updated cells to all sites.

Syntax

```
check_local_modified 'TRUE' | 'FALSE'
```

Example

```
check_local_modified 'TRUE'
```

Corresponding UI Option

None

CHECK_MISSING_PINS

Checks for pin properties that are no longer attached to pins.

Syntax

```
CHECK_MISSING_PINS 'ON'|'OFF'
```

Example

```
CHECK_MISSING_PINS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Electrical Checks — Missing Pins

See Also

- CHECK_GLOBAL_LOCAL_SHORT
- CHECK_UNCONNECTED_WIRES

CHECK_MOVE_SHORT

Setting this directive to ON warns you about a change in connectivity when nets are shorted while moving components and nets in a design.

Syntax

```
CHECK_MOVE_SHORT 'ON'|'OFF'
```

Example

```
CHECK_MOVE_SHORT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Online Checks — Wire Short Check during move

CHECK_MULTIPLE_IBD_PROPS

When this directive is ON, DE-HDL checks for multiple net group properties on a net group object.

Syntax

```
CHECK_MULTIPLE_IBD_PROPS 'ON'|'OFF'
```

Example

```
CHECK_MULTIPLE_IBD_PROPS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

CHECK_NONSYNCPROPS

Checks for the presence of non-synchronous properties in the design.

Syntax

```
CHECK_NONSYNCPROPS 'ON'|'OFF'
```

Example

```
CHECK_NONSYNCPROPS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

CHECK_ON_WRITE

Runs a check whenever you save a design. Errors are recorded in the `cp.mkr` and `netlister.mkr` marker files.

Syntax

```
CHECK_ON_WRITE 'ON'|'OFF'
```

Example

```
CHECK_ON_WRITE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Check On Write

CHECK_PACK_SEC_TYPE_PROPS

Checks for multiple SEC-type properties on an instance.

Syntax

```
CHECK_PACK_SEC_TYPE_PROPS 'ON'|'OFF'
```

Example

```
CHECK_PACK_SEC_TYPE_PROPS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Misc. Checks — Multipackage Sections

See Also

CHECK_PAGE_NUMBER_SYNCH

CHECK_PACK_SYNC_ON_IMPORT

When importing a block with this directive set to ON, DE-HDL checks if the packaged data and the design connectivity of the block in the source design are in sync. All designs marked for physical or logical reuse are checked for blocks that are modified but not packaged.

When importing blocks whose package data is not in sync with the design connectivity, a warning message similar to the following is displayed:

```
The packaging of reuse blocks 'MEMORY1' selected for import has not
been updated and is out of sync with the source design connectivity.
These blocks should be packaged at source to update the reuse block
packaging.

Do you still want to continue to import these blocks?
```

If you click *Yes*, DE-HDL continues importing the block.

Syntax

```
CHECK_PACK_SYNC_ON_IMPORT 'ON'|'OFF'
```

Example

```
CHECK_PACK_SYNC_ON_IMPORT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

CHECK_PAGE_NUMBER_SYNCH

Checks and corrects the PAGE_NUMBER directive conflicts in the ASCII (`.csa`) and binary files (`.csb`) for all the pages of the design.

Syntax

```
CHECK_PAGE_NUMBER_SYNCH 'ON'|'OFF'
```

Example

```
CHECK_PAGE_NUMBER_SYNCH 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Misc. Checks — Page Number Mismatch

See Also

CHECK_PACK_SEC_TYPE_PROPS

CHECK_PIN_WIRE_DIST_THRESH

Specifies the threshold value for wires that do not quite contact pins. Design Entry HDL generates an error message if the distance between a wire end and a pin falls below a minimum distance called the threshold. The threshold is calculated based on an internal algorithm. This value is either 10 Design Entry HDL coordinates or higher based on the grid size.

Note: This directive can only work if the value of the CHECK_PINS_NEAR_WIRE_ENDPT directive is set to *ON*.

Syntax

```
CHECK_PIN_WIRE_DIST_THRESH '<value>'
```

where

value	is the threshold value for the distance between pin and wire

Example

```
CHECK_PIN_WIRE_DIST_THRESH '10'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Pin Near Wire End — Threshold Value

See Also

CHECK_PINS_NEAR_WIRE_ENDPT

CHECK_PINS_AT_ORIGIN

Checks for pins at the origin (0,0) in symbol drawings.

Syntax

```
CHECK_PINS_AT_ORIGIN 'ON'|'OFF'
```

Example

```
CHECK_PINS_AT_ORIGIN 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Checks — Pins At Origin

See Also

- CHECK_SYMBOLS_AT_SAME_LOCATION
- CHECK_TWO_WIRES_AT_PINS

CHECK_PINS_NEAR_WIRE_ENDPT

Checks for wires that do not quite contact pins.

Syntax

```
CHECK_PINS_NEAR_WIRE_ENDPT 'ON'|'OFF'
```

Example

```
CHECK_PINS_NEAR_WIRE_ENDPT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Pin Near Wire End

See Also

CHECK_PIN_WIRE_DIST_THRESH

CHECK_PROP_PLACE_HOLDERS

Checks for placeholder properties that appear due to changes in the related library.

Syntax

```
CHECK_PROP_PLACE_HOLDERS 'ON'|'OFF'
```

Example

```
CHECK_PROP_PLACE_HOLDERS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Misc. Checks — Property Place Holders

See Also

CHECK_SYMBOL_PLACE_HOLDERS

CHECK_SHORTED_PINS

Checks for two pins on one component that are connected to the same wire, i.e. two pins shorted together.

Syntax

```
CHECK_SHORTED_PINS 'ON'|'OFF'
```

Example

```
CHECK_SHORTED_PINS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Electrical Checks — Shorted Pins

CHECK_SIGNAL_NAMES

Checks for multiple names attached to the same signal.

Syntax

```
CHECK_SIGNAL_NAMES 'ON'|'OFF'
```

Example

```
CHECK_SIGNAL_NAMES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Name Checks — Signal Names

See Also

CHECK_SYMBOL_SIGNAL_NAMES

CHECK_SYMBOL_PLACE_HOLDERS

Checks for placeholder components that appear due to changes in the related library.

Syntax

```
CHECK_SYMBOL_PLACE_HOLDERS 'ON'|'OFF'
```

Example

```
CHECK_SYMBOL_PLACE_HOLDERS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Misc. Checks — Symbol Place Holders

See Also

CHECK_PROP_PLACE_HOLDERS

CHECK_SYMBOL_SIGNAL_NAMES

Checks for the SIG_NAME property on a pin in a symbol file.

Syntax

```
CHECK_SYMBOL_SIGNAL_NAMES 'ON'|'OFF'
```

Example

```
CHECK_SYMBOL_SIGNAL_NAMES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Name Checks — Signal Name in Symbols

See Also

CHECK_SYMBOL_SIGNAL_NAMES

CHECK_SYMBOLS_AT_SAME_LOCATION

Checks for overlaid components.

Syntax

```
CHECK_SYMBOLS_AT_SAME_LOCATION 'ON'|'OFF'
```

Example

```
CHECK_SYMBOLS_AT_SAME_LOCATION 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Checks — Symbols at Same Location

See Also

- CHECK_PINS_AT_ORIGIN
- CHECK_TWO_WIRES_AT_PINS

CHECK_TWO_WIRES_AT_PINS

Checks for wires overlapping a component at the pin.

Syntax

```
CHECK_TWO_WIRES_AT_PINS 'ON'|'OFF'
```

Example

```
CHECK_TWO_WIRES_AT_PINS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Graphics Checks — Two Wires At Pins

See Also

- CHECK_PINS_AT_ORIGIN
- CHECK_SYMBOLS_AT_SAME_LOCATION

CHECK_UNCONNECTED_WIRES

Checks for unnamed wires connected to only one pin (NC wires) and for named nets not connected to any pins.

Syntax

```
CHECK_UNCONNECTED_WIRES 'ON'|'OFF'
```

Example

```
CHECK_UNCONNECTED_WIRES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Electrical Checks — Unconnected Wires

See Also

- CHECK_GLOBAL_LOCAL_SHORT
- CHECK_VOLTAGE_ON_HDL
- CHECK_MISSING_PINS

CHECK_VOLTAGE_ON_HDL

Checks for the presence of the VOLTAGE property on an HDL_POWER symbol. If the VOLTAGE property is not present, a warning message is displayed.

Syntax

```
CHECK_VOLTAGE_ON_HDL 'ON'|'OFF'
```

Example

```
CHECK_VOLTAGE_ON_HDL 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Check page — Electrical Checks — Voltage on HDL Symbols

See Also

- CHECK_GLOBAL_LOCAL_SHORT
- CHECK_MISSING_PINS
- CHECK_UNCONNECTED_WIRES

CLOCK_PIN_PATTERN

Defines the patterns used for identifying pins as CLOCK pins.

This directive is used when the `Unconnected Clock pins of ICs when MISO or MOSI pins are connected` audit rule is run.

Syntax

```
CLOCK_PIN_PATTERN '<pin pattern>'
```

Example

```
CLOCK_PIN_PATTERN 'CLK' 'CLOCK' 'CK'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *CLOCK Pin Patterns*

CLICK_TO_TYPE

Activates a window when you click in it. Otherwise, a window is activated when you move the cursor into it.

Syntax

```
CLICK_TO_TYPE 'ON'|'OFF'
```

Example

```
CLICK_TO_TYPE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Click To Activate View

CNAME_CELL

Allows you to display the cell name in the canonical name in the Global Find, Global Navigation and Attributes dialog boxes.

Syntax

```
CNAME_CELL 'ON'|'OFF'
```

Example

```
CNAME_CELL 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Canonical Names — Cell

See Also

- CNAME_DEPTH
- CNAME_LIB
- CNAME_VIEW

CNAME_DEPTH

Specifies the levels of Lib.Cell:View that is shown in a canonical name.

Syntax

```
CNAME_CELL '<level>'
```

where

level is the levels of Lib.Cell:View.

Example

```
CNAME_CELL '0'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Canonical Names — Depth

See Also

- CNAME_CELL
- CNAME_LIB
- CNAME_VIEW

CNAME_LIB

Allows you to display the library name in the canonical name in the Global Find, Global Navigation and Attributes dialog boxes.

Syntax

```
CNAME_LIB 'ON'|'OFF'
```

Example

```
CNAME_LIB 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Canonical Names — Library

See Also

- CNAME_DEPTH
- CNAME_CELL
- CNAME_VIEW

CNAME_VIEW

Allows you to display the view name in the canonical name in the Global Find, Global Navigation and Attributes dialog boxes.

Syntax

```
CNAME_VIEW 'ON'|'OFF'
```

Example

```
CNAME_VIEW 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Canonical Names — Library

See Also

- CNAME_DEPTH
- CNAME_CELL
- CNAME_LIB

COMBINED_MARKERS_ON_HIERWRITE

When you run the save hierarchy command, by default, DE HDL generates a single marker file listing all existing errors in your design blocks. If you want DE HDL to generate a marker file for each block in your design when you perform a save hierarchy operation, set this directive to OFF.

Syntax

```
COMBINED_MARKERS_ON_HIERWRITE 'ON'|'OFF'
```

Example

```
COMBINED_MARKERS_ON_HIERWRITE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

COMMENT_COLOR

Use this directive to specify the color of the comment text that is added to the generated document schematic

Syntax

```
comment_color <Blue>
```

The valid values are: Red, Blue, Green, Yellow, Orange, Salmon, Violet, Brown, Skyblue, White, Peach, Pink, Purple, Aqua, Gray, and Mono.

Example

```
comment_color 'Blue'
```

Corresponding UI Option

Project — Settings — Document Schematic Generation — Colors — Comments Color

See Also

ADD_COMMENTS

BBOX_COLOR

COMP_COLOR

PROP_COLOR

WIRE_COLOR

COMP_COLOR

Use this directive to specify the color used to draw component instances in the generated document schematic.

Syntax

```
comp_color <color>
```

The valid values are: Red, Blue, Green, Yellow, Orange, Salmon, Violet, Brown, Skyblue, White, Peach, Pink, Purple, Aqua, Gray, and Mono.

Example

```
comp_color 'Green'
```

Corresponding UI Option

Project — Settings — Document Schematic Generation — Colors — Symbol Color

See Also

BBOX_COLOR

COMMENT_COLOR

PROP_COLOR

WIRE_COLOR

COMP_DEF_PROP

The COMP_DEF_PROP Directive is used in Allegro System Capture and Packager-XL. This directive specifies the names of properties to be treated as component definition properties. Allegro System Capture and Packager-XL use component definition properties to create alternate physical parts.

The REF_DES_LENGTH directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
COMP_DEF_PROPERTY property [, property ] ... ;
```

property	represents a component definition property.

The default properties are *JEDEC_TYPE, ALT_SYMBOLS, MERGE_NC_PINS, MERGE_POWER_PINS, NC_PINS, POWER_GROUP, POWER_PINS,* and *PINCOUNT*.

Using the COMP_DEF_PROP directive, overrides the default component definition properties. Therefore, you should ensure that you include the default property names when you add additional properties.

Example

If you specify the MYPROP directive as a COMP_DEF_PROP directive in your Packager-XL project file and if you have an instance of a 74LS00 in your design with the attached property, MYPROP = ALT2, a new entry is generated in the `pstchip.dat` file as follows:

```
74LS00-ALT2
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

COMP_INST_PROP

The COMP_INST_PROP directive specifies the names of component instance properties attached to the schematic instances in Allegro System Capture and Packager-XL.

The COMP_INST_PROP directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
COMP_INST_PROP property [, property ] ...;
```

where

property	is a component instance property.

The default value for the COMP_INST_PROP directive is none.

Caution

Specify ROOM as a component instance property.

Example

```
COMP_INST_PROP ROOM;
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

COMPONENT_BROWSER

Controls the display of Part Information Manager to add or replace components from available libraries. When set to `OFF`, Part Information Manager is no longer displayed on choosing *Component – Add* or *Component – Replace* menu commands. Instead, a warning message is flagged, prompting you to select the component to add.

Syntax

```
COMPONENT_BROWSER 'ON'|'OFF'
```

Example

```
COMPONENT_BROWSER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Component Browser (Add)

See Also

- DRAWING_BROWSER
- LIBRARY_BROWSER

ConceptSetup_Assertion_Read

The `ConceptSetup_Assertion_Read` directive determines the pin name suffix, either * or _N, that should be used in addition to the suffix specified through the `ConceptSetup_Assertion_Write '*' 'Suffix'` directive to read low-asserted pins.

Syntax

```
ConceptSetup_Assertion_Read '<assertion_character>' 'Suffix'
```

The following assertion characters are supported:

- *
- _N

Example

```
ConceptSetup_Assertion_Read '_N' 'Suffix'
```

Corresponding UI Option for Part Developer

Additional Read option on *Tools – Setup*

See Also

ConceptSetup_Assertion_Write

ConceptSetup_Assertion_UseMinusInChips

The `ConceptSetup_Assertion_UseMinusInChips` directive determines if the low-assertion character in a pin name is to be replaced with - in `chips.prt`. This is the default behavior. To indicate that the low-assertion character is not to be replaced with - in `chips.prt`, specify:

```
ConceptSetup_Assertion_UseMinusInChips ''
```

Syntax

```
ConceptSetup_Assertion_UseMinusInChips '-|'
```

Example

```
ConceptSetup_Assertion_UseMinusInChips '-'
```

Corresponding UI Option for Part Developer

Use minus [-] sign for low-asserted pins in Package view check box on *Tools – Setup*

ConceptSetup_Assertion_Write

The `ConceptSetup_Assertion_Write` directive determines whether the `_N` or `*` suffix in pin names should be used to treat pins as low-asserted when reading or saving a part.

Syntax

```
ConceptSetup_Assertion_Write '<assertion_character>' 'Suffix'
```

The following assertion characters are supported:

- *
- _N

Example

```
ConceptSetup_Assertion_Write '*' 'Suffix'
```

Corresponding UI Option for Part Developer

Read/Write option on *Tools – Setup*

See Also

ConceptSetup_Assertion_Write

ConceptSetup_SplitPart_AddSwapInfo

The `ConceptSetup_SplitPart_AddSwapInfo` directive determines if the `SWAP_INFO` property is to be added to the `chips.prt` file. Part Developer determines the value of the `SWAP_INFO` property from the `SPLIT_INST_GROUP` information.

Syntax

```
ConceptSetup_SplitPart_AddSwapInfo '0'|'1'
```

Example

```
ConceptSetup_SplitPart_AddSwapInfo '0'
```

Corresponding UI Option for Part Developer

Auto Add SWAP_INFO to Chips check box on *Tools – Setup*

ConceptSetup_SplitPart_SymbolProp

The `ConceptSetup_SplitPart_SymbolProp` directive determines if split symbols are to be assigned `SPLIT_INST` and `$LOCATION` properties or the `SPLIT_INST_NAME` property.

The following values are supported:

- 0

  Assigns `SPLIT_INST` and `$LOCATION` properties

- 1

  Assigns the `SPLIT_INST_NAME` property

Syntax

```
ConceptSetup_SplitPart_SymbolProp '0'|'1'
```

Example

```
ConceptSetup_SplitPart_SymbolProp '0'
```

Corresponding UI Option for Part Developer

None

CONFIRM_WRITE

Provides confirmation about saving the drawing.

Syntax

```
CONFIRM_WRITE 'ON'|'OFF'
```

Example

```
CONFIRM_WRITE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write — Confirm Write

CONNECTION_SWAP_PINS

Use this directive to enable connection swaps when pin swaps are performed.

Syntax

```
connection_swap_pins 'ON|OFF'
```

Example

```
connection_swap_pins 'ON'
```

Corresponding UI Option for System Connectivity Manager

None

CP_NO_OF_COL_TOC

Controls the default number of columns displayed in the TOC of a design. You can decide on the number of columns which can be accommodated on the TOC page based on the size of the page.

Syntax

```
CP_NO_OF_COL_TOC '<number_of_columns>'
```

Example

```
CP_NO_OF_COL_TOC '3'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Table of Contents - Column Count

See Also

- DEFAULT_PHYS_DES_PREFIX
- CP_TOC_COL_<column_number>
- CP_TOC_COL_<column_number>_LABEL

CP_NO_OF_ROW_TOC

Controls the default number of rows displayed in the Table of Contents (TOC) of a design. You can decide on the number of rows which can be accommodated on the TOC page based on the size of the page.

Syntax

```
CP_NO_OF_ROW_TOC '<number_of_rows>'
```

Example

```
CP_NO_OF_ROW_TOC '10'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Table of Contents - Row Count

CP_TOC_COL_<column_number>

Controls the property to be displayed in a specified column of the TOC. This can be extended to support more columns with user-defined page properties.

Syntax

```
CP_TOC_COL_<columns number> '<property_name>'
```

Example

```
CP_TOC_COL_1 'SHEET_NO'
CP_TOC_COL_2 'SHEET_NAME'
CP_TOC_COL_3 'BLOCK_NAME'
```

Corresponding UI Option

None

See Also

- DEFAULT_PHYS_DES_PREFIX
- CP_NO_OF_COL_TOC
- CP_TOC_COL_<column_number>_LABEL

CP_TOC_COL_<column_number>_LABEL

Displays the column header for a specified column in the TOC.

Syntax

```
CP_TOC_COL_<column_number>_LABEL 'column_header'
```

Example

```
CP_TOC_COL_1_LABEL 'Sheet No.'
CP_TOC_COL_2_LABEL 'Sheet Name'
CP_TOC_COL_3_LABEL 'BLOCK NAME'
```

Corresponding UI Option

None

See Also

- DEFAULT_PHYS_DES_PREFIX
- CP_NO_OF_COL_TOC
- CP_TOC_COL_<column_number>

CREATE_CACHE_PROJECT

Controls the creation of a cache containing all the components used in the project. When a cache is created for a project, Part Manager can be used to compare the parts in use with their versions in the reference libraries.

Syntax

```
CREATE_CACHE_PROJECT='TRUE'|'FALSE'
```

where,

TRUE	Enables the cache-mode
FALSE	Disables the cache-mode.

Example

```
CREATE_CACHE_PROJECT='TRUE'
```

CREATE_USER_PROP

The *Create user-defined properties* check box in the Export Physical dialog is selected when this directive value is set to 'ON'.

User properties are added automatically to the board when you run the export physical command. When you delete such a property in Design Entry HDL, it is automatically deleted from the PCB Editor board.

Syntax

```
create_user_prop 'YES'|'NO'|'ON'|'OFF'|'1'|'0'
```

Example

```
create_user_prop 'NO'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Create user-defined properties check box

CREF_DATA_FILE

The CREF_DATA_FILE directive defines the location of the cref data file. If this directive is not specified, System Capture uses the CSF search to find the cref data file.

Note: This directive is specified in the `START_CUSTOMVAR` section of the project's `.cpm` file.

Syntax

```
CREF_DATA_FILE '<path to the cref data file>'
```

Example

```
CREF_DATA_FILE 'C:/Cadence/SPB_17.4/share/cdssetup/creferhdl/cref.dat'
```

Corresponding UI Option for Allegro System Capture

Edit – Preferences – Project Preferences – Custom Variables

CS_PIN_PATTERN

Defines the patterns used for identifying pins as *Chip Select*.

This directive is used when the `Unconnected Chip Select pins of ICs when MISO or MOSI pins are connected` audit rule is run.

Syntax

```
CS_PIN_PATTERN '<pin pattern>'
```

Example

```
CS_PIN_PATTERN 'CS' 'CS*'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *CS Pin Patterns*

CTAP

The CTAP directive is used in Allegro System Capture and Schgen. This directive is used to specify the *lib:cell:view* of the symbol as tap body (for extracting a bit from the bus) during the document schematic generation process.

The CTAP directive is specified in the `START_DSSCHGEN...END_DSSCHGEN` and `START_CANVAS...END_CANVAS` sections. System Connectivity Manager reads the directive value from the `START_DSSCHGEN...END_DSSCHGEN` section and Allegro System Capture reads the directive value from the `START_CANVAS...END_CANVAS` section.

Syntax

```
ctap <library.cell:view>
```

Example

```
ctap 'standard.ctap:sym_1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Ctap

Corresponding UI Option for Allegro System Capture

None

CTRLLMB_DRAGSELECT

Changes the behavior of the select and drag mouse operation and for running commands with strokes.

Note: For more information, refer to *Design Entry HDL Options — General — Ctrl+LMB Select and Drag*.

Syntax

```
CTRLLMB_DRAGSELECT 'ON'|'OFF'
```

Example

```
CTRLLMB_DRAGSELECT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Ctrl+LMB Select and Drag

CTRLRMB_CONTEXTMENU

Changes the behavior of the right mouse button (RMB).

Syntax

```
CTRLRMB_CONTEXTMENU 'ON'|'OFF'
```

where

OFF	If the directive is set to off, clicking the right mouse button displays the context (pop-up) menu and Pressing Ctrl+RMB causes a command-dependent action
ON	If the directive is set to on, this functionality is reversed, where clicking right causes a command dependent action and pressing Ctrl+RMB displays the context menu.

Example

```
CTRLRMB_CONTEXTMENU 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Ctrl+RMB Context Menu

CURRENTPDFVIEWER

Defines the current PDF viewer to view the generated PDF.

Syntax

```
CURRENTPDFVIEWER '0'|'2'
```

The default viewer is indicated by `0`. If you specify a custom viewer. the value is set to `2`.

Example

```
CURRENTPDFVIEWER '2'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — Advanced — PDF Viewer

CURRENTPDFVIEWERPATH

Defines the path to the PDF viewer if you specify a custom viewer.

Syntax

```
CURRENTPDFVIEWERPATH 'Default'
```

Example

```
CURRENTPDFVIEWERPATH 'C:\Program Files (x86)\Adobe\Acrobat
     11.0\Acrobat\Acrobat.exe'
```

Corresponding UI Option for Design Entry HDL

None

CURSOR_SHAPES

Enables different cursor shapes based on command mode

Syntax

```
CURSOR_SHAPES 'ON'|'OFF'
```

Example

```
CURSOR_SHAPES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Cursor Shapes

CUSTOM_CDSLIB_SEARCH

Provides a parallel search mechanism in case the default search mechanism for cds.lib fails which results in failure in opening a schematic. This method is similar to the default search method.

Syntax

```
CUSTOM_CDSLIB_SEARCH 'ON'|'OFF'
```

Example

```
CUSTOM_CDSLIB_SEARCH 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

6

D Directives

This chapter lists the CPM directives that start with `D` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

DAO_Timeout

The default timeout for the library server connection setup is 300 `ms`. This directive lets you define the time period (in seconds) after which the library server connection will be timed out. This is useful if you are unable to connect to the server because of network latency issues.

The directive can be set at the site or project levels.

Important

This directive is applicable only to the database mode.

Syntax

```
DAO_Timeout 'value'
```

Example

```
DAO_Timeout '5'
```

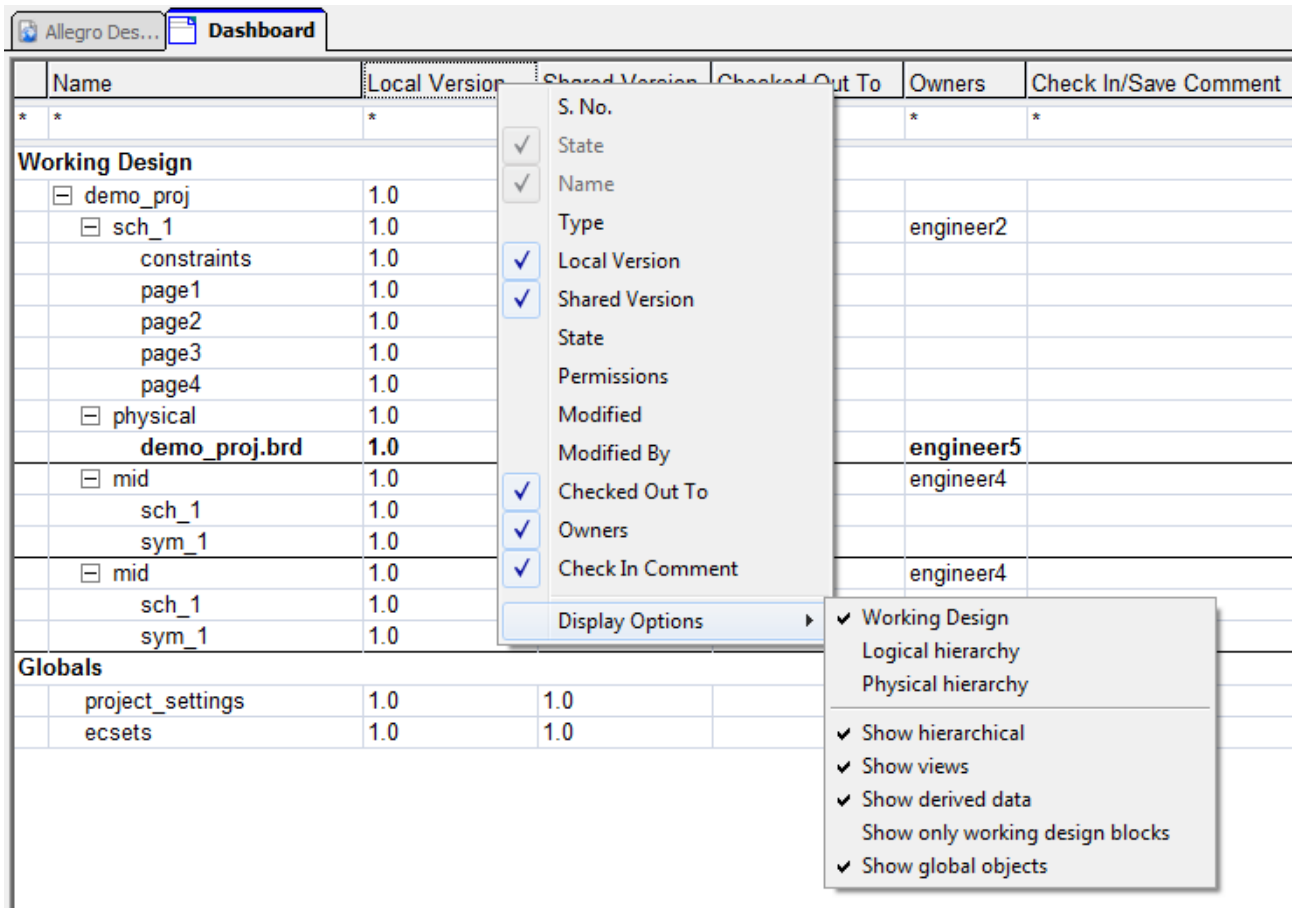
Corresponding UI Option

Configuration – Setup – General – Server Configuration – Time Out

DASHBOARD_SHOW_LOGICAL_HIERARCHY

Set this directive to ON if you want the *Logical hierarchy* view in the Allegro Design Management (team design) *Dashboard* to be preserved between sessions.

The Allegro Design Management (team design) *Dashboard* automatically displays a particular view depending on the role of the user who has logged in. You can customize or configure the dashboard view by right-clicking on the dashboard header and choosing *Working Design*, *Logical hierarchy*, or *Physical hierarchy*. You can also choose all three views.



The view options you choose in the Allegro Design Management *Dashboard* are only valid for the current session.

If you choose the *Logical hierarchy* view and want to preserve it between sessions, set this directive to ON.

Syntax

```
DASHBOARD_SHOW_LOGICAL_HIERARCHY 'YES'|'NO'|'ON'|'OFF'
```

Example

```
DASHBOARD_SHOW_LOGICAL_HIERARCHY 'YES'
```

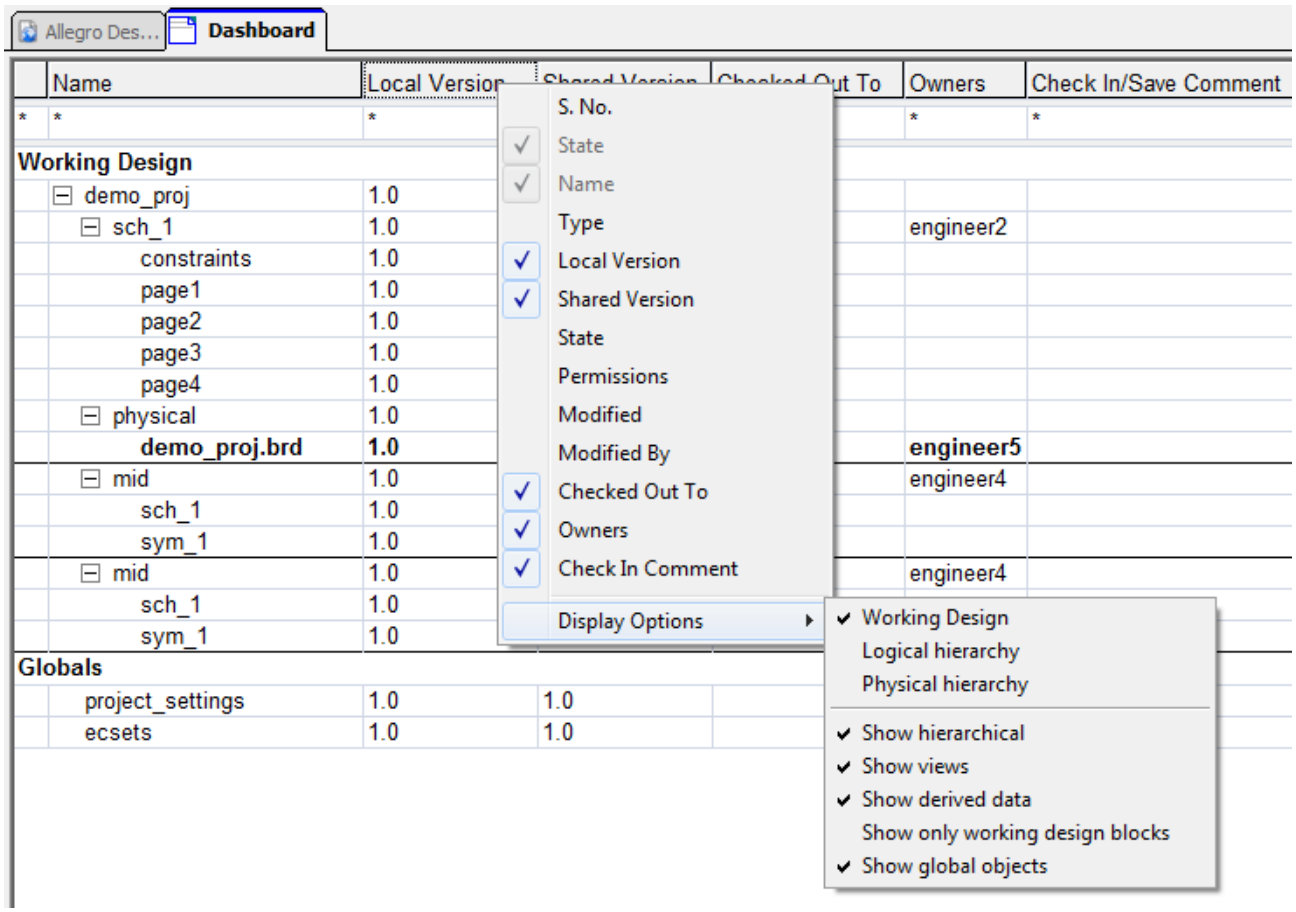
Corresponding UI Option

None

DASHBOARD_SHOW_PHYSICAL_HIERARCHY

Set this directive to `ON` if you want the *Physical hierarchy* view in the Allegro Design Management (team design) *Dashboard* to be preserved between sessions.

The Allegro Design Management (team design) *Dashboard* automatically displays a particular view depending on the role of the user who has logged in. You can customize or configure the dashboard view by right-clicking on the dashboard header and choosing *Working Design*, *Logical hierarchy*, or *Physical hierarchy*. You can also choose all three views.



The view options you choose in the Allegro Design Management *Dashboard* are only valid for the current session.

If you choose the *Physical hierarchy* view and want to preserve it between sessions, set this directive to `ON`.

Syntax

```
DASHBOARD_SHOW_PHYSICAL_HIERARCHY 'YES'|'NO'|'ON'|'OFF'
```

Example

```
DASHBOARD_SHOW_PHYSICAL_HIERARCHY 'YES'
```

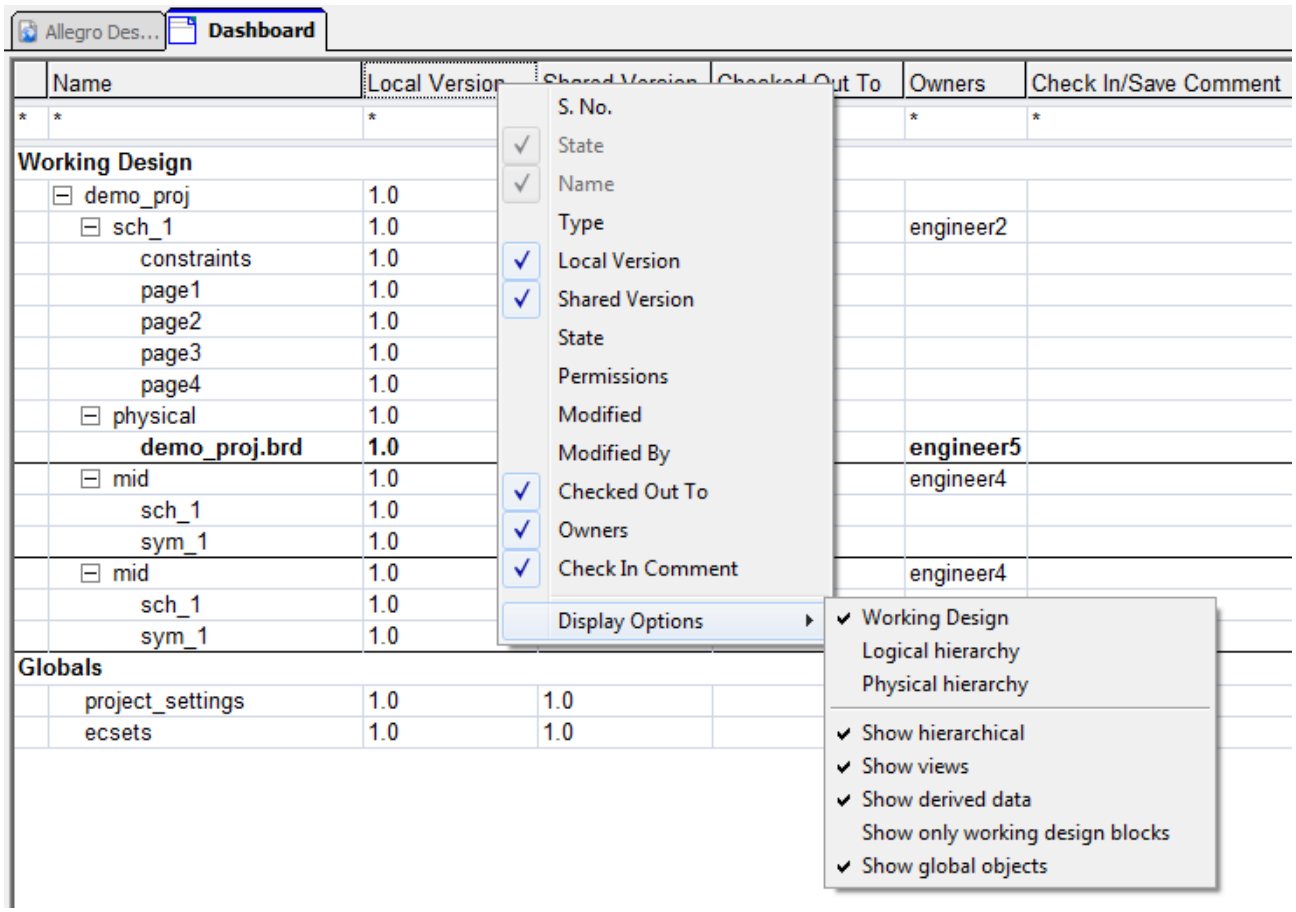
Corresponding UI Option

None

DASHBOARD_SHOW_WORKING_DESIGN

Set this directive to `ON` if you want the *Working Design* view in the Allegro Design Management (team design) *Dashboard* to be preserved between user sessions.

The Allegro Design Management (team design) *Dashboard* automatically displays a particular view depending on the role of the user who has logged in. You can customize or configure the dashboard view by right-clicking on the dashboard header and choosing *Working Design*, *Logical hierarchy*, or *Physical hierarchy*. You can also choose all three views.



The view options you choose in the Allegro Design Management *Dashboard* are only valid for the current session.

If you choose the *Working Design* view and want to preserve it between sessions, set this directive to `ON`.

Syntax

```
DASHBOARD_SHOW_WORKING_DESIGN 'YES'|'NO'|'ON'|'OFF'
```

Example

```
DASHBOARD_SHOW_WORKING_DESIGN 'YES'
```

Corresponding UI Option

None

DataCompress

Specifies whether the data to be sent to or received from the server should be compressed. This is done to enhance Part Information Manager performance.

Syntax

```
DataCompress 'TRUE' | 'FALSE'
```

Example

```
DataCompress 'TRUE'
```

Datasheet_URL

Defines the location (a directory or the URL of a web page or intranet) where a file, which you want to refer to decide which part to add to your design, resides. For example:

```
Datasheet_URL 'D:/datasheets/@<column name>@'
```

The value of the column name you specify within the at the symbol signs will be appended to the file or website page that opens. For example, there are 10 datasheets in PDF format in the D drive in a folder called datasheets. You can specify the directive as follows:
`Datasheet_URL 'D:\datasheets\@Part Number@.pdf'`

When you right-click on a part row in Part Information Manager, and select the *Datasheet URL* option, Part Information Manager searches for a PDF with the same name as the column value, and opens the relevant PDF.

Syntax

```
DataSheet_Url '<URL>'|'<directory path>'
```

Example

```
DataSheet_Url 'https://www.cadence.com/parts/datasheets/@part number@.pdf'
```

```
Datasheet_URL 'D:/datasheets/@<column name>@'
```

```
Datasheet_URL 'http://www.intel.com/content/www/us/en/support/processors/desktop-processors/000006479.html/@<column name>@'
```

Corresponding UI Option

None

See Also

Display_URL

DATATIPS_PROP_NAME

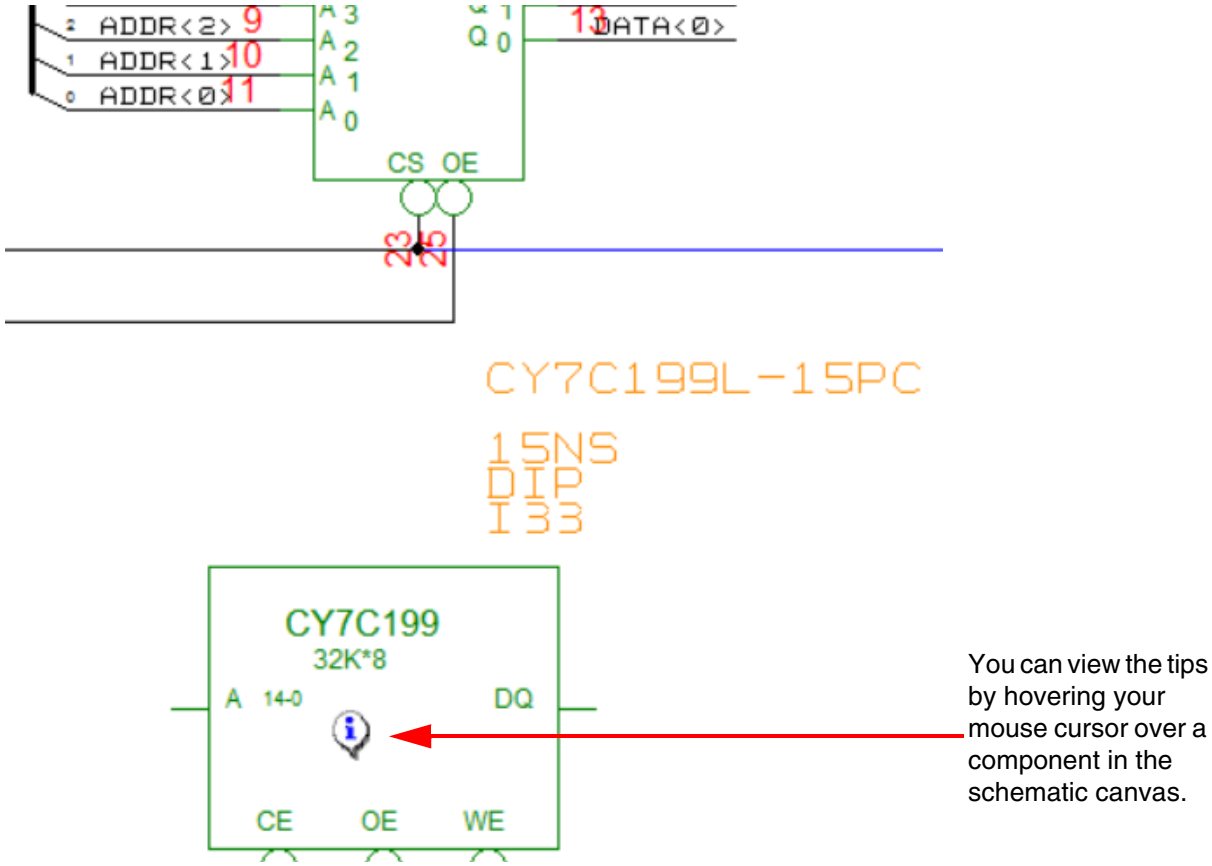
When defined, this directive allows you to provide part-specific recommendations to designers for guidance purposes in the form of data tips. Designers will see these data tips in the schematic after adding components to a design. The directive can be configured at the project or site level.

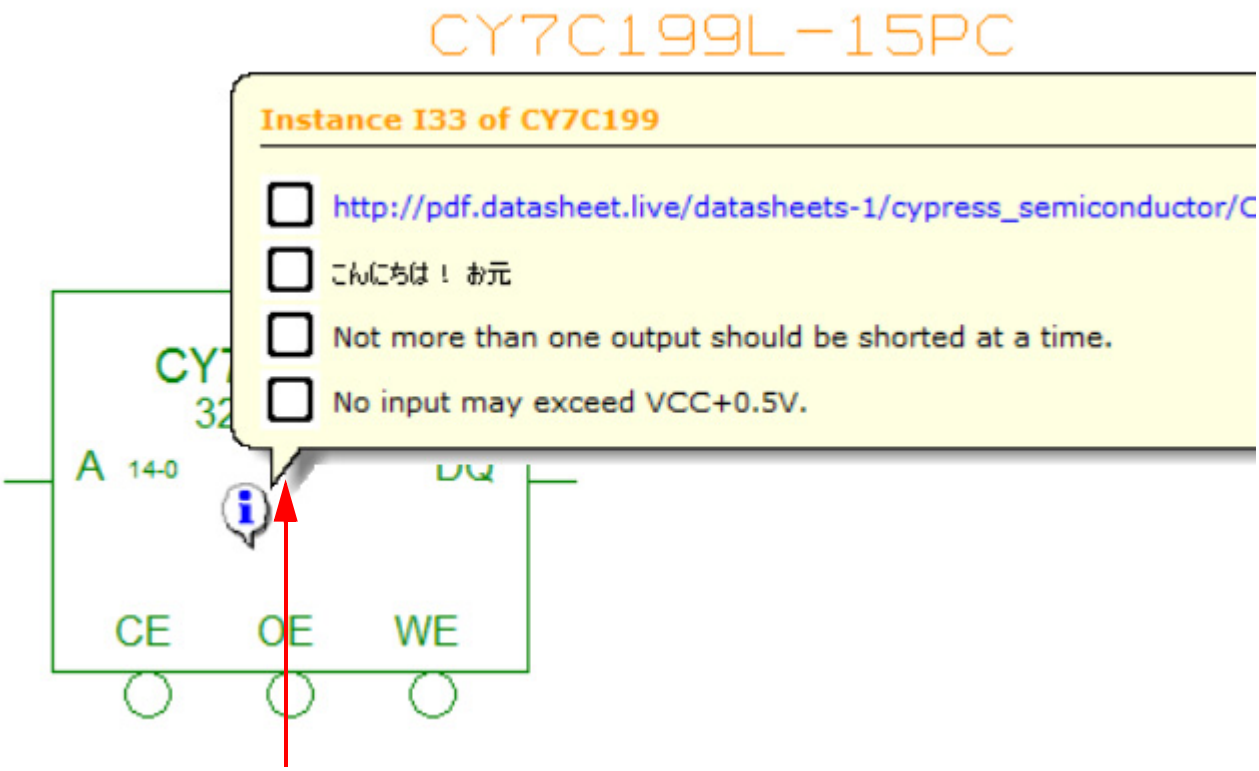
Data tips need to be defined in a text file with the name `datatips.txt`. The data tips file can contain text, hyperlinks as well as Japanese characters.

A property is used to associate the part on the schematic canvas with the entry in the data tips file. The value of the property on the part is matched with the entry in the data tips file and the corresponding data tips are displayed. The name of the property to be used to associate the part needs to be specified in the `.cpm` file.

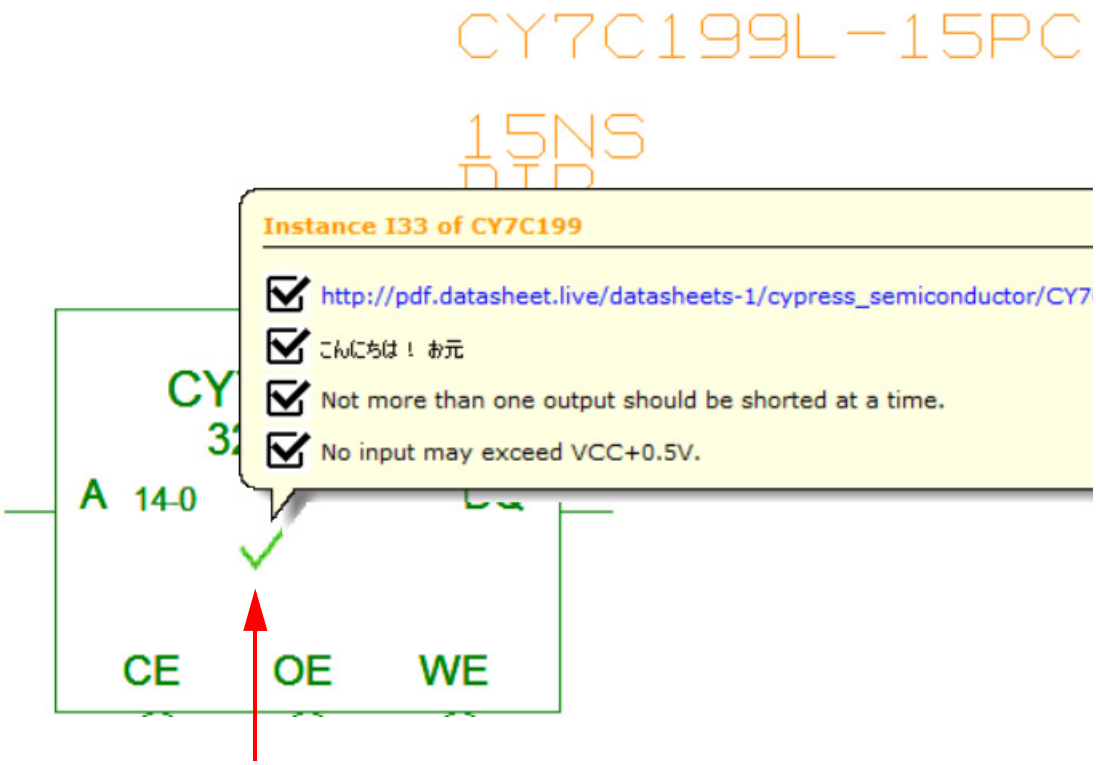
For example, if some parts in a design have a common property such as `PART_NUMBER`, this property can be used to define data tips. The data tips file can contain entries corresponding

to different values of the `PART_NUMBER` property. An information icon is displayed on components that have data tips in a schematic.





You can select the relevant check box to indicate that you have adhered to the guideline.

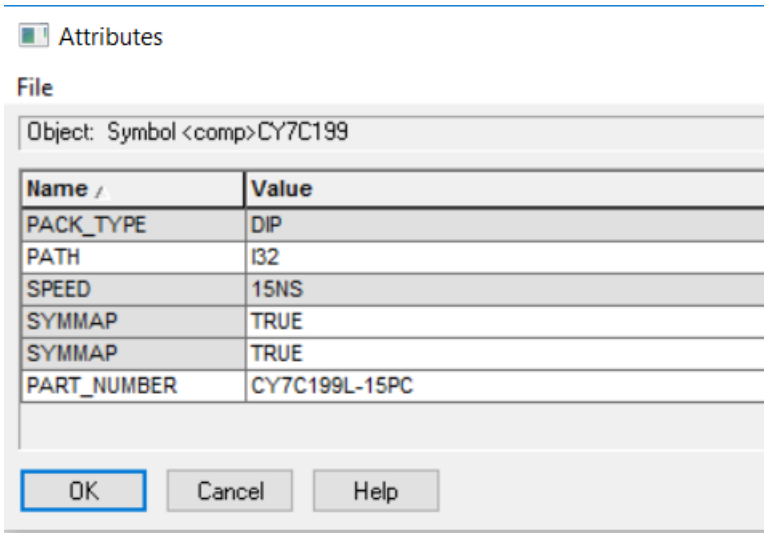


Information icon changes to a check mark if all the guidelines have been followed

To add data tips for a part, do the following:

1. Specify the `DATATIPS_PROP_NAME` '*<Part_Property_Name>*' directive in the `START_CONCEPTHDL...END_CONCEPTHDL` section in the `.cpm` file.

You can add and view the part property names and values in the Attributes box on the schematic canvas as illustrated:



For example, specify `PART_NUMBER` as the part property name in the `.cpm` file. `DATATIPS_PROP_NAME 'PART_NUMBER'`

2. Create a `.txt` file for the data tips that you want to define and name it `datatips.txt`. Store the file in the `cdssetup` folder at the project, `$HOME`, or site level.
3. Define the part property value in this `.txt` file. For example: `PART "CY7C199L-15PC"`. `PART` is a keyword to define the part for which data tips will be defined.
4. Define the data tips you want for this part in the following format:

   `<NUM>:<Data Tip Details>`

   For example:

```
PART "CY7C199L-15PC"
1: "http://pdf.datasheet.live/datasheets-1/cypress_semiconductor/CY7C199L-25ZC
2: "こんにちは！ お元"
3: "Not more than one output should be shorted at a time."
4: "No input may exceed VCC+0.5V."
```

   The data tips are displayed in the same order as the defined number in the data tip file. The same number is used to keep track of which data tip has been marked checked. If you check all the guidelines, the information icon changes to a check mark.

5. Save the file.

Syntax

DATATIPS_PROP_NAME'<*Part_Property_Name*>'

Example

DATATIPS_PROP_NAME'PART_NUMBER'

Corresponding UI Option for Allegro Design Entry HDL

None

Default_Diffpair_Value

The `ConceptSetup_SplitPart_SymbolProp` directive specifies the prefix or suffix that is to be added to the basename of constituent differential pair pins to create a differential pair name

Syntax

```
Default_Diffpair_Value <<string>:<affix>>
```

Example

```
Default_Diffpair_Value 'DP_:Prefix'
```

This directive creates a differential pair called DP_A when the constituent pins are A+ and A-.

Corresponding UI Option for Part Developer

None

See Also

DiffPair_Recognition_Rules

DEFAULT_PAGE_BORDER_NAME

Specifies the default page border used for the current design. Default value is the "B SIZE PAGE". You can change it to any of the Cadence supplied or custom page borders stored in the reference libraries.

Syntax

```
DEFAULT_PAGE_BORDER_NAME 'A SIZE PAGE'|'B SIZE PAGE'|'C SIZE PAGE'|'D SIZE PAGE'|'E
     SIZE PAGE'
```

Example

```
DEFAULT_PAGE_BORDER_NAME 'B SIZE PAGE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Page Border — Symbol

See Also

DEFAULT_PAGE_BORDER_VERSION

DEFAULT_PAGE_BORDER_VERSION

Specify the version of the page border symbol in the Version field.

Syntax

```
DEFAULT_PAGE_BORDER_VERSION <version number>
```

Example

```
DEFAULT_PAGE_BORDER_VERSION '1'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Page Border — Version

See Also

DEFAULT_PAGE_BORDER_NAME

DEFAULT_PHYS_DES_PREFIX

The DEFAULT_PHYS_DES_PREFIX directive specifies the prefix string for reference designator values when no PHYS_DES_PREFIX is found in Allegro System Capture and Packager-XL.

The DEFAULT_PHYS_DES_PREFIX directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Caution

DEFAULT_PHYS_DES_PREFIX overrides the default naming convention that Packager-XL uses. DEFAULT_PHYS_DES_PREFIX should only be used for custom reference designator requirements.

Syntax

```
default_phys_des_prefix <pattern>;
```

The default value for the DEFAULT_PHYS_DES_PREFIX directive is U.

Example

```
default_phys_des_prefix MYPREFIX;
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

Default_Search_Tab

Defines the default, active tab in Part Information Manager. When Part Information Manager is launched, and when you select a classification in Part Information Manager, or when you load search criteria, the tab you define using this directive will be the active tab by default.

The directive is defined in the START_COMPBROWSER section of the `.cpm` file.

Syntax

```
DEFAULT_SEARCH_TAB '<Tab Name>'
```

Example

```
DEFAULT_SEARCH_TAB 'Properties'
```

Corresponding UI Option

None

Default_ShoppingCart_Quantity

Lets you define the number of parts that are added to the Shopping Cart.

Important

This directive is applicable only to the database mode.

Syntax

```
Default_ShoppingCart_Quantity 'value'
```

Example

```
Default_ShoppingCart_Quantity '1'
```

Corresponding UI Option

Configuration – Setup – General – Default Shopping Cart Quantity

Default_Zoom_Factor

The `Default_Zoom_Factor` directive specifies the factor by which a symbol is magnified or shrunk in zoom in and zoom out operations

Syntax

```
Default_Zoom_Factor '<VALUE>'
```

Example

```
Default_Zoom_Factor '50'
```

Corresponding UI Option for Part Developer

None

DELETE_ASCII

Removes the existing ASCII files in your schematic, when you want only binary files to be written.

For more information, see the Design Entry HDL Options dialog box help.

Syntax

```
DELETE_ASCII 'ON'|'OFF'
```

where

ON	If this directive is set to ON, the `.csb` file(s) is saved and the `.csa` file(s), if present, is deleted.
OFF	This is the default value.

Example

```
DELETE_ASCII 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write — Remove ASCII File

See Also

DELETE_BINARY

DELETE_BINARY

Removes the existing binary files in your schematic, when you want only ASCII files to be written. If this directive is set to ON, the `.csa` file(s) is saved and the `.csb` file(s), if present, is deleted.

For more information, see the Design Entry HDL Options dialog box help.

Syntax

```
DELETE_BINARY 'ON'|'OFF'
```

where

ON	If this directive is set to ON, the `.csb` file(s) is saved and the `.csa` file(s), if present, is deleted.
OFF	This is the default value.

Example

```
DELETE_BINARY 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write — Remove Binary File

See Also

DELETE_ASCII

delete_folders_on_copyproj

This directive is used when creating a copy of a project in Allegro System Capture. If the project being copied has subfolders, such an output folder, use this directive to skip folders to be copied to the new design.

Syntax

```
delete_folders_on_copyproj './output/^design_name^/foldername'
```

Example

```
delete_folders_on_copyproj './output/^design_name^/bom' './output/^design_name^/
     packaged'
```

In this example, two folders, namely *bom* and *packaged* under the `output` folder will not be copied to the newly created project.

Corresponding UI Option for Allegro System Capture

None

See Also

rename_folders_on_copyproj

DELETE_UNATTACHED_INVISIBLE_PROPS

When a group is created, some properties that are close to the components that are being selected for the group are also included in the group. Use the `DELETE_UNATTACHED_INVISIBLE_PROPS` directive to remove these properties.

Removing these properties using this directive also ensures that DE-HDL does not delete the invisible properties of components that were not selected.

Syntax

```
DELETE_UNATTACHED_INVISIBLE_PROPS 'ON'|'OFF'
```

Example

```
DELETE_UNATTACHED_INVISIBLE_PROPS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

DESIGN_TYPE

Defines the grid type for the schematic.

Syntax

```
DESIGN_TYPE 'DECIMAL'|'FRACTIONAL'|'METRIC'
```

where

DECIMAL	Bases drawings on the decimal system (500 units per physical inch). This is the default value.
FRACTIONAL	Bases drawings on 400 units per inch. Components appear 25 percent larger.
METRIC	Bases drawings on the metric system (20 units per millimeter; 508 units per inch).

Example

```
DESIGN_TYPE 'DECIMAL'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Type

Detail_Tab_Order

Controls the order in which the `PPT`, `Properties`, `Attributes`, and `Graphics` nodes are displayed in the details pane.

Syntax

```
Detail_Tab_Order '<node 1>' '<node 2>' '<node 3>' '<node 4>'
```

Example

```
Detail_Tab_Order 'PPT' 'Properties' 'Attributes' 'Graphics'
```

Corresponding UI Option

Configuration – Setup – Details – Details Tab Order

DIFFPAIR_PIN_SUFFIX_N

Defines the pattern for negative pin polarity of differential pair pins.

Nets or pins are identified as negative polarities when the following *Connectivity Checks* audit rules are run:

- `Differential pair net polarity mismatch`
- `Differential pair pin polarity mismatch`

If the polarities of the pins of different ICs connected through a differential pair do not match, the error is flagged based on the patterns defined in this directive.

Syntax

```
DIFFPAIR_PIN_SUFFIX_N '<pin pattern>'
```

The following pin patters are supported:

```
M, _N, _M, _NX, _NEG, _MINUS, \\-, \\*
```

Example

```
DIFFPAIR_PIN_SUFFIX_N 'N' 'NX'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Negative Polarity*

DIFFPAIR_PIN_SUFFIX_P

Defines the pattern for positive pin polarity of differential pair pins.

Nets or pins are identified as positive polarities when the following *Connectivity Checks* audit rules are run:

- `Differential pair net polarity mismatch`
- `Differential pair pin polarity mismatch`

If the polarities of the pins of different ICs connected through a differential pair do not match, the error is flagged based on the patterns defined in this directive.

Syntax

```
DIFFPAIR_PIN_SUFFIX_P '<pin pattern>'
```

Example

```
DIFFPAIR_PIN_SUFFIX_P 'P' 'PX' '_P' '_PX' '_POS' '_PLUS' '\\+'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Positive Polarity*

DiffPair_Recognition_Rules

The `DiffPair_Recognition_Rules` directive specifies a set of prefixes and suffixes that Part Developer uses to identify differential pairs in automatic differential-pair creation.

Syntax

```
DiffPair_Recognition_Rules '<negative_pin_identifier:AFFIX>
     <positive_pin_identifier:AFFIX>'
```

Note: You can use the semicolon as a separator to add multiple naming rules.

Important

The positive pin and negative pin identifiers in a differential pair recognition rule must have the same affix. In other words, a naming rule such as `n:SUFFIX,p:PREFIX` is not valid.

Example

```
DiffPair_Recognition_Rules 'n:SUFFIX,p:SUFFIX;-
:SUFFIX,+:SUFFIX;_L:SUFFIX,_H:SUFFIX;_LOW:SUFFIX,_HIGH:SUFFIX'
```

Corresponding UI Option for Part Developer

None

See Also

Default_Diffpair_Value

Specifying a Naming Convention for Autocreation of Differential Pairs

DISABLE_SMARTPDF_SMART_FEATURES

Inactivates the smart features for Smart PDF when the *Print* dialog box is opened. In case you need the smart features to be inactive when the Print dialog box is opened, set this to ON in the canvas section of `cds.cpm` file.

Syntax

```
DISABLE_SMARTPDF_SMART_FEATURES 'ON'|'OFF'
```

The default value of this directive is OFF.

Example

```
DISABLE_SMARTPDF_SMART_FEATURES 'OFF'
```

Corresponding UI Option for Allegro X System Capture

None

DISALLOW_MULTIUSER_PAGE_OVERWRITE

Stops a user from saving a modified page being locked by another user in a single session of Design Entry HDL even if the lock file (`.lck`) is deleted.

Syntax

```
DISALLOW_MULTIUSER_PAGE_OVERWRITE 'ON'|'OFF'
```

Example

```
DISALLOW_MULTIUSER_PAGE_OVERWRITE 'ON'
```

Scenario

User 1 opens a design and edits page 1 of the design. The `.lck` file is created for page 1 with ownership of user 1. Another user, user 2, opens the same design and gets the message that page 1 is locked by user 1. User 2 now modifies page 1 and moves to page 2. At this point, user 1 exits the design and the `.lck` file is deleted. Now user 2 moves back to page 1. If the `DISALLOW_MULTIUSER_PAGE_OVERWRITE` directive is set to 'ON', the message stating that the page is locked appears again and user 2 is not able to save page 1. If the directive is set to 'OFF', no message is displayed and user 2 is able to save page 1.

Corresponding UI Option for Allegro Design Entry HDL

None

DISCONNECT_PIN_TEXT_NAME

When adding pins to a symbol, by default *Display Name* and *Pin ID* are kept as same, and editing *Pin ID* is not allowed. In case you want the *Pin ID* to be different, set this directive to ON in the `site.cpm` file to make this field editable.

Syntax

```
DISCONNECT_PIN_TEXT_NAME 'ON'|'OFF'
```

The default value is OFF.

Example

```
DISCONNECT_PIN_TEXT_NAME 'ON'
```

Corresponding UI Option for Allegro System Capture

None

DISPLAY_UNCONNECTED_PINS

Defines whether unconnected pins should be displayed on the canvas or not.

Syntax

```
DISPLAY_UNCONNECTED_PINS 'OFF'|'ON'
```

The default value is ON.

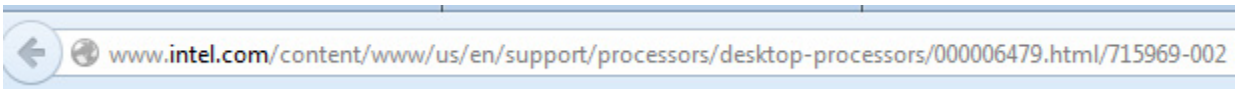
Example

```
DISPLAY_UNCONNECTED_PINS 'ON'
```

Display_URL

Defines how you want Part Information Manager to display the option defined in the `Datasheet_URL` directive.

For example, setting `Display_URL 'View the Cadence datasheet for @PartNumber@'` shows the option as follows:



You can specify as many datasheet and display URLs, as required. Ensure however that the number of datasheet URLs is greater than or equal to the number of display URLs. Also make sure that the Datasheet_URL value matches the column name exactly, including the casing. The DataSheet (and Display) URL option is available from the Search Details tab and from the shopping cart or shopping list.

Syntax

```
Display_Url '<Column Header>'
```

Example

```
DataSheet_Url 'http://www.google.co.in/search?hl=en&output=search&sclient=@Part
     Number@.pdf'
```

Corresponding UI Option

None

See Also

Datasheet_URL

DO_NOT_HONOR_ANNOTATIONS

If this directive is set to ON, only key properties are annotated when you add a new component instance.

Syntax

```
DO_NOT_HONOR_ANNOTATIONS 'OFF'|'ON'
```

Example

```
DO_NOT_HONOR_ANNOTATIONS 'ON'
```

DOC_GRID_MULTIPLE

Displays every nth grid line to define where objects can be placed so that pins do not fall off-grid. This ensures the correct connectivity of wires and symbols.

Syntax

```
DOC_GRID_MULTIPLE '<value>'
```

where

value	any number greater than 0.002.

Example

```
DOC_GRID_MULTIPLE '5'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Document Grid — Multiple

See Also

- DOC_GRID_SIZE
- DOC_GRID_TOGGLE
- DOC_GRID_TYPE
- LOGIC_GRID_MULTIPLE
- SYMBOL_GRID_MULTIPLE

DOC_GRID_SIZE

Adjusts the grid size to be smaller or larger for DOC drawings.

Syntax

```
DOC_GRID_SIZE '<value>'
```

where

value	any positive integer greater than or equal to 0.002.

Example

```
DOC_GRID_SIZE '0.100'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Document Grid — Size

See Also

- DOC_GRID_MULTIPLE
- DOC_GRID_TOGGLE
- DOC_GRID_TYPE

DOC_GRID_TOGGLE

Displays or hides the grid for DOC drawings.

Syntax

```
DOC_GRID_TOGGLE 'ON'|'OFF'
```

Example

```
DOC_GRID_TOGGLE 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Document Grid check box

See Also

- DOC_GRID_MULTIPLE
- DOC_GRID_SIZE
- DOC_GRID_TYPE

DOC_GRID_TYPE

Displays the grid for DOC drawings as dots or dashed lines.

Syntax

```
DOC_GRID_TYPE 'DOTS'|'LINE'
```

Example

```
DOC_GRID_TYPE 'DOTS'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Document Grid — Style — Lines | Dots

See Also

- DOC_GRID_MULTIPLE
- DOC_GRID_SIZE
- DOC_GRID_TOGGLE

DONT_ALLOW_UPREV

When set to 'ON', does not allow uprev of pre-16.5 designs to 16.5 or later versions.

Instead of the uprev dialog, DE-HDL displays a message. When you click *OK* in the message box, DE-HDL closes without upreving the design.

This directive can be used to control the uprev of designs to post-16.5 releases to ensure that synchronization steps are executed before the design is uprev-ed.

Syntax

```
DONT_ALLOW_UPREV 'ON'|'OFF'
```

Example

```
DONT_ALLOW_UPREV 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

DONT_FORCE_ORIGIN_ONGRID

This directive can be used for designs whose component origin is off the grid, but whose pins are on the grid.

When the directive is ON and you move components, DE-HDL moves component by grid units. As a result, off-the-grid components stay off the grid, and on-the-grid components remain on the grid.

When OFF, by default, DE-HDL moves components to the grid by calculating the final position of the component after it is moved.

Note: Sometimes, you might have off-the-grid properties attached to schematic pages or bodies. In certain cases, you might want these properties placed off the grid while maintaining relative distance from the grid when using the Copy Repeat or Move commands. For this, you could use the `SET` command to turn this directive on or off from within DE-HDL instead of turning it on in the `.cpm` file.

Syntax

```
DONT_FORCE_ORIGIN_ONGRID 'ON'|'OFF'
```

Example

```
DONT_FORCE_ORIGIN_ONGRID 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

DONT_SET_OFFSET_PINNUMBER

This directive can be used to disable the offset on pin numbers when moving or placing pins in the symbol view.

Unlike pin names and pin text, pin numbers are usually placed over pin stubs. Therefore, to avoid an overlap with pin stubs or symbol bodies, numbers are placed with a slight offset (10 units on the x and y directions). You can switch the offset off, if needed.

Syntax

```
DONT_SET_OFFSET_PINNUMBER 'ON'|'OFF'
```

Example

```
DONT_SET_OFFSET_PINNUMBER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

DONT_SHOW_CM_DLG

When you launch Constraint Manager from Design Entry HDL, a message pops up to prompt you that the current version of Constraint Manager connected to Design Entry HDL is compatible only with the corresponding version of Allegro PCB Editor and Allegro SI version. You can hide this dialog box, by setting this directive to `ON`.

Syntax

```
DONT_SHOW_CM_DLG 'ON'|'OFF'
```

Example

```
DONT_SHOW_CM_DLG 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Edit — Constraints — Compatibility with PCB Editor and Allegro SI dialog box — Don't show me the message again check box

DOT_COLOR

Changes the default dot color on a schematic drawing.

Syntax

```
DOT_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
DOT_COLOR 'white'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Dot

See Also

- BACKGROUND_COLOR
- ARC_COLOR
- HIGHLIGHT_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

DOTS

Adds open or filled dots at wire connections.

Syntax

```
DOTS 'FILLED'|'OPEN'
```

Example

```
DOTS 'FILLED'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Dots — Style — Open/Filled

See Also

DOT_COLOR

DPPIN_PREFIX

Use this directive to specify the default prefix string to be used for naming differential pairs for pins.

Syntax

```
dppin_prefix '<string>'
```

Example

```
dppin_prefix 'DP_'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Differential Pairs — Prefixes for Differential Pair Names — For Pins

See Also

- DPPIN_RULES
- DPSIG_PREFIX
- DPSIG_RULES

DPPIN_RULES

Use this directive to specify the characters that are used as suffixes or prefixes with the pin names to indicate the pins of a differential pair.

Syntax

```
dppin_rules '<character>;<S or P>;<character>;<S or P>'
```

Where S and P represent the suffix and prefix respectively.

Example

```
dppin_rules '-;S;+;S'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Differential Pairs — Formats for Naming Differential Pair Pins

See Also

- DPPIN_PREFIX
- DPSIG_PREFIX
- DPSIG_RULES
-

DPSIG_PREFIX

Use this directive to specify the default string prefix to be used for naming differential pair for signals.

Syntax

```
dppin_prefix '<prefix>'
```

Example

```
dppin_prefix 'DS_'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Differential Pairs — Prefixes for Differential Pair Names — For Signals

See Also

- DPPIN_PREFIX
- DPPIN_RULES
- DPSIG_RULES

DPSIG_RULES

Use this directive to specify the characters that are used as suffixes or prefixes with the signal names to indicate the member nets of a differential pair.

Syntax

dpsig_rules `<character>;<character>;<S or P>`

Where S and P represent suffix and prefix respectively.

Example

```
dpsig_rules '-;+;S'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Differential Pairs — Formats for Naming Differential Pair Signals

See Also

- DPPIN_PREFIX
- DPPIN_RULES
- DPSIG_PREFIX

DRAWING_BROWSER

Activates the View Open dialog box when you enter the *edit* command in the console window and then press Return.

Syntax

```
DRAWING_BROWSER 'ON'|'OFF'
```

Example

```
DRAWING_BROWSER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Drawing Browser

See Also

- COMPONENT_BROWSER
- LIBRARY_BROWSER

DRC_ERROR

Specifies a comma separated list of DRC rules which, if fail, are reported as errors in the violation window on running DRC checks.

Syntax

```
DRC_ERROR '<list_of_DRC_rules>'
```

list_of_DRC_rules The list of DRCs include, but not restricted to the following rules:

- asda_inst_note_overlap
- asda_inst_overlap
- asda_inst_prop_offset
- asda_inst_prop_overlap
- asda_inst_seg_overlap
- asda_iscline_present
- asda_jedec_type
- asda_min_wire_spacing
- asda_missing_asymm_function
- asda_missing_split_function
- asda_note_overlap
- asda_note_prop_overlap
- asda_prop_overlap
- asda_seg_note_overlap
- asda_seg_prop_overlap
- asda_single_node_net
- asda_unconnected_diffpair_signal
- asda_unconnected_pin
- asda_wire_prop_offset

Example

```
DRC_ERROR 'asda_unconnected_pin', 'asda_missing_asymm_function',
'asda_missing_split_function'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Design Rule Check

DRC_INFO

Specifies a comma separated list of DRC rules which, if fail, are reported as information in the violation window on running DRC checks.

Syntax

```
DRC_INFO '<list_of_DRC_rules>'
```

list_of_DRC_rules Same as DRC_ERROR.

Example

```
DRC_INFO 'asda_note_overlap', 'asda_note_prop_overlap',
'asda_single_node_net'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Design Rule Check

DRC_WARN

Specifies a comma separated list of DRC rules which, if fail, are reported as warnings in the violation window on running DRC checks.

Syntax

```
DRC_WARN '<list_of_DRC_rules>'
```

list_of_DRC_rules Same as DRC_ERROR.

Example

```
DRC_WARN 'asda_single_node_net', 'asda_min_wire_spacing', 'asda_inst_overlap'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Design Rule Check

dump_FileName

Specifies the name of the dump file that contains up-to-date revision information about cells and blocks in Library Revision Manager. If you do not provide a file name, the default name, `lrmDumpFile.lrmDump`, is used.

Syntax

`dump_FileName '<name of file>'`

Example

```
dump_FileName 'LRM11.lrmDump'
```

Corresponding UI Option

None

7

E Directives

This chapter lists the CPM directives that start with `E` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

EDIT_PHYSICAL_NET_NAME

Set this directive to enable or disable editing of Physical Net Names.

Syntax

```
EDIT_PHYSICAL_NET_NAME 'OFF'|'ON'
```

where

`ON`	Editing of Physical Net Names is enabled.
`OFF`	Editing of Physical Net Names is disabled.

Example

```
EDIT_PHYSICAL_NET_NAME 'OFF'
```

Corresponding UI Option

None

EDIT_PHYSICAL_SPACING_CONSTRAINTS

This directive controls the editing of physical and spacing constraints in Design Entry HDL-Constraint Manager.

When the directive value is set to ON, the logic designer can view and edit physical and spacing constraints in Design Entry HDL-Constraint Manager.

When the directive value is set to OFF, logic designers can **only view** the physical and spacing constraints in Design Entry HDL-Constraint Manager. Designers cannot edit these constraints. Constraint values can be captured only in the physical layout. When the layout is synchronized with the logic design, these constraints flow from the layout to the logic design and are available for viewing in the logic design. During the front to back flow, these constraints are not transferred from the logic design to the layout.

Electrical constraints are always editable and are synchronized during the front to back and back to front flows.

You can lock the directive at the site level by setting the following in `site.cpm`:

```
START_CONSTRAINT_MGR_CONTROL_SETTINGS
EDIT_PHYSICAL_SPACING_CONSTRAINTS ‘LOCK’
END_CONSTRAINT_MGR_CONTROL_SETTINGS
```

Syntax

```
EDIT_PHYSICAL_SPACING_CONSTRAINTS 'ON'|'OFF'
```

Example

```
START_CONSTRAINT_MGR
EDIT_PHYSICAL_SPACING_CONSTRAINTS 'OFF'
END_CONSTRAINT_MGR
```

Corresponding UI Option for Allegro Design Entry HDL

None

EDITABLE_IMPORT_TABLE

The tables created with data being sourced from a CSV file are read only, by default. These tables cannot be edited. Set the value of this directive to ‘True’ to edit the tables imported from the csv file.

Syntax

```
EDITABLE_IMPORT_TABLE 'TRUE'|'FALSE'
```

Example

```
EDITABLE_IMPORT_TABLE 'TRUE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - General - Allow Editing of Table created using CSV File

See Also

- MAX_ROW_IMPORT_TABLE
- MAX_COL_IMPORT_TABLE

ELECTRICAL_CONSTRAINTS

When this directive is set to ON, the Enable Export option is available in the Export Physical dialog. When a project is created, this directive is set to OFF by default. When Constraint Manager is launched from Design Entry HDL, the directive is set to ON and the *Enable Export* option is checked ON in the Export Physical dialog.

Syntax

```
ELECTRICAL_CONSTRAINTS 'ON'|'OFF'
```

on	When the value is set to `on`, it enables the *Enable Export* option in the Export Physical dialog. The default value for the `ELECTRICAL_CONSTRAINTS` directive is `ON`.
off	When the value is set to `off` it disables the *Enable Export* option in the Export Physical Form

Release 16.6 onwards, this directive is set to ON by default.

Example

```
ELECTRICAL_CONSTRAINTS 'ON'
```

ELECTRICAL_LOW_STRESS_LEVEL

Displays the components under the safe stress level in the *Electrical Stress Results* report based on the specified value. For example, if the specified value is 30, a component stressed below 30% is flagged.

Syntax

```
ELECTRICAL_LOW_STRESS_LEVEL '<Value>'
```

Example

```
ELECTRICAL_LOW_STRESS_LEVEL '40'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Electrical Stress Settings – Devices tab – *Parameter Settings - Safestress Level(%)*

See Also

- ELECTRICAL_OVER_STRESS_LEVEL

ELECTRICAL_OVER_STRESS_LEVEL

Displays the components that exceed the over stress level in the *Electrical Stress Results* report based on the specified value. For example, if the specified value is 80, a component stressed above 80% is flagged.

Syntax

```
ELECTRICAL_OVER_STRESS_LEVEL '<Value>'
```

Example

```
ELECTRICAL_OVER_STRESS_LEVEL '85'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Electrical Stress Settings – Devices tab – *Parameter Settings - Overstress Level(%)*

See Also

- ELECTRICAL_LOW_STRESS_LEVEL

ENABLE_ALPHANUMERIC_CUSTOMKEYS

Allows you to set custom alphanumeric shortcut keys for commonly used commands in DE-HDL. When set to ON, you can add alphanumeric shortcut keys to the `concepthdl_keys.txt` file as well as set alphanumeric shortcut keys from the *Tools — Customize* menu option.

For more information about setting shortcut keys, refer to the *Customizing Keys* section of *Allegro Design Entry HDL Reference Guide*.

Syntax

```
ENABLE_ALPHANUMERIC_CUSTOMKEYS 'ON'|'OFF'
```

Example

```
ENABLE_ALPHANUMERIC_CUSTOMKEYS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Customize — Keys tab

ENABLE_FONT_BASED_BBOX_COMPUTATION

Computes the size of text based on the font used for symbols.

In Rules Checker, bounding boxes for text and properties in a schematic or symbol are computed based on the string length and default width of the ConceptFont.

If you use a different font for symbol text, you can use this directive. This ensures that when body rules with overlap checks are selected, Rules Checker calculates the size of symbol text (schematic text is not supported in this release) based on the font used.

This directive is set to ON by default.

Syntax

```
ENABLE_FONT_BASED_BBOX_COMPUTATION 'ON'|'OFF'|'0'|'1'|'TRUE'|'FALSE'
```

Example

```
ENABLE_FONT_BASED_BBOX_COMPUTATION 'OFF'
```

Corresponding UI Option for Design Entry HDL

None

ENABLE_SEL_LOGICAL_UPDATE_VDD

When set to ON, this directive allows you to update selected design differences from the *Visual Design Differences* pane in System Connectivity Manager.

The following two options are available in the *Visual Design Differences* pane when this directive is set to ON:

- *Update — All*: Updates design and constraint differences
- *Update — Selected Items Only*: Updates only the selected design differences. Constraint differences cannot be updated with this option. Choose *Update — All* to update constraint differences.

For more information about updating selected design differences, refer to the *Updating Design Differences in System Connectivity Manager* section of *System Connectivity Manager User Guide*.

Syntax

```
ENABLE_SEL_LOGICAL_UPDATE_VDD 'ON'|'OFF'
```

Example

```
ENABLE_SEL_LOGICAL_UPDATE_VDD 'ON'
```

Corresponding UI Option for System Connectivity Manager

None

ERR_OK_NET_ONE_PIN_MULTI_NODES

Set this directive if you want an error to be reported when a net is connected to more than one component pin in a design.

When this directive is ON and is used with the OK_NET_ONE_PIN property attached to a net, DE-HDL displays an error message similar to the following when you save the design or hierarchy:

```
The '%s' net is connected to more than one component pin but it has
the 'OK_NET_ONE_PIN' property.
```

```
The 'OK_NET_ONE_PIN' property should be specified only for nets that
are connected to one pin. Check the connections for the net and if
required, delete the 'OK_NET_ONE_PIN' property from the net.
```

If you package the design without saving it or the design hierarchy, DE-HDL displays only a warning. The error is reported in the `pxl.log` report.

For more information about the OK_NET_ONE_PIN property, refer to the *OK_NET_ONE_PIN* section of the *Allegro Platform Properties Reference* guide.

Syntax

```
ERR_OK_NET_ONE_PIN_MULTI_NODES  'ON'|'OFF'
```

Example

```
START_NETLIST

ERR_OK_NET_ONE_PIN_MULTI_NODES 'ON'

END_NETLIST
```

Corresponding UI Option for Allegro Design Entry HDL

None

ERROR_MESSAGES

Specifies where you want error messages to display.

Syntax

```
ERROR_MESSAGES 'DIALOG'|'COMMANDPANE'|'SUPPRESS'
```

where

DIALOG	If you set the value to Dialog, Design Entry HDL displays the messages in a dialog box.
COMMANDPANE	If you set this directive to COMMANDPANE, Design Entry HDL displays the messages in the Console Command Window.
SUPPRESS	If you set the variable to SUPPRESS Design Entry HDL does not display the type of messages.

Example

```
ERROR_MESSAGES 'DIALOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Messages — Error

See Also

- FATAL_MESSAGES
- INFORMATIONAL_MESSAGES
- WARNING_MESSAGES

ERROR_NO_CSB_FILES

Pages in designs created in Design Entry HDL are saved as ASCII and binary design data files with a `.csa` and `.csb` extension respectively. When a design has `.csa` files, it must have corresponding `.csb` files, although a design can have `.csb` files without corresponding `.csa` files.

CRefer currently uses logical page numbers to map cross-references for signals on the schematic. If there is a mismatch between pages because of missing `.csb` files, the resulting cross-references do not point to the right physical page numbers, since they point to the logical page numbers.

In such cases, this directive is useful. If `.csb` files corresponding to `.csa` files in a design are missing, CRefer displays an error message when this directive is set to ON. The error lists the missing `.csb` files in the design that you are cross-referencing and will exit. The missing `.csb` files can be regenerated by saving the required pages.

When this directive is set to OFF, CRefer will prompt you about missing `.csb` files as a warning. It will ignore these missing pages and continue.

This directive is set to ON by default.

Syntax

```
ERROR_NO_CSB_FILES 'ON'|'OFF'
```

Example

```
ERROR_NO_CSB_FILES 'ON'
```

Corresponding UI Option

None

ERROR_ON_PARTIAL_INSTANTIATION_OF_HSS

Use this directive to ensure that a design is packaged only when the hierarchical split blocks are completely instantiated in the design.

Syntax

```
ERROR_ON_PARTIAL_INSTANTIATION_OF_HSS 'ON'|'OFF'
```

on	When the value is set to `on`, Packager-XL stops processing if there are *partially instantiated* hierarchical split blocks present in the design.
off	When the value is set to `off`, Packager-XL continues to process even if there are partially instantiated hierarchical split blocks present in the design.

By default, this directive is set to 'OFF'.

Example

```
ERROR_ON_PARTIAL_INSTANTIATION_OF_HSS 'ON'
```

ETCH_REMOVAL

This directive corresponds to the *Allow Etch Removal During ECO* check box in the Export Physical dialog.

If a pin is removed from a net because of an engineering change order (ECO), use this directive to specify that the etch be ripped up from a removed pin to the closest T connection or pin.

Do not select this option if you want PCB Editor to rip up the etch interactively.

Syntax

```
etch_removal 'ON'|'OFF'|'YES'|'NO'|'1'|'0'
```

Example

```
etch_removal 'NO'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Allow Etch Removal During ECO check box

See Also

- GEN_PSTFILES
- IGNORE_FIXED
- LAUNCH_OPTION
- OVERWRITE_CONSTRAINTS

exclude_autoupdate_props

Use this directive to specify injected property value mismatches that you do not want LRM to auto-update even when the `auto_update_minor_ptf` has been set to *`AUTO`*.

Syntax

```
exclude_autoupdate_props '<prop1>' '<prop2>' '<prop3>'
```

Example

```
exclude_autoupdate_props 'VALUE' 'COST'
```

Corresponding UI Option

None

See Also

auto_update_minor_ptf

EXCLUDE_FILE_PATH

Using this directive, you can specify the name of the file that contains the file names or extensions, or folder names that you want excluded when archiving a project.

Syntax

```
EXCLUDE_FILE_PATH <excludes.txt>
```

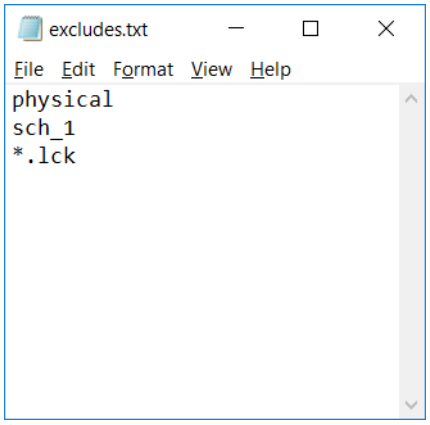
where

`excludes.txt` contains file names or extensions, or folder names that need to be excluded. Specify at least one string. All the strings must be specified on a separate line in the file.

Note: Specify the name of the file along with the path.

Example

```
EXCLUDE_FILE_PATH 'c:/excludes.txt'
```



When archiving a project, to exclude:

- all the lock files in the design, add `.lck` to the `excludes` file.
- a folder named physical, add `physical` to the `excludes` file.
- a file named sessionlog.txt, add `sessionlog.txt` to the `excludes` file.

Corresponding UI Option

None

EXCLUDE_PPT

The EXCLUDE_PPT directive is used to prevent the loading of following files:

1. Ptf files when directories are specified using the `PPT` or `USE_LIBRARY_PPT` directives.
2. Cell level ptf files.

The ptf files are identified by a `.ptf` file extension.

If you specify the name `lsttl`, Packager-XL excludes any file with this name, and any file named `lsttl` with a `.ptf`. You cannot use the `EXCLUDE_PPT` directive in conjunction with the `INCLUDE_PPT` directive. If both directives are specified, an error message is generated and the `EXCLUDE_PPT` directive is ignored.

- You must specify the `EXCLUDE_PPT` directive in the Part Table section of the Project Setup form.
- You can use the `EXCLUDE_PPT` directive for file names only.

Syntax

```
EXCLUDE pptfile_name [,pptfile_name]...;
```

pptfile_name	The name of the part table file. The file extension `.ptf` is optional.

The default value for the `EXCLUDE_PPT` directive is none.

Example

If you have the following directives:

```
PPT /lib/ptfs;

EXCLUDE_PPT sim.ptf;
```

and if the contents of the `/lib/ppts` directory are as follows:

```
lsttl.ptf
analog.ptf
sim.ptf
```

Packager-XL loads the `lsttl.ptf` and `analog.ptf` files.

EXCLUDE_VIEW

Using this directive, you can specify the name of the view you want excluded when archiving a project.

Note: This directive is always used in conjunction with the EXCLUDE_FILE_PATH directive.

Syntax

```
EXCLUDE_VIEW <view_name>
```

where

`<view_name>` is the name of the view.

Example

When archiving a project, if you want to exclude the `sch_1` view, specify:

```
EXCLUDE_VIEW 'sch_1'
```

For more information about different views, refer to the *Selecting Views for the Project* section of *Allegro Project Manager User Guide*.

Corresponding UI Option

None

EXPLICIT_BASE_NET_IDENTIFIER

Defines the text to be displayed on winning or base net names in Allegro System Capture designs to make it easier to identify them.

Syntax

```
EXPLICIT_BASE_NET_IDENTIFIER <text to be shown as the suffix for the base net>
```

Example

```
EXPLICIT_BASE_NET_IDENTIFIER <{base net}>
```

Corresponding UI Option

None

See Also

BASE_NET_OVERLAY

EXPORT

Allows you to specify the information to be exported to the published PDF document. The value of this directive identifies which layers are exported to the published PDF document and which are not

Syntax

`EXPORT '<decimal_value>'`

For each layer, a bit is set in the directive value. To set the bit, you need to add the decimal corresponding values of the layer for the `EXPORT` and VISIBLE directives. For example, if you want to export the `Component` (value = `2`), `Nets` (value = `4`), `Pin Numbers` (value = `8`), and `Signal Names` (value = `16`) layers, you would add the corresponding values for the layers and assign the resultant value, `30`, to the `EXPORT` directive. The decimal value for each layer is listed in the table below.

Layer	Description	Decimal Value of the Layer
Page Border & Title Block	Published to the PDF document by default.	`1`
Component	Published to the PDF document by default.	`2`
Nets	Published to the PDF document by default.	`4`
Pin Numbers	This attribute is attached to a pin as specified in the `chips.prt` file. Pin numbers are published to the PDF document by default. Pin numbers with '`?`' and '`#`' in their names are not exported to the published PDF document.	`8`
Signal Names	This attribute is attached to signals. Signal names are published to the PDF document by default.	`16`

Layer	Description	Decimal Value of the Layer
Cross Reference	This attribute lets you traverse cross-references within the schematic. Cross-references are translated to links on the published PDF document. By default, cross reference attributes are exported to the PDF document.	32
Section and Reference Designator attributes	Published to the PDF document by default.	64
Visible Net attributes	All the net attributes for which the visibility has been set to true in the Attributes dialog box. Published to the PDF document by default.	128
Visible Component attributes	All the component attributes for which the visibility has been set to true in the Attributes dialog box. Published to the PDF document by default.	256
Invisible attributes	This option refers to the invisible properties of nets, components, and pins. This layer corresponds to only those properties of nets, pins, and components which are not visible on schematic canvas. This layer appears in the published PDF document only if you export invisible properties to the PDF document. By default, invisible properties are not published to the PDF document.	512

Layer	Description	Decimal Value of the Layer
Constraints	By default, constraints are not published to the PDF document. To publish constraints, you need to create placeholders in the schematic and publish the PDF document in the Occurrence Edit mode. Constraints which do not have placeholder in schematic are not exported to the PDF document.	`1024`
Notes	By default, notes are not published to the PDF document.	`2048`

Example

```
EXPORT '511'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Layers — Export to File

See Also

VISIBLE

PRINTLAYER

Export_Csv_Delimeter

The `Export_Csv_Delimeter` directive specifies the character to be used as the delimiter in the CSV file created in CSV export.

Syntax

```
Export_Csv_Delimeter '<delimiter>'
```

Example

```
Export_Csv_Delimeter ','
```

Corresponding UI Option for Part Developer

None

Export_Csv_Replace_<property>

The `Export_Csv_Replace_<property>` directives specify the row and column labels that are to be written in the CSV file to store information about different package and pin properties from the exported part.

Note: Package properties are written as rows and pin properties are written in columns.

Directives for the following properties are supported:

- assertionchar
- jedectype
- packagename
- pinlocation
- pinname
- pinnumber
- pinposition
- pintype
- symbol

Syntax

```
Export_Csv_Replace_<property> '<string>'
```

Example

```
Export_Csv_Replace_assertionchar 'ASSERTION_CHAR'
```

Corresponding UI Option for Part Developer

None

Export_ViewLogic_Visibility_<property>

The `Export_ViewLogic_Visibility_<property>` directive controls the visibility of the following pin properties:

- pin name
- pin number
- pin type

Syntax

```
Export_ViewLogic_Visibility_<PinName|PinNumber|PinType> '<1|0>'
```

Example

```
Export_ViewLogic_Visibility_PinName '1'
```

Corresponding UI Option for Part Developer

None

EXTERNAL_ALLEGRO_BOARD_FOLDER

Specify the location where you want the external board files to be generated.

Syntax

```
EXTERNAL_ALLEGRO_BOARD_FOLDER '<path>'
```

where

<path>	Specify the path where you want the external board files to be generated.

Example

```
EXTERNAL_ALLEGRO_BOARD_FOLDER './output/external_allegro_boards'
```

Corresponding UI Option

None

8

F Directives

This chapter lists the CPM directives that start with `F` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

F2B_OVERWRITE_CONSTRAINTS

This directive controls how constraints are synchronized during the front to back flow.

If this directive is set to `ON`, all the constraints in the physical layout are overwritten by the constraints propagated from the logic design.

If this directive is set to `OFF`, constraints from the logic design are merged into the layout using the Changes Only mode. This means that only those constraints that have been modified in the logic database since the last synchronization between the logic design and the layout are transferred from the logic design to the layout. Constraints that have been updated in the layout since the last synchronization are not updated in the front to back flow. This allows users to capture constraints in the logic design and the layout concurrently with no loss of data.

If this directive is set in the `.cpm` file, the `OVERWRITE_CONSTRAINTS` directive is ignored during the front to back flow.

You can also lock the directive at the site level. This ensures that if changes are being made simultaneously in the logic design and the layout, then the layout changes are not overwritten. Define the directive in `site.cpm` as follows:

```
START_PKGRXL_CONTROL_SETTINGS

F2B_OVERWRITE_CONSTRAINTS 'LOCK'

END_PKGRXL_CONTROL_SETTINGS
```

Syntax

```
F2B_OVERWRITE_CONSTRAINTS 'ON'|'OFF'
```

Example

```
START_PKGRXL

F2B_OVERWRITE_CONSTRAINTS 'ON'
```

Corresponding UI Option for Design Entry HDL

None

FATAL_MESSAGES

Controls where to display the fatal messages flagged by Design Entry HDL.

Syntax

```
FATAL_MESSAGES 'DIALOG'|'COMMANDPANE'|'SUPPRESS'
```

where

DIALOG	If you set the value to Dialog, Design Entry HDL displays the messages in a dialog box.
COMMANDPANE	If you set this directive to COMMANDPANE, Design Entry HDL displays the messages in the Console Command Window.
SUPPRESS	If you set the variable to SUPPRESS Design Entry HDL does not display the type of messages.

Example

```
FATAL_MESSAGES 'DIALOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Messages — Fatal

See Also

- ERROR_MESSAGES
- INFORMATIONAL_MESSAGES
- WARNING_MESSAGES

FEEDBACK

The FEEDBACK directive enables you to run Packager-XL in feedback mode. This directive operates identically to the `-f` command line option.

If no state file exists, you must run Packager-XL in forward mode to create the state file before running the Packager-XL in the feedback mode. You can specify more than one `pst` feedback option. However in PCB Editor, only the PCB Editor option is required. The other options are for layout packages other than PCB Editor.

Syntax

```
FEEDBACK off|feedback_type[,feedback_type]...;
```

`off`	Packager-XL reads and packages the design in forward mode; it does not run in feedback mode.
`feedback_type`	Packager runs in feedback mode. The possible feedback types include: ■ `allegro` - Packager-XL reads the design and the state file, `pxl.state`, as well as output files generated by *a2fet*, and updates the design and the state file. ■ `pstfnet` - The file type is FEEDBACK_NETLIST. Packager-XL reads the design, the state file, the `pxl.state` file, and the `pstfnet.dat` file, and updates the state file and the design. ■ `pstprtx` - The file type is PART_TRANS. Packager-XL reads the design, the state file, the `pxl.state` file, and the `pstprtx.dat` file, and updates the design and the state file. ■ `pstsecx` - The file type is SECTION_TRANS. Packager-XL reads the design, the state file, the `pxl.state` file, and the `pstsecx.dat` file, and updates the design and the state file. ■ `pstnetx` - The file type is NETLIST_TRANS. Packager-XL reads the design, the state file, the `pxl.state` file, and the `pstnetx.dat` file, and updates the design and the state file.

The default value for the FEEDBACK directive is off.

Example

```
FEEDBACK allegro;
```

```
FEEDBACK pstprtx, pstsecx, pstnetx;
```

FIDUCIAL_FOOTPRINT_NAME_PATTERN

Defines footprint patterns that identify components as fiducials for the following rules in a schematic audit:

- `Fiducials not present`
- `Fiducials present is less than the minimum specified limit`

Syntax

```
FIDUCIAL_FOOTPRINT_NAME_PATTERN '<pin pattern>'
```

Example

```
FIDUCIAL_FOOTPRINT_NAME_PATTERN 'FID'
```

Corresponding UI Option in Allegro System Capture

Graphical Rule – Invalid Net Name – Configure – Schematic Audit Settings – Rules

FIDUCIAL_MIN_NUMBER

Defines the minimum number of fiducials that should be present in a design. If the design does not have the required number of fiducials, a violation is reported.

This directive is used when the `Fiducials present is less than the minimum specified limit` audit rule is used.

Syntax

```
FIDUCIAL_MIN_NUMBER '<Number>'
```

Example

```
FIDUCIAL_MIN_NUMBER '2'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Rule

FILTER_CONFLICTING_PROP

The FILTER_CONFLICTING_PROP directive specifies the names of the properties to be filtered from the `pstprop.dat` file. You can list any number of properties to be omitted.

Syntax

`FILTER_CONFLICTING_PROP` *`property`* `[,`*`property`*` ] ... ;`

`property`	Any property used in the Design Entry drawings.

The default value for the FILTER_CONFLICTING_PROP directive is none.

Example

```
FILTER_CONFLICTING_PROP SEC;
```

FILTER_PROP_CNS_IMPORT

The properties specified in this directive are filtered out during importing DE-HDL import design. These are DE-HDl specific properties not required in System Capture.

Syntax

```
FILTER_PROP_CNS_IMPORT '<property name1>' '<property name2>'
'<property name3>' '<property name4>'
```

Example

```
FILTER_PROP_CNS_IMPORT 'VLOG_PARAM' 'VLOG_PARAM01' 'VLOG_PARAM02'
'VLOG_PARAM03' 'VLOG_PARAM04' 'VLOG_PARAM05' 'VLOG_PARAM06'
'VLOG_PARAM07' 'VLOG_PARAM08' 'VLOG_PARAM09'
```

FILTER_PROPERTY

The FILTER_PROPERTY directive specifies the properties to be omitted from the output files. You can list any number of properties to be omitted. You can enter the FILTER_PROPERTY directive as many times as needed in the project file.

Syntax

```
FILTER_PROPERTY property [,property ] ... ;
```

property	Any property used in the Design Entry drawings.

The default value for the FILTER_PROPERTY directive is none.

Example

```
FILTER_PROPERTY drawing, dir, xy, ver;
```

FilterZeroNet

Controls the flow of zero node nets from System Capture to PCB Editor. By default, this directive is set to `'ON'`. Consequently, all zero nodes in front-end are filtered and not passed on to the back-end in the front-to-back flow. This prevents these zero node nets from being reported as differences. To suppress this behavior and include zero node nets in differences, change the value for this directive to `'OFF'`.

Syntax

```
FilterZeroNet 'OFF'|'ON'
```

Example

```
FilterZeroNet 'OFF'
```

Note: This directive is available from ISR 44 onwards.

FLATTEN_BLOCK_ON_ADD

If you would like to use the sheets of a hierarchical block in a flat design, you can set this directive. After setting it, when you add a block instance, the sheets of the block are added to the current design. Note that the design hierarchy is not added to the design; only the flat schematic sheets. The block that you want to add to a design must be a single level of hierarchy. This directive is not supported for Design Entry HDL scripts in a nongraphical mode.

With this directive ON, when you click the Add button in Part Information Manager and select a flat block, the Import Design dialog is opened. After you select the block that you want to import, all the pages in the block are automatically selected. You cannot deselect any of these pages. When you then click the Import button, the Import Design dialog is displayed with the destination design. The page before which the sheets will be added is also already populated.

When you click OK in this dialog, the sheets in the block are imported flattening the block pages in the current design.

If you use this option, make a note of the following points:

- Constraint data is lost for the objects in the block being flattened.
- If port objects were specified in the block which is being flattened, these ports will be irrelevant to the root design.
- No block reuse data will be used for packaging the flattened sheets.
- Sheet import will fail if read-only sheets are present after the current page.

Syntax

```
FLATTEN_BLOCK_ON_ADD 'ON'|'OFF'
```

Example

```
FLATTEN_BLOCK_ON_ADD 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

FORCE_PTF_ENTRY

The FORCE_PTF_ENTRY directive enables Packager-XL to verify that the ppt files are present in the cell view for all instances, and a ppt entry is defined for each instance in the ppt file.

Syntax

```
FORCE_PTF_ENTRY 'OFF'|'ON'
```

on	When the value is set to `on`, Packager-XL verifies that the ppt files are present in the cell view for all instances, and a ppt entry is defined for each instance in the ppt file.
off	When the value is set to `off`, Packager-XL does not verify whether or not the ppt files are present in the cell view for all instances. Packager-XL will also not verify whether or not a ppt entry is defined for each instance in the ppt file.

FORCE_SUBDESIGN

The FORCE_SUBDESIGN directive reads the corresponding subdesign state file and applies packaging from the state file to every instance of the subdesign in the top-level design.

This is the recommended way to use subdesigns. If you have made changes to the subdesign, these changes are propagated to all instances of the subdesign.

In the feedback mode, instances that have this directive applied on them read the subdesign state file and revert to the value they had in the schematic and ignore any new value that PCB Editor might have assigned to them.

Syntax

```
FORCE_SUBDESIGN subdesign[,subdesign ] ... ;
```

subdesign	A subdesign (hierarchical block) for which a state file has been previously created by using the GEN_SUBDESIGN directive. The subdesign name is the same as the drawing name used in Design Entry.

The default value for the FORCE_SUBDESIGN directive is none.

Example

```
FORCE_SUBDESIGN counter;
```

FORMAT_CREF_REPORTS

When this directive is set to OFF, CRefer creates text reports for basenets, netsbypage, and synonyms with all columns starting on different rows causing increased number of schematic report pages. Further, setting the `FORMAT_CREF_REPORTS` directive to `OFF` prevents zone wrapping.

By default, this directive is set to ON.

Note: It is recommended that you avoid changing the value of the `FORMAT_CREF_REPORTS` directive.

Value

```
FORMAT_CREF_REPORTS 'ON'|'OFF'
```

Example

```
FORMAT_CREF_REPORTS 'ON'
```

9

G Directives

This chapter lists the CPM directives that start with `G` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

GEN_PSTFILES

Use this directive to specify whether to package the System Connectivity Manager design before exporting it to the physical layout tool.

Syntax

```
gen_pstfiles 'TRUE'|'FALSE'|'1'|'0'
```

Example

```
gen_pstfiles 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project —Export — Physical — Generate Package file

See Also

- ETCH_REMOVAL
- IGNORE_FIXED
- LAUNCH_OPTION
- OVERWRITE_CONSTRAINTS

GEN_SUBDESIGN

The GEN_SUBDESIGN directive is used to specify the modules (hierarchical blocks) for which you want to generate subdesign state files. If Packager-XL finds an instance of a subdesign with the SUBDESIGN_SUFFIX property, it uses that instance as the source for generating the subdesign state file. Otherwise, it uses the first instance of the subdesign that it comes across as the source for generating the subdesign state file.

Syntax

```
GEN_SUBDESIGN subdesign[,subdesign ] ...;
```

subdesign	A hierarchical block name. The subdesign name is the same as the drawing name used in Design Entry.

The default value for the GEN_SUBDESIGN directive is none.

Example

```
GEN_SUBDESIGN counter;
```

This creates a subdesign state file called `pxl_COUNTER.state`.

GENERATE_FLATTENED_SCHEMATIC

Creates a new flattened view (`schcref_1`) view in the top-level cell for the current project for the cross referenced design.

Syntax

```
GENERATE_FLATTENED_SCHEMATIC 'ON'|'OFF'
```

Example

```
GENERATE_FLATTENED_SCHEMATIC 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Content page — Run Options — Generate Flattened Schematic

GENERATE_SCH_METADATA

Enables generation of schematic-related metadata in Allegro Design Entry HDL, by default.

Syntax

```
GENERATE_SCH_METADATA 'ON'|'OFF'
```

Example

```
GENERATE_SCH_METADATA 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Metadata Options page — Schematic Metadata and Revision Check Options — Generate Schematic Metadata

See Also

- SYNC_ON_STARTUP
- SYNC_ON_PAGE_EDIT

GENERATE_SEPARATE_CELL

Creates a separate cell structure, into the sch_1 view of which the CRefer report pages are added.

Syntax

```
GENERATE_SEPARATE_CELL 'ON'|'OFF'
```

Example

```
GENERATE_SEPARATE_CELL 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Reports page — Add as a Separate Cell

GENERATE_TDD_NETLIST

Generates con/dcf files in the sch_1 folder which will be used by System Connectivity Manager/Allegro System Architect for integration of Design Entry HDL blocks in System Connectivity Manager.

Syntax

```
GENERATE_TDD_NETLIST 'ON'|'OFF'
```

Example

```
GENERATE_TDD_NETLIST 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Allegro System Architect — Generate Connectivity and Property Files

GENERATE_XR_FOR_ALL_NETS

Generates cross references for all the nets in the design. The cross references generated after selecting this option contain data considering nets from all levels of the hierarchy. You can view and navigate to the nets from all the levels of a hierarchical design.

Syntax

```
GENERATE_XR_FOR_ALL_NETS 'ON'|'OFF'
```

Example

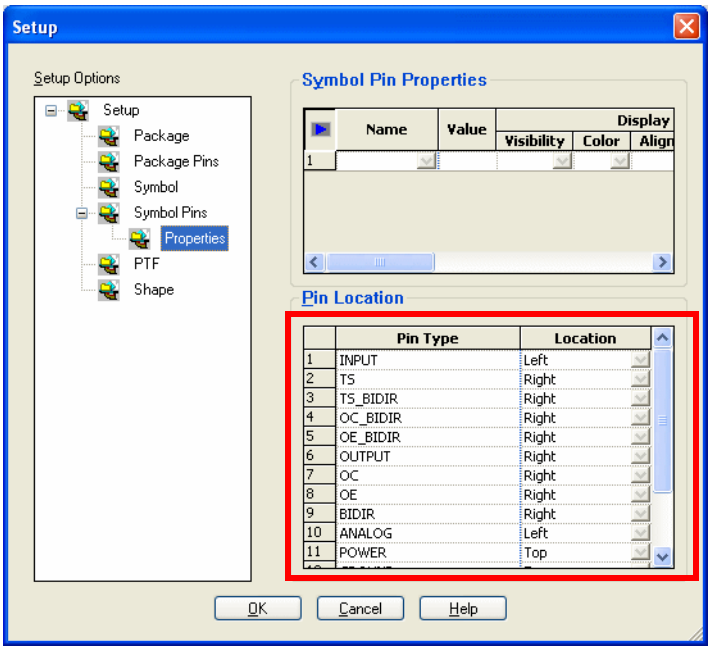
```
GENERATE_XR_FOR_ALL_NETS 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Content page — Generate Cross References for all nets

Global_Modify_Pin_Graphics

The `Global_Modify_Pin_Graphics` directive specifies if the pin type and pin location association set on the symbol pin properties page in Setup is to be applied when a pin type is changed.



Syntax

```
Global_Modify_Pin_Graphics 'Yes'|'No'
```

Example

```
Global_Modify_Pin_Graphics 'Yes'
```

Corresponding UI Option for Part Developer

None

GLOBAL_NETS

Defines the pattern for the global nets in a design. Global net connections are used to identify power and ground nets connected to various power and ground pins.

Syntax

```
GLOBAL_NETS '<pattern 1>' '<pattern 2>'... '<pattern N>'
```

The following pin patters are supported:

```
.*VCC.*,.*VBB.*,.*VTT.*,.*VIN.*,.*V.*[0-9]P[0-9].*|
```

Example

```
GLOBAL_NETS 'NC' 'POWER' '.*VDD.*' '.+[0-9]V.*'
```

Corresponding UI Option in Allegro System Capture

None

GRAPHIC_BLOCK_FILL_COLOR

Sets the default fill color for graphic blocks.

Syntax

```
GRAPHIC_BLOCK_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color for graphic blocks. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FFFFFF'` (White).

Examples

```
GRAPHIC_BLOCK_FILL_COLOR '#E8EFF7'
```

GRAPHIC_BLOCK_LINE_COLOR

Sets the line color for graphic blocks.

Syntax

```
GRAPHIC_BLOCK_LINE_COLOR '<line_color>'
```

line_color	The default line color for graphic blocks. The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue. The default color is `'#FCD054'`

Example

```
GRAPHIC_BLOCK_LINE_COLOR '#0000CC'
```

GRAPHIC_CONNECTOR_FILL_COLOR

Sets the default fill color for graphic blocks.

Syntax

```
GRAPHIC_CONNECTOR_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color for graphic connectors. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FFFFFF'` (White).

Examples

```
GRAPHIC_CONNECTOR_FILL_COLOR '#E8EFF7'
```

GRAPHIC_CONNECTOR_LINE_COLOR

Sets the line color for graphic connectors.

Syntax

```
GRAPHIC_CONNECTOR_LINE_COLOR '<line_color>'
```

line_color	The default line color for graphic connectors. The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue. The default color is `'#0000FF'`

Example

```
GRAPHIC_CONNECTOR_LINE_COLOR '#0000CC'
```

GRID_DISPLAY_ENABLED

Specifies which grid to display, electrical, documentation, or both the grids together.

Syntax

```
GRID_DISPLAY_ENABLED 'documentation'|'electrical'|'both'| 'LOCK'
```

where,

- `DOCUMENTATION` grid is a fine grid and provides neat placement of drawing objects such as text.
- `ELECTRICAL` grid is coarser compared to the `DOCUMENTATION` grid. It is used for placing components and wires.

Example

```
GRID_DISPLAY_ENABLED 'both'
GRID_DISPLAY_ENABLED 'documentation'
GRID_DISPLAY_ENABLED 'electrical'
```

GRID_DISPLAY_MULTIPLE

Displays every nth grid line to define where objects can be placed on the electrical grid. This ensures the correct connectivity of wires and symbols.

Syntax

```
GRID_DISPLAY_MULTIPLE '<value>'
```

where

value	1, 2, 5, 10, 50.

Example

```
GRID_DISPLAY_MULTIPLE '5'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Grid - Display Electrical Grid

GRID_DOC_SNAP_FRACTION

Defines the documentation grid with respect to the pin-to-pin spacing. The doc snap fraction defines how far each documentation grid is placed on the canvas in terms of pin-to-pin spacing.

Syntax

```
GRID_DOC_SNAP_FRACTION '<snap_fraction_value>'
```

Example

```
GRID_DOC_SNAP_FRACTION '0.1'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Grid - Electrical Grid

GRID_PIN_PITCH

Sets the pin pitch or the pin-to-pin spacing of a grid. This variable also controls the pin-to-pin spacing for all new designs when set in the `site.cpm` file.

Syntax

```
GRID_PIN_PITCH '<pin_pitch>'
```

Example

```
GRID_PIN_PITCH '0.1'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Grid - Pin-to-Pin Spacing

GRID_PIN_PITCH

Sets the pin pitch or the pin-to-pin spacing of a grid.

Syntax

```
GRID_PIN_PITCH '<pin_pitch>'
```

Example

```
GRID_PIN_PITCH '0.1'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Grid - Pin-to-Pin Spacing

GRID_SNAP_FRACTION

Defines the electrical grid with respect to the pin-to-pin spacing. The snap fraction defines how far each electrical grid is placed on the canvas in terms of pin-to-pin spacing. This variable also controls the electrical grid to the pin-to-pin spacing for all new designs when set in the `site.cpm` file.

Syntax

```
GRID_SNAP_FRACTION '<snap_fraction_value>'
```

Example

```
GRID_SNAP_FRACTION '0.5'
```

GRID_STYLE

Specifies which grid style to display the grid, *lines* or *dots*. You can choose the appropriate grid style based on the task being performed.

Syntax

```
GRID_STYLE 'LINES'|'DOTS'
```

where,

- `LINES` is the default value. By default, the grid displays as a rectangular patterns of lines. To place and align components, use Lines.
- `DOTS` displays the grid as patterns of dots at various intervals.

Example

```
GRID_STYLE 'LINES'
GRID_STYLE 'DOTS'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Grid - Grid Style

GRID_UNIT_MEASURE

Specifies whether to measure grid units in inches or millimeters. The basic grid unit is defined as a factor of pin-to-in spacing or pin pitch. The default value is `INCHES`.

This variable also controls the grid units for all new designs when set in the `site.cpm` file. For example, if your libraries use millimeters and you want all new designs to be configured for millimeters, set `GRID_UNIT_MEASURE` to '`MILLIMETERS`'

Syntax

```
GRID_UNIT_MEASURE 'INCHES'|'MILLIMETERS'
```

Example

```
GRID_UNIT_MEASURE 'INCHES'
GRID_UNIT_MEASURE 'MILLIMETERS'
```

GND_PIN_NAME

Defines the ground pin names in a design when the following audit rules are run:

- `Grounded IC Output Pins`
- `IC Output Pins Without Receiver`
- `Unconnected IC Power/Ground Pins`
- `IC Input Pins Without Driver`
- `Nets connected to SDA pins missing a pull-up`
- `Nets connected to SCL pins missing a pull-up`
- `Open collector output pin missing a pull-up`
- `All IC Input Pins Pulled Up`
- `All IC Input Pins Pulled Down`
- `Low Pullup Resistance Value`
- `High Pulldown Resistance Value`
- `High Pullup Resistance Value`
- `Low Pulldown Resistance Value`
- `Bypass capacitor voltage exceeds rated voltage`
- `Same group power pins connected to nets with different voltage values`
- `Power Nets Without Voltage`
- `Power Nets With 0V`
- `Ground Nets Without Voltage`
- `Ground Nets With Non-Zero Voltage`
- `Missing Bypass Capacitors`
- `Nets with pull-up and pull-down resistors`

Design Integrity identifies pins as ground pins based on the following conditions:

- If the pin name matches the patterns defined in this directive

- If the pin type is defined as ground in `chips.prt`

Syntax

```
GND_PIN_NAME '<pin pattern>'
```

The following pin patterns are supported:

```
GNDAPLL, GNDUSB, GROUND, PGND, AGND, 0, ACOM, DCOM, CGND, DGND
```

Example

```
GND_PIN_NAME 'GND' 'GNDADC'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Power or *Ground* tab – *Ground Pins*

GUIDE_LINES_COLOR_DARK

Using this directive, you can set the color to be used for *Guide Lines* while moving or adding parts in System Capture when using Dark Theme.

Syntax

```
GUIDE_LINES_COLOR_DARK '<color_hex_code_value>'
```

where

`color_hex_code_value`	Specify the hex code of the color. Default value is '#ffffff' (White)

Example

```
GUIDE_LINES_COLOR_DARK '#ffffff'
```

If the above value is specified, white color is used for *Guide Lines* while moving or adding parts in System Capture when using Dark Theme.

GUIDE_LINES_COLOR_LIGHT

Using this directive, you can set the color to be used for *Guide Lines* while moving or adding parts in System Capture when using Light Theme.

Syntax

```
GUIDE_LINES_COLOR_LIGHT '<line_color>'
```

where

`line_color`	The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue. The default value is '#000000' (Black).

Example

```
GUIDE_LINES_COLOR_LIGHT '#000000'
```

If the above value is specified, Black color is used for *Guide Lines* while moving or adding parts in System Capture when using Dark Theme.

10

H Directives

This chapter lists the CPM directives that start with `H` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

HARD_LOC_SEC

The HARD_LOC_SEC directive is used to distinguish between soft properties and hard properties for packaging purposes in feedback mode.

Syntax

```
HARD_LOC_SEC 'off' | 'on'
```

on	When the value is set to on, all hard properties retain their values during packaging in feedback mode. An attempt to change their property values will generate a warning message, which is recorded in the pxl.log file. However, soft properties can be changed during packaging.
off	This is the default value of the HARD_LOC_SEC directive. When set to off, the values of both hard and soft properties can be changed during packaging in feedback mode.

Note: If the STATE_WINS_OVER_DESIGN directive is set to on, then irrespective of the value of the HARD_LOC_SEC directive, all properties, whether hard or soft, retain their values during packaging. Therefore, you can change the values of soft properties only if the STATE_WINS_OVER_DESIGN directive is set to off. You can change the values of hard properties only when both the STATE_WINS_OVER_DESIGN directive and HARD_LOC_SEC directive are set to off.

Note: If you have used the LOCATION or ROOM or HARD property to group together instances, then all instances will be treated as hard properties. You will have to set the value of the HARD_LOC_SEC directive to on to ensure that the packaging of these properties is retained.

Note: If the user specifies the LOCATION property as LOCATION=?, it is treated as a hard property. If the user wants PXL to assign the LOCATION property on its own, then a placeholder should be specified as $LOCATION=?.

HARD_REFDES_CONFLICT_RESOLVE

This directive allows you to manually resolve reference designator conflicts.

When a reference designator conflicts arises, Allegro System Capture resolves the conflict by automatically assigning a unique refdes to the part that is already placed on the schematic. The most recent user assigned refdes always wins.

If this directive is set to OFF, an error message is displayed in *Violation Window* when a refdes conflict arises and you can manually resolve the conflict.

Syntax

```
HARD_REFDES_CONFLICT_RESOLVE 'OFF'|'ON'
```

OFF	An error message is received whenever there is a reference designator conflict. This conflict must be resolved manually.
ON	Reference designator conflicts are automatically resolved by assigning unique refdes to the part that is already on the canvas.

Example

```
HARD_REFDES_CONFLICT_RESOLVE 'OFF'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Auto-resolve RefDes Conflict

HIDE_HIERARCHY_PAGES

Shows or hides hierarchy page name for different pages in the design.

Syntax

```
HIDE_HIERARCHY_PAGES 'ON'|'OFF'
```

Example

```
HIDE_HIERARCHY_PAGES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Design Navigation page — Hierarchy Viewer Options — Show Hierarchy Pages

See Also

- HIDE_INSTANCE_NAME
- HIDE_SHEET_NUMBER

HIDE_INSTANCE_NAME

Hides instance names from the design hierarchy in hierarchy viewer.

Syntax

```
HIDE_HIERARCHY_PAGES 'ON'|'OFF'
```

Example

```
HIDE_HIERARCHY_PAGES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Design Navigation page — Hierarchy Viewer Options — Hide Instance Names

See Also

- HIDE_HIERARCHY_PAGES
- HIDE_SHEET_NUMBER

HIDE_SHEET_NUMBER

Hides sheet numbers from the design hierarchy in hierarchy viewer.

Syntax

```
HIDE_SHEET_NUMBER 'ON'|'OFF'
```

Example

```
HIDE_SHEET_NUMBER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Design Navigation page — Hierarchy Viewer Options — Hide Instance Number

See Also

- HIDE_INSTANCE_NAME
- HIDE_HIERARCHY_PAGES

HIGHLIGHT_COLOR

Changes the default highlight color.

Syntax

```
HIGHLIGHT_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
HIGHLIGHT_COLOR 'DEFAULT'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Highlight

See Also

- ARC_COLOR
- BACKGROUND_COLOR
- DOT_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

HOLE_FOOTPRINT_NAME_PATTERN

Defines the footprint patterns used for identifying components as holes.

This directive is used when the `Holes not present` audit rule is run.

The part instances in the design are checked and the JEDEC type of the part instances are compared with the listed patterns. If no holes are found, a violation is reported.

Syntax

```
HOLE_FOOTPRINT_NAME_PATTERN '<pattern>'
```

Example

```
HOLE_FOOTPRINT_NAME_PATTERN 'HOLE'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Rule

HONOR_PPTOPTION_ON_ADD_OR_REPLACE

When this directive is set to 'ON', annotation and visibility of injected properties is controlled by the PPT option set. When replacing a component, DE-HDL honors the PPT option set over the current annotation on the schematic canvas.

Syntax

```
HONOR_PPTOPTION_ON_ADD_OR_REPLACE 'ON'|'OFF'
```

Example

```
HONOR_PPTOPTION_ON_ADD_OR_REPLACE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

HPF_BATCH

Allows you to plot the design in batch mode from the console window. You need to setup the HPF plotting options and then plot the design to a single file. The default filename is `vw.spool`.

Syntax

```
HPF_BATCH 'YES'|'NO'
```

Example

```
HPF_BATCH 'YES'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Plotter Options — Plot to File

See Also

- HPF_BUS_SCALEFACTOR
- HPF_FONT
- HPF_PAGESIZE
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE
- HPF_WIRE_SCALEFACTOR

HPF_BUS_SCALEFACTOR

Specifies the width of buses in the design.

Syntax

```
HPF_BUS_SCALEFACTOR '<value>'
```

where

value any number

Example

```
HPF_BUS_SCALEFACTOR '3'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Line Width — Bus Scale

See Also

- HPF_BATCH
- HPF_FONT
- HPF_PAGESIZE
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE
- HPF_WIRE_SCALEFACTOR

HPF_FONT

Specifies the font style to be used for plotting.

Syntax

```
HPF_FONT '<value>'
```

where

value	any supported font

Example

```
HPF_FONT 'ARIAL'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Plotter Options — Font

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE

HPF_PAGESIZE

Specifies the standard page to which the design gets scaled.

Syntax

```
HPF_PAGESIZE 'A'|'B'|'C'|'D'|'E'|'F'
```

Example

```
HPF_PAGESIZE 'A'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Page Scaling — Scale to Page Size list box

See Also

- HPF_BUS_SCALEFACTOR
- HPF_FONT
- HPF_BATCH
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE
- HPF_WIRE_SCALEFACTOR

HPF_PLOT_PAGESIZE

Specifies the paper on which to plot.

Syntax

```
HPF_PLOT_PAGESIZE '<paper_size>'
```

where

paper_size a positive number.

Example

```
HPF_PLOT_PAGESIZE '2'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Plotter Options — Specify Paper Size field

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_FONT
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE

HPF_PLOTTER

Specifies the name of the plotter in the `.cdsplotinit` file.

Syntax

```
HPF_PLOTTER '<plotter_name>'
```

where

`plotter_name` name of the plotter

Example

```
HPF_PLOTTER 'printer1_2a'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Plotter Options — Plotter

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_FONT
- HPF_PLOT_PAGESIZE
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE

HPF_SCALEFACTOR

Specifies the scaling factor. Scaling factor is a factor applied to the drawing to determine the final plot size. The default factor is 1.

Syntax

```
HPF_SCALEFACTOR '<scale_factor>'
```

where

`scale_factor`	a positive number

Example

```
HPF_SCALEFACTOR '2'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Page Scaling — Scale by Factor

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_FONT
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE

HPF_SCALETYPE

Specifies the mode of scaling to be used for plotting.

Syntax

```
HPF_SCALETYPE 'Default'|'Scale by factor'|'Scale to Page Size'
```

Example

```
HPF_SCALETYPE 'Default'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Page Scaling — Default| Scale by Factor| Scale by Page Size

See Also

- HPF_BUS_SCALEFACTOR
- HPF_FONT
- HPF_BATCH
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_PAGESIZE
- HPF_SPEC_PLOT_PAGESIZE
- HPF_WIRE_SCALEFACTOR

HPF_SPEC_PLOT_PAGESIZE

Sets the paper size.

Syntax

```
HPF_SPEC_PLOT_PAGESIZE 'YES'|'NO'
```

Example

```
HPF_SPEC_PLOT_PAGESIZE 'YES'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Plotter Options — Specify Paper Size check box

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_FONT
- HPF_PLOTTER
- HPF_SCALEFACTOR
- HPF_SCALETYPE
- HPF_PLOT_PAGESIZE

HPF_WIRE_SCALEFACTOR

Specifies the width of wires, text, and component boundaries in the design.

Syntax

```
HPF_WIRE_SCALEFACTOR '<scale_factor>'
```

where

`scale_factor`	a positive number

Example

```
HPF_WIRE_SCALEFACTOR '2'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — HPF — Line Width — Wire Scale Factor

See Also

- HPF_BUS_SCALEFACTOR
- HPF_BATCH
- HPF_PAGESIZE
- HPF_FONT
- HPF_PLOT_PAGESIZE
- HPF_PLOTTER
- HPF_SCALETYPE
- HPF_SPEC_PLOT_PAGESIZE

HYPERLINK_HIGHLIGHTED_DARK_THEME_COLOR

Using this directive, you can set the color for highlighting hyperlinks in Dark Theme.

Syntax

```
HYPERLINK_HIGHLIGHTED_DARK_THEME_COLOR '<hyperlink_color>'
```

hyperlink_color The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue.

The default value is "#00BFFF" (Deep Sky Blue).

Example

```
HYPERLINK_HIGHLIGHTED_DARK_THEME_COLOR '#00BFFF'
```

HYPERLINK_HIGHLIGHTED_LIGHT_THEME_COLOR

Using this directive, you can set the color for highlighting hyperlinks in Light Theme.

Syntax

```
HYPERLINK_HIGHLIGHTED_LIGHT_THEME_COLOR '<hyperlink_color>'
```

hyperlink_color The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue.

The default value is '#0000FF' (Blue).

Example

```
HYPERLINK_HIGHLIGHTED_LIGHT_THEME_COLOR '#0000FF'
```

HYPERLINK_VISTED_COLOR

Using this directive, you can set the color of visited hyperlinks.

Syntax

```
HYPERLINK_VISTED_COLOR '<hyperlink_color>'
```

hyperlink_color The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue.

The default value is `'#820099'` (Dark Magenta).

Example

```
HYPERLINK_VISTED_COLOR '#820099'
```

HYPERLINKS

Activates cross-references in a design as hyperlinks. By default, this directive is set to ON and cross-references in your design are active links.

Syntax

```
HYPERLINKS 'ON'|'OFF'
```

Example

```
HYPERLINKS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

11

I Directives

This chapter lists the CPM directives that start with `I` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL

IGNORE_BLOCK_WITHOUT_SCHEMATIC

Use this directive to specify if an the schematic generation process should be stopped and an error reported when the block for which the document schematic is to be created does not have an existing document schematic.

If this is set to 1, the document schematic is generated with a warning message.

Syntax

```
ignore_block_without_schematic <0 or 1>
```

Example

```
ignore_block_without_schematic '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — General — Stop document schematic generation for the project if block level document schematic is not available.

IGNORE_BUNDLED_CONSTRAINTS_ERROR_ON_ZERO _NODE_NETS

Release 16.5 onwards, constraints are defined and captured in Constraint Manager, instead of in the schematic. As a result, when you uprev a pre–16.5 design to 16.5 or a later release, bundled constraints in the schematic sheets are upreved to the constraints database, that is, the `.dcf`. After the uprev, all these constraints are removed from the schematic sheet.

However, bundled constrains on nets with zero nodes are not removed from the schematic leading to errors when you try and save schematic pages.

Bundled constraints are constraints that have more than one property in the `.dcf` file. For example, `PROPAGATION_DELAY` can have two properties: `MIN_PROPAGATION_DELAY` and `MAX_PROPAGATION_DELAY`.

Note: `NET_SPACING_TYPE` and `NET_PHYSICAL_TYPE` could not be specified as sync constraints prior to 16.5 since both represented a group of nets in the `.dcf` file. As result, they are also treated as bundled constraints in DE-HDL.

When this directive is `ON`, DE-HDL ignores `NET_SPACING_TYPE`, `NET_PHYSICAL_TYPE`, and bundled constraints errors on zero-node nets in the design connectivity, allowing a design to be upreved. The directive should be added to the `START_NETLIST` section of the `.cpm` file.

Syntax

```
ignore_bundled_constraints_error_on_zero_node_nets 'ON'|'OFF'
```

Example

```
ignore_bundled_constraints_error_on_zero_node_nets 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

IGNORE_FIXED

This directive corresponds to the *Ignore FIXED Property* check box in the Export Physical dialog.

Use this directive to define whether components with the fixed property can be moved or deleted (or ripped up if assigned on a net) when a schematic is imported.

Selecting the Ignore FIXED Property option means that the fixed property attached to components or nets can be ignored during the import and that parts or nets from the database that are not in the netlist can be moved, deleted, or ripped up.

Syntax

```
ignore_fixed 'YES'|'NO'|'TRUE'|'FALSE'|'ON'|'OFF'|'1'|'0'
```

Example

```
ignore_fixed 'NO'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Ignore FIXED property check box

See Also

- ETCH_REMOVAL
- GEN_PSTFILES
- LAUNCH_OPTION
- OVERWRITE_CONSTRAINTS

IGNORE_INSTANCE_WITH_ERRORS

Use this directive to stop the schematic generation process and generate an error if problems are encountered due to an incorrect symbol or packaging error.

If this is set to 1, the document schematic is generated with a warning message.

Syntax

```
ignore_instance_with_errors <0 or 1>
```

Example

```
ignore_instance_with_errors '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — General — Report an error if proper symbol is not found for any instance in the design or if the instance has packaging errors

IGNORE_VAR_STATUS_COL

Set this directive to ON if you do not want the STATUS column to be included in the BOM reports for base schematics and variants.

Syntax

```
IGNORE_VAR_STATUS_COL 'ON'|'OFF';
```

Example

```
IGNORE_VAR_STATUS_COL 'ON';
```

ILLEGAL_NET_NAME_CHAR

Defines the patterns to identify invalid net names.

This directive is used when the `Invalid net name` audit rule under the *Connectivity Checks* category is run.

Specify an empty value if no invalid characters are allowed. Else, the invalid characters listed in the default cpm `cds.cpm`, are used.

Syntax

```
ILLEGAL_NET_NAME_CHAR '<pin pattern>'
```

Example

```
ILLEGAL_NET_NAME_CHAR '@' '='
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Rule*

IMAGE* Directives

Sets the graphic and background colors, as specified in Graphic Color and Background Color boxes, for an image of the schematic captured and placed on the clipboard.

See Also

IMAGE_ARC_COLOR

IMAGE_BACKGROUND_COLOR

IMAGE_DEFAULT_DPI

IMAGE_DOT_COLOR

IMAGE_NOTE_COLOR

IMAGE_OCCPROP_COLOR

IMAGE_PROP_COLOR

IMAGE_SYMBOL_COLOR

IMAGE_WIRE_COLOR

IMAGE_ARC_COLOR

Changes the default arc color of image captured by choosing *Edit – Image – Capture*.

Syntax

```
IMAGE_ARC_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_ARC_COLOR 'yellow'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Arc

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_PROP_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IMAGE_BACKGROUND_COLOR

Changes the default color of the drawing area.

Syntax

```
IMAGE_BACKGROUND_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_BACKGROUND_COLOR 'WHITE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Background

See Also

- IMAGE_ARC_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_PROP_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IMAGE_COLOR_MODE_PRINT

When printing a design or screenshot in black and white, some issues might appear, such as lines are not shown and text is missing. Set this directive to print images in color regardless of the mode selected in the print dialog.

Syntax

```
IMAGE_COLOR_MODE_PRINT 'true' | 'false'
```

Example

```
IMAGE_COLOR_MODE_PRINT 'true'
```

Corresponding UI Option for Allegro System Capture

None

See Also

MONOCHROME_PRINTING_THRESHOLD

IMAGE_DEFAULT_DPI

Changes the dpi value of an image pasted on the schematic. When you paste an image, the resolution of the image is set to 72 dots per inch (dpi). The dpi value defines the resolution or the pixel density of the image pasted on the schematic canvas. If you increase this value, the size of the graphic reduces resulting in sharper images.

Syntax

```
IMAGE_DEFAULT_DPI '<DPI_value>'
```

Example

```
IMAGE_DEFAULT_DPI '150'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

IMAGE_ARC_COLOR

IMAGE_BACKGROUND_COLOR

IMAGE_DEFAULT_DPI

IMAGE_DOT_COLOR

IMAGE_NOTE_COLOR

IMAGE_OCCPROP_COLOR

IMAGE_PROP_COLOR

IMAGE_SYMBOL_COLOR

IMAGE_WIRE_COLOR

IMAGE_DOT_COLOR

Changes the default dot color of image captured by choosing *Edit – Image – Capture*.

Syntax

```
IMAGE_DOT_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_DOT_COLOR 'PINK'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Dot

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_ARC_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_PROP_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_ WHEN_PASTING

Controls if the power signal names get transfered along or not when wires are copied and pasted without the attached power source in System Capture. This directive is set to `TRUE` by default

Syntax

```
IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING 'TRUE' | 'FALSE'
```

Example

```
IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING 'TRUE'
```

Corresponding UI Option for Allegro System Capture

Edit — Preferences — Schematic — General — Ignore hidden scalar power and global signals when pasting

See Also

- IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_ PASTING
- IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_ NAMES_WHEN_PASTING
- IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING

IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_PASTING

Controls the transfer of all hidden signal names when a selection is pasted. This directive is set to `TRUE` by default.

Syntax

```
IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_PASTING 'TRUE' | 'FALSE'
```

Example

```
IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_PASTING 'TRUE'
```

Corresponding UI Option for Allegro System Capture

Edit — Preferences — Schematic — General — Ignore all hidden named scalar signal names when pasting

See Also

- IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING
- IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_ NAMES_WHEN_PASTING
- IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING

IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_ NAMES_WHEN_PASTING

Controls the transfer of the hidden ‘unnamed_*’ signal names when a selection is pasted. This directive is set to `TRUE` by default.

Syntax

```
IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_NAMES_WHEN_PASTING 'TRUE' | 'FALSE'
```

Example

```
IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_NAMES_WHEN_PASTING 'TRUE'
```

Corresponding UI Option for Allegro System Capture

Edit — Preferences — Schematic — General — Ignore hidden ‘unnamed’ scalar signal names when pasting

See Also

- IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING
- IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_ PASTING
- IGNORE_HIDDEN_SCALAR_POWER_SIGNAL_NAMES_WHEN_PASTING

IMAGE_NOTE_COLOR

Changes the default note color of image captured by choosing *Edit – Image – Capture*.

Syntax

```
IMAGE_NOTE_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_NOTE_COLOR 'YELLOW'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Note

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_ARC_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_PROP_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IMAGE_OCCPROP_COLOR

Changes the Occurrence Property color.

Syntax

```
IMAGE_OCCPROP_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_OCCPROP_COLOR 'RED'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Occurrence property

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_ARC_COLOR
- IMAGE_PROP_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IMAGE_PROP_COLOR

Changes the default property color.

Syntax

```
IMAGE_PROP_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_PROP_COLOR 'Green'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Property

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_ARC_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_WIRE_COLOR

IMAGE_SYMBOL_COLOR

Changes the default symbol/body color.

Syntax

```
IMAGE_SYMBOL_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_SYMBOL_COLOR 'Green'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Symbol

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_ARC_COLOR
- IMAGE_PROP_COLOR
- IMAGE_WIRE_COLOR

IMAGE_WIRE_COLOR

Changes the default wire color.

Syntax

```
IMAGE_WIRE_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
IMAGE_WIRE_COLOR 'DEFAULT'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Color Settings — Image Capture — Graphics Color — Wire

See Also

- IMAGE_BACKGROUND_COLOR
- IMAGE_DEFAULT_DPI
- IMAGE_DOT_COLOR
- IMAGE_NOTE_COLOR
- IMAGE_OCCPROP_COLOR
- IMAGE_ARC_COLOR
- IMAGE_SYMBOL_COLOR
- IMAGE_PROP_COLOR

Import_AllegroFtprint_DefaultPinType

The `Import_AllegroFtprint_DefaultPinType` directive specifies the default pin type assigned in Allegro footprint import.

Syntax

```
Import_AllegroFtprint_DefaultPinType '<pin_type>'
```

Example

```
Import_AllegroFtprint_DefaultPinType 'BIDIR'
```

Corresponding UI Option for Part Developer

None

Import_APD_PwrGndNCPinsInGlobalSection

The `Import_APD_PwrGndNCPinsInGlobalSection` directive determines whether POWER, GROUND, and NC pins are to be imported as global pins in APD import.

The following values are supported:

- TRUE

  POWER, GROUND, and NC pins are imported as global pins; these pins are placed in the Global Pins grid.

- FALSE

  POWER, GROUND, and NC pins are imported as logical pins; these pins are placed in the Logical Pins grid.

Syntax

```
Import_APD_PwrGndNCPinsInGlobalSection 'TRUE'|'FALSE'
```

Example

```
Import_APD_PwrGndNCPinsInGlobalSection 'TRUE'
```

Corresponding UI Option for Part Developer

None

Import_APD_strippinnum

The `Import_APD_strippinnum` directive determines if pin names are to be retained in *`<logical_name>_<physical_name>`* format or split into logical pin names and physical pin numbers when a part is created using the Import APD Component Files option.

The following values are supported:

- TRUE

  Splits pin names in *`<logical_name>_<physical_name>`* format into logical pin names and physical pin numbers

- FALSE

  Retains pin names in *`<logical_name>_<physical_name>`* format

Syntax

```
Import_APD_strippinnum 'TRUE'|'FALSE'
```

Example

```
Import_APD_strippinnum 'TRUE'
```

Corresponding UI Option for Part Developer

None

Import_Csv_ApplyVectorConversion

The `Import_Csv_ApplyVectorConversion` directive determines if pins with square brackets in their basenames are to be treated as scalar pins when a part is created through CSV import and the `VectorSqrBracket` directive is set to 1.

Syntax

```
Import_Csv_ApplyVectorConversion 'TRUE'|'FALSE'
```

Example

```
Import_Csv_ApplyVectorConversion 'FALSE'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Delimeter

The `Import_Csv_Delimeter` directive specifies the delimiter that should be used in CSV import to parse property names and values from the CSV file.

Syntax

```
Import_Csv_Delimeter '<delimiter>'
```

Example

```
Import_Csv_Delimeter ';'
```

Corresponding UI Option for Part Developer

None

Import_CSV_GenerateSymbolForNoLocation

The `Import_CSV_GenerateSymbolForNoLocation` directive specifies whether symbols are to be generated in CSV import when pin location information is not available in the input file.

The following values are supported:

- TRUE

  Symbols are generated when location information is missing.

- FALSE

  Symbols are not generated when location information is missing.

Syntax

```
Import_CSV_GenerateSymbolForNoLocation 'TRUE'|'FALSE'
```

Example

```
Import_CSV_GenerateSymbolForNoLocation 'FALSE'
```

Corresponding UI Option for Part Developer

None

Import_Csv_LowAssertFlag

The `Import_Csv_LowAssertFlag` directive specifies the strings in pin names used to determine low assertion for parts created through CSV import.

Syntax

```
Import_Csv_LowAssertFlag '<string>'
```

Example

```
Import_Csv_LowAssertFlag 'L' 'Low'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_assertion

The `Import_Csv_Replace_assertion` directive specifies the name of the column in the CSV file whose value is used to determine whether a pin is low-asserted. Default values for the assertion column are L and Low. If the CSV file uses any other values, you need to configure the `Import_Csv_LowassertFlag` directive in your CPM file.

Syntax

```
Import_Csv_Replace_assertion '<column_name>'
```

Example

```
Import_Csv_Replace_assertion 'assertion'
```

Corresponding UI Option for Part Developer

None

See Also

Import_Csv_LowAssertFlag

Import_Csv_Replace_assertionchar

The `Import_Csv_Replace_assertionchar` directive specifies the name of the package property in the CSV file whose value should be imported as assertion characters. If the `assertion_char` property is defined, the values specified in the *Read/Write* and *Additional Read* fields specified in Setup are ignored.

Syntax

```
Import_Csv_Replace_assertionchar '<row_name>'
```

Example

```
Import_Csv_Replace_assertionchar 'assertion_char'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_DIFFPAIRPINSNEG

The `Import_Csv_Replace_DIFFPAIRPINSNEG` directive specifies the name of the column in the CSV file whose values should be imported to create negative differential pair pins.

Syntax

```
Import_Csv_Replace_DIFFPAIRPINSNEG '<column_name>'
```

Example

```
Import_Csv_Replace_DIFFPAIRPINSNEG 'DIFF_PAIR_PINS_NEG'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_DIFFPAIRPINSPOS

The `Import_Csv_Replace_DIFFPAIRPINSPOS` directive specifies the name of the column in the CSV file whose values should be imported to create positive differential pair pins.

Syntax

```
Import_Csv_Replace_DIFFPAIRPINSPOS '<column_name>'
```

Example

```
Import_Csv_Replace_DIFFPAIRPINSPOS 'DIFF_PAIR_PINS_POS'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_jedectype

The `Import_Csv_Replace_jedectype` directive specifies the name of the package property in the CSV file whose values should be assigned to the `JEDEC_TYPE` property.

Syntax

```
Import_Csv_Replace_jedectype '<row_name>'
```

Example

```
Import_Csv_Replace_jedectype 'jedec_type'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_loadsetupfile

The `Import_Csv_Replace_loadsetupfile` directive specifies the name of the package property in the CSV file whose value should be used in CSV import to identify the project file from which load values for the part are to be imported. You need to specify the absolute path to the project file.

Syntax

```
Import_Csv_Replace_loadsetupfile '<row_name>'
```

Example

```
Import_Csv_Replace_loadsetupfile 'load_setupfile'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_packagename

The `Import_Csv_Replace_packagename` directive specifies the name of the package-level property in the CSV file whose values should be imported as package names.

Syntax

```
Import_Csv_Replace_packagename '<row_name>'
```

Example

```
Import_Csv_Replace_packagename 'package_name'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pinlocation

The `Import_Csv_Replace_pinlocation` directive specifies the name of the column in the CSV file whose values should be imported as pin locations.

Syntax

```
Import_Csv_Replace_pinlocation '<column_name>'
```

Example

```
Import_Csv_Replace_pinlocation 'pin_location'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pinname

The `Import_Csv_Replace_pinname` directive specifies the name of the column in the CSV file whose values should be imported as pin names.

Syntax

```
Import_Csv_Replace_pinname '<column_name>'
```

Example

```
Import_Csv_Replace_pinname 'pin_name'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pinnumber

The `Import_Csv_Replace_pinnumber` directive specifies the name of the column in the CSV file whose values should be imported as pin numbers.

Syntax

```
Import_Csv_Replace_pinnumber '<column_name>'
```

Example

```
Import_Csv_Replace_pinnumber 'pin_number'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pinposition

The `Import_Csv_Replace_pinposition` directive specifies the name of the column in the CSV file whose values should be imported as pin positions.

Syntax

```
Import_Csv_Replace_pinposition '<column_name>'
```

Example

```
Import_Csv_Replace_pinposition 'pin_position'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pinshape

The `Import_Csv_Replace_pinshape` directive specifies the name of the column in the CSV file whose values should be imported as pin shapes.

Syntax

```
Import_Csv_Replace_pinshape '<column_name>'
```

Example

```
Import_Csv_Replace_pinshape 'pin_shape'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_pintype

The `Import_Csv_Replace_pintype` directive specifies the name of the column in the CSV file whose values should be imported as pin types.

Syntax

```
Import_Csv_Replace_pintype '<column_name>'
```

Example

```
Import_Csv_Replace_pintype 'pin_type'
```

Corresponding UI Option for Part Developer

None

Import_Csv_Replace_symbol

The `Import_Csv_Replace_symbol` directive specifies the name of the column in the CSV file whose values should be imported as symbols.

Syntax

```
Import_Csv_Replace_symbol '<column_name>'
```

Example

```
Import_Csv_Replace_symbol 'symbol'
```

Corresponding UI Option for Part Developer

None

IMPORT_DEHDL_PNS_CONSTRAINTS

When creating a System Capture project based on a 17.2 DE-HDL Design, to ensure that all the Physical and Spacing constraints are imported, set this directive to ON.

Syntax

```
IMPORT_DEHDL_PNS_CONSTRAINTS 'OFF'|'ON'
```

Example

```
IMPORT_DEHDL_PNS_CONSTRAINTS 'ON'
```

Note: In 17.4 designs, Physical and Spacing constraints are imported by default.

Import_DML_Braces_TreatedAs_Vector

The `Import_DML_Braces_TreatedAs_Vector` directive specifies the braces to be used for vector notation in DML import when the `Import_DML_Pins_DefaultVector` directive is set to `TRUE`.

Syntax

```
Import_DML_Braces_TreatedAs_Vector 'comma-separated_braces_list'
```

Example

```
Import_DML_Braces_TreatedAs_Vector '{,(,['
```

Corresponding UI Option for Part Developer

None

IMPORT_DEHDL_SKIP_ASCII

Set this directive to skip the .ascii files being copied from source libraries. By default value the directive is set to 'NO'.

Syntax

```
IMPORT_DEHDL_SKIP_ASCII 'YES'
```

Example

```
IMPORT_DEHDL_SKIP_ASCII 'YES'
```

Import_DML_DefaultPinType

The `Import_DML_DefaultPinType` directive specifies the default pin type to be assigned in DML import if pin type information is not available in the input file.

Syntax

```
Import_DML_DefaultPinType '<pin_type>'
```

Example

```
Import_DML_DefaultPinType 'BIDIR'
```

Corresponding UI Option for Part Developer

None

Import_DML_Pins_DefaultVector

The `Import_DML_Pins_DefaultVector` directive determines if pins with ( ), { }, or [] in pin names are to be treated as vector or scalar in DML import.

The following values are supported:

- TRUE

  Pins with ( ), { }, or [] in pin names are to be treated as vector pins.

- FALSE

  Pins with ( ), { }, or [] in pin names are to be treated as scalar pins.

Syntax

```
Import_DML_Pins_DefaultVector 'TRUE'|'FALSE'
```

Example

```
Import_DML_Pins_DefaultVector 'FALSE'
```

Corresponding UI Option for Part Developer

None

Import_DML_PwrGndNCPinsInGlobalSection

The `Import_DML_PwrGndNCPinsInGlobalSection` directive determines whether POWER, GROUND, and NC pins are to be imported as global pins in DML import.

The following values are supported:

- TRUE

  POWER, GROUND, and NC pins are imported as global pins; these pins are placed in the Global Pins grid.

- FALSE

  POWER, GROUND, and NC pins are imported as logical pins; these pins are placed in the Logical Pins grid.

Syntax

```
Import_DML_PwrGndNCPinsInGlobalSection 'TRUE'|'FALSE'
```

Example

```
Import_DML_PwrGndNCPinsInGlobalSection 'TRUE'
```

Corresponding UI Option for Part Developer

None

Import_FPGA_DefaultPinType

The `Import_FPGA_DefaultPinType` directive specifies the default pin type to be assigned in FPGA import when pin type information is not available in the input file. If required, you can change the assigned pin type on the Preview of Import Data page or after the part is created.

Syntax

```
Import_FPGA_DefaultPinType '<pin_type>'
```

Example

```
Import_FPGA_DefaultPinType 'BIDIR'
```

Corresponding UI Option for Part Developer

None

Import_FPGA_PwrGndNCPinsInGlobalSection

The `Import_FPGA_PwrGndNCPinsInGlobalSection` directive determines whether POWER, GROUND, and NC pins are to be imported as global pins or not in FPGA import.

The following values are supported:

- TRUE

  POWER, GROUND, and NC pins are imported as global pins; these pins are placed in the Global Pins grid.

- FALSE

  POWER, GROUND, and NC pins are imported as logical pins; these pins are placed in the Logical Pins grid.

Syntax

```
Import_FPGA_PwrGndNCPinsInGlobalSection 'TRUE'|'FALSE'
```

Example

```
Import_FPGA_PwrGndNCPinsInGlobalSection 'TRUE'
```

Corresponding UI Option for Part Developer

None

Import_FPGA_Standard_PartNameAsCellName

The `Import_FPGA_Standard_PartNameAsCellName` directive specifies if the cell created in standard FPGA import should have the same name as the primitive name in the source or in the destination.

The following values are supported:

- TRUE

  Cell name is same as the primitive name in the destination

- FALSE

  Cell name is same as the primitive name in the source

Syntax

```
Import_FPGA_Standard_PartNameAsCellName 'TRUE'|'FALSE'
```

Example

```
Import_FPGA_Standard_PartNameAsCellName 'FALSE'
```

Corresponding UI Option for Part Developer

None

Import_FPGA_Xilinx_CSVSeparationChar

The `Import_FPGA_Xilinx_CSVSeparationChar` directive specifies the default separator in the CSV file in Xilinx import.

Syntax

```
Import_FPGA_Xilinx_CSVSeparationChar '<separator>'
```

Example

```
Import_FPGA_Xilinx_CSVSeparationChar ','
```

Corresponding UI Option for Part Developer

None

Import_FPGA_Xilinx_SeparationChar

The `Import_FPGA_Xilinx_SeparationChar` directive specifies the default separator for the pad file in Xilinx import.

Syntax

```
Import_FPGA_Xilinx_SeparationChar '<separator>'
```

Example

```
Import_FPGA_Xilinx_SeparationChar '|'
```

Corresponding UI Option for Part Developer

None

Import_IBIS_With_Ibischk4

The `Import_IBIS_With_Ibischk4` directive determines whether `ibischk4` will be run during conversion.

Syntax

```
Import_IBIS_With_Ibischk4 'TRUE'|'FALSE'
```

Example

```
Import_IBIS_With_Ibischk4 'False'
```

Corresponding UI Option for Part Developer

None

Import_IBIS_With_Unchanged_ModelName

The `Import_IBIS_With_Unchanged_ModelName` directive determines the format in which the model name is written in the DML file.

The following values are supported:

- TRUE
- FALSE

  Model name is written as:

  <ibis_filename>_<model_name_in_ibisfile>

Syntax

```
Import_IBIS_With_Unchanged_ModelName 'TRUE'|'FALSE'
```

Example

```
Import_IBIS_With_Unchanged_ModelName 'TRUE'
```

Corresponding UI Option for Part Developer

None

IMPORT_SOURCE_AT_SITE_UNIT

When a new System Capture project is created that is based on an existing OrCAD Capture design, the new design has the same pin-to-pin spacing as the OrCAD Capture source project it is based on. To override the source pin-to-pin spacing with a site-level setting, add the following directive to the `site.cpm` in the `canvas` section:

```
IMPORT_SOURCE_AT_SITE_UNIT 'YES'
```

This ensures all OrCAD Capture designs are imported at a common grid value regardless of the original design unit.

Syntax

```
IMPORT_SOURCE_AT_SITE_UNIT 'YES'
```

Example

```
IMPORT_SOURCE_AT_SITE_UNIT 'YES'
```

Corresponding UI Option

None

Import_Text_Braces_TreatedAs_Vector

The `Import_Text_Braces_TreatedAs_Vector` directive specifies the braces to be used for vector notation when the `Import_Text_Pins_DefaultVector` directive is set to `TRUE`.

Syntax

```
Import_Text_Braces_TreatedAs_Vector 'comma-separated_braces_list'
```

Example

```
Import_Text_Braces_TreatedAs_Vector '{,(,['
```

Corresponding UI Option for Part Developer

None

See Also

Import_Text_Pins_DefaultVector

Import_Text_DefaultPinType

The `Import_Text_DefaultPinType` directive specifies the default pin type to be assigned in text import if pin type information is not available in the input file. You can modify the pin type on the Preview of Derived Data page or after the part is created.

Syntax

```
Import_Text_DefaultPinType '<pin_type>'
```

Example

```
Import_Text_DefaultPinType 'UNSPEC'
```

Corresponding UI Option for Part Developer

None

Import_Text_Pins_DefaultVector

The `Import_Text_Pins_DefaultVector` directive determines if pin names with ( ), { }, or [] are to be treated as vector or scalar.

The following values are supported:

- TRUE

  Pins with ( ), { }, or [] in pin names are to be treated as vector

- FALSE

  Pins with ( ), { }, or [] in pin names are to be treated as scalar

Syntax

```
Import_Text_Pins_DefaultVector 'TRUE'|'FALSE'
```

Example

```
Import_Text_Pins_DefaultVector 'FALSE'
```

Corresponding UI Option for Part Developer

None

Import_Text_PwrGndNCPinsInGlobalSection

The `Import_Text_PwrGndNCPinsInGlobalSection` directive determines whether POWER, GROUND, and NC pins are to be imported as global pins in text import.

Syntax

```
Import_Text_PwrGndNCPinsInGlobalSection 'TRUE'|'FALSE'
```

Example

```
Import_Text_PwrGndNCPinsInGlobalSection 'TRUE'
```

Corresponding UI Option for Part Developer

None

Import_ViewLogic_gridratio

The `Import_ViewLogic_gridratio` directive specifies the factor by which imported ViewLogic symbols are magnified. The default value, 5, ensures that imported ViewLogic symbols are displayed correctly in Design Entry HDL.

Caution

Changing the default value of the Import_ViewLogic_gridratio directive is likely to cause distortion in the imported symbols.

Syntax

```
Import_ViewLogic_gridratio '<magnification_factor>'
```

Example

```
Import_ViewLogic_gridratio '5'
```

Corresponding UI Option for Part Developer

None

IN_PORT_PIN_SIDE

This directive defines the default placement location of the input ports on the symbol outline.

Syntax

```
IN_PORT_PIN_SIDE 'LEFT'|'RIGHT'|'UP'|'DOWN'
```

LEFT	Input ports are automatically placed on the left side of the symbol outline. This is the default value.
RIGHT	Input ports are automatically placed on the right side of the symbol outline.
UP	Input ports are automatically placed on the top side of the symbol outline.
DOWN	Input ports are automatically placed on the bottom side of the symbol outline.

Example

```
IN_PORT_PIN_SIDE 'LEFT'
IN_PORT_PIN_SIDE 'UP'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - In

INCLUDE_PPT

The INCLUDE_PPT directive controls the loading of ptf files when directories are specified using the PPT or USE_LIBRARY_PPT directives.

Note: To package a cell-level ptf file, you will have to add the file name as a value of the INCLUDE_PPT directive.

The ptf files are identified by a .ptf extension. All ptf files located at the cell level must have this extension. Packager-XL, by default, loads all ptf files in a directory. The INCLUDE_PPT directive is used to modify this behavior to load only the ptf files listed.

If you specify the name lsttl, Packager-XL uses any file with this name, and any file named lsttl with a .ptf extension. You cannot use the INCLUDE_PPT directive in conjunction with the EXCLUDE_PPT directive. If both directives are specified, an error message is generated and the EXCLUDE_PPT directive is ignored.

- You must specify the INCLUDE_PPT directive in the Part Table section of the Project Setup form.
- You can use the INCLUDE_PPT directive for file names only.

Syntax

```
INCLUDE pptfile_name [,pptfile_name]...;
```

pptfile_name The name of a part table file. The file extension .ptf is optional.

The default value for the INCLUDE_PPT directive is none.

Example

The ptf files are identified by a .ptf extension.

If you have the following directives:

```
PPT /lib/site_ptfs;

INCLUDE_PPT site_1;
```

and if the contents of the /lib/site_ptfs files are as follows:

```
site_1.ptf
site_2.ptf
site_3.ptf
```

Packager-XL loads the `site_1.ptf` file.

INCREMENTAL_RUN

Identifies the unchanged subcircuits since the previous simulation run.

When you run stress analysis with this directive set to ‘ON’, instead of the entire design only the subcircuits that changed since the last analysis are simulated.

This directive is available only with the following licenses:

- `Allegro System Capture Designer`
- `Allegro PCB Venture`
- `Allegro System Capture Venture`
- `Allegro Enterprise System Design Authoring`
- `Allegro Enterprise Authoring Solution`

Syntax

```
INCREMENTAL_RUN 'ON'|'OFF'
```

Example

```
INCREMENTAL_RUN 'ON'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Electrical Stress Settings – System – General tab *– Incremental Analysis*

INFORMATIONAL_MESSAGES

Specifies where you want Informational messages to display.

Syntax

```
INFORMATIONAL_MESSAGES 'DIALOG'|'COMMANDPANE'|'SUPPRESS'
```

where

DIALOG	If you set the value to Dialog, Design Entry HDL displays the messages in a dialog box.
COMMANDPANE	If you set this directive to COMMANDPANE, Design Entry HDL displays the messages in the Console Command Window.
SUPPRESS	If you set the variable to SUPPRESS Design Entry HDL does not display the type of messages.

Example

```
INFORMATIONAL_MESSAGES 'DIALOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Messages — Informational

See Also

- FATAL_MESSAGES
- ERROR_MESSAGES
- WARNING_MESSAGES

INOUT_PORT_PIN_SIDE

This directive defines the default placement location of the bidirectional (inout) ports on the symbol outline.

Syntax

```
INOUT_PORT_PIN_SIDE 'LEFT'|'RIGHT'|'UP'|'DOWN'
```

LEFT	Bidirectional ports are automatically placed on the left side of the symbol outline.
RIGHT	Bidirectional ports are automatically placed on the right side of the symbol outline. This is the default value.
UP	Bidirectional ports are automatically placed on the top side of the symbol outline.
DOWN	Bidirectional ports are automatically placed on the bottom side of the symbol outline.

Example

```
INOUT_PORT_PIN_SIDE 'LEFT'
INOUT_PORT_PIN_SIDE 'UP'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - In/Out

INPORT

The INPORT directive is used in Allegro System Capture and Schgen. This directive is used to add input port to the *Port* section of the *Special Symbols* bucket.

Syntax

```
INPORT '<library:cell:view>'
```

where

library	is the library name in which you want the symbol to be added.
cell	is the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	is the view name in which you want the symbol to be added.

Example

```
INPORT 'standard.inport:sym_1'
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for System Connectivity Manager

None

See Also

IOPORT

OUTPORT

INPUT_SCRIPT

Specify the path to the file that contains Design Entry HDL console commands to be run when you start Design Entry HDL.

Syntax

```
INPUT_SCRIPT '<path>'
```

where

path	is the path to the directory containing the input script.

Example

```
INPUT_SCRIPT ''
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Paths page — Input Paths — Input Script

See Also

- CATPATH
- ATTRIBUTES_DIR
- PPT_OPTIONSET_PATH

INST_BLOCKAGE_MARGIN

Defines the number of grid units to be added to instance blockage margin.

Syntax

```
INST_BLOCKAGE_MARGIN '<value>'
```

where

`value`	Number of grid units. The default value is `0`.

Example

```
INST_BLOCKAGE_MARGIN '0'
```

INST_DEFAULT_ALIGNMENT

Use this directive to specify the default alignment of the properties visible on the schematic page.

Syntax

```
inst_default_alignment <value>.
```

The valid values are: Left, Right, and Center.

Example

```
inst_default_alignment 'Left'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Properties — Default Alignment

See Also

INST_DEFAULT_PROP_SIZE

INST_DEFAULT_VISIBILITY

INST_DEFAULT_PROP_SIZE

Use this directive to specify the default size of the text used for displaying property name and values on the schematic page.

Syntax

```
inst_default_prop_size <size>
```

Example

```
inst_default_prop_size '41'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Properties — Size

See Also

INST_DEFAULT_ALIGNMENT

INST_DEFAULT_VISIBILITY

INST_DEFAULT_VISIBILITY

Use this directive to specify the visibility settings for displaying property name and values on the schematic page.

Syntax

```
inst_default_visibility <Value>
```

The valid options are: None, Name, Value, and Both.

Example

```
inst_default_visibility 'Value'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Properties — Default Visibility.

See Also

INST_DEFAULT_ALIGNMENT

INST_DEFAULT_PROP_SIZE

Instantiation_Packaging_Validation_Type

The `Instantiation_Packaging_Validation_Type` directive determines the type of instantiation and packaging checks to be run with the PCB Librarian XL license.

The following values are supported:

- 1

  Runs the `con2con` utility

- 0
- Runs the `hlibftb` utility

Syntax

```
Instantiation_Packaging_Validation_Type '1'|'0'
```

Example

```
Instantiation_Packaging_Validation_Type '1'
```

Corresponding UI Option for Part Developer

None

IOPORT

The IOPORT directive is used in Allegro System Capture and Schgen. This directive is used to add InOut port to the *Port* section of the *Special Symbols* bucket.

Syntax

```
IOPORT '<library:cell:view>'
```

where

library	is the library name in which you want the symbol to be added.
cell	is the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	is the view name in which you want the symbol to be added.

Example

```
IOPORT 'standard.ioport:sym_1'
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for System Connectivity Manager

None

See Also

INPORT

OUTPORT

Import_IBIS_With_Dmlcheck

The `Import_IBIS_With_Dmlcheck` directive determines whether DML checks will be run on the converted DML file.

Syntax

```
Import_IBIS_With_Dmlcheck 'TRUE'|'FALSE'
```

Example

```
Import_IBIS_With_Dmlcheck 'False'
```

Corresponding UI Option for Part Developer

None

ITEM_OPACITY

Controls the opacity or transparency of a picture on the canvas.

Syntax

```
ITEM_OPACITY '<opacity_factor>'
```

opacity_factor	Any number between 0 and 99. A lower value results in a more opaque picture. An opacity factor of 1 indicates that the picture is completely opaque, while a value of 99 indicates that the background of the picture is transparent.

Examples

```
ITEM_OPACITY '1'
ITEM_OPACITY '99'
```

12

L Directives

This chapter lists the CPM directives that start with `L` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

LAST_BOARD_FILE

This directive corresponds to the *Output Board File* field in the Export Physical dialog. If you want to specify an output board file to update the PCB Editor board with changes in the schematic when exporting a design, specify a value for this directive.

The `last_board_file` directive is written to the `START_DESIGNSYNC` section with the value '<*board file name*>' if you select the *Update PCB Editor Board (Netrev)* check box in the Export Physical dialog and specify an output board file.

Design Synchronization uses this value to remember the last setting with which the dialog was last run. The next time the design is packaged and if you select the Update PCB Editor Board (Netrev) check box, Design Synchronization by default displays the same output board file.

Syntax

```
last_board_file '<board file name>'
```

Example

```
last_board_file 'b164205.brd'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Output Board File

LAST_OUTPUT_FILE

Stores the path of the output file that contains the BOM report. By default, the value is picked from this directive.

Value

```
LAST_OUTPUT_FILE '<complete path to the .rpt file>'
```

Example

```
LAST_OUTPUT_FILE './worklib/ps0/bom/BOM.rpt'
```

Corresponding UI Option for Project Manager

Tools — Packager Utilities — Bill of Materials — BOM-HDL dialog box — Output Options — Output File

LAST_TEMPLATE_FILE

Stores the path to the template file.

Value

```
LAST_TEMPLATE_FILE '<complete path to the .bom file>'
```

Example

```
LAST_TEMPLATE_FILE './worklib/ps0/bom/BOM.rpt'
```

Corresponding UI Option for Project Manager

Tools — Packager Utilities — Bill of Materials — BOM-HDL dialog box — Output Options — Template File

LAST_VARIANT_FILE

The LAST_VARIANT_FILE directive is used in Allegro Design Entry HDL - BOM-HDL and Allegro Design Entry HDL - Design Variance. This directive stores the name of the last variant file that was used to create the Variant BOM report. It also stores the name of the last variant file opened in Variant Editor.

Variant Editor uses the value of this directive to remember the name of the last variant file that was used to create the Variant BOM report. The next time you open the BOM-HDL dialog, the same variant file name is displayed in the *Variant File* field. Variant Editor also uses the value of this directive to remember the name of the last variant file that was opened.

Syntax

```
LAST_VARIANT_FILE '<variant file name>.dat'
```

When Operated Under Variant BOM Report

Example

```
START_BOMHDL
last_variant_file 'variant_orig.dat'
END_BOMHDL
```

Corresponding UI Option for Design Entry HDL

Tools — Packager Utilities — Bill of Materials — BOM-HDL dialog box — Variant BOM — Variant File

When Operated Under Variant Editor

Example

```
START_VARIANT

LAST_VARIANT_FILE 'variant_orig.dat'

END_VARIANT
```

Corresponding UI Option

None

LastFlow

Specifies the type of flow being used by the project.

Syntax

```
LastFlow '<type of flow>'
```

The flow files are RDF (Resource Description Framework) files and are available at the `$adw_conf_root/<company_name>/<site_name>/cdssetup/projmgr/flows` location. Flow files include:

- Block Library Flow
- Board Design flow
- Design reference flow
- High Speed Board Design Flow
- Reference Board Design Flow

Example

```
LastFlow 'High Speed Board Design Flow'
```

Corresponding UI Option

Admin – Open Flow File – `<flow.rdf file>`

LAUNCH_OPTION

Use this option to specify the tool using which Export Physical will open the board file after packaging it. You can set Export Physical to open the board file in Allegro PCB Editor, Allegro PCB SI, or Allegro Package Designer, or not to open the board.

Syntax

```
launch_option <option number>
```

Valid values are: 0: Allegro PCB Editor; 1: Allegro PCB SI; 2: Allegro Package Designer; 3: SiP Digital Layout; 4: None

Example

```
launch_option '3'
```

Corresponding UI Option for System Connectivity Manager

Project — Export — PCB Board— Layout Launching Options

See Also

- ETCH_REMOVAL
- GEN_PSTFILES
- IGNORE_FIXED
- OVERWRITE_CONSTRAINTS

LEFT_IN_OFFPAGE

Use this directive to specifies the lib:cell:view of the symbol to be used as offpage connector for the input signal on the left of the schematic page.

Note: This option works only when the use_offpage is set to 1.

Syntax

```
left_in_offpage <library.cell:view>
```

Example

```
left_in_offpage 'standard.offpage:sym_1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Input

See Also

USE_OFFPAGE

LEFT_IN_ROT

LEFT_IO_OFFPAGE

LEFT_IO_ROT

LEFT_OUT_OFFPAGE

LEFT_OUT_ROT

LEFT_IN_ROT

Use this directive to specify the angle in degrees by which the offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by right_in_offpage as input offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
left_in_rot <angle>
```

Example

```
left_in_rot '45'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Left Side — Input — Rotate By

See Also

USE_OFFPAGE

LEFT_IN_OFFPAGE

LEFT_IO_OFFPAGE

LEFT_IO_ROT

LEFT_OUT_OFFPAGE

LEFT_OUT_ROT

LEFT_IO_OFFPAGE

Use this directive to specifies the lib:cell:view of the symbol to be used as offpage connector for the input/output signal on the left of the schematic page.

Note: This option works only when the use_offpage is set to 1.

Syntax

```
left_io_offpage <library.cell:view>
```

Example

```
left_io_offpage 'standard.offpage:sym_6'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Left Side — InOut

See Also

USE_OFFPAGE

LEFT_IN_OFFPAGE

LEFT_IN_ROT

LEFT_IO_ROT

LEFT_OUT_OFFPAGE

LEFT_OUT_ROT

LEFT_IO_ROT

Use this directive to specify the angle in degrees by which the offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by as input offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
left_io_rot <angle>
```

Example

left_io_rot '0'

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Input — Rotate By

See Also

USE_OFFPAGE

LEFT_IN_OFFPAGE

LEFT_IN_ROT

LEFT_IO_OFFPAGE

LEFT_OUT_OFFPAGE

LEFT_OUT_ROT

LEFT_OUT_OFFPAGE

Use this directive to specifies the lib:cell:view of the symbol to be used as offpage connector for the output signal on the left of the schematic page.

Note: This option works only when the use_offpage is set to 1.

Syntax

```
left_out_offpage <library.cell:view>
```

Example

```
left_out_offpage 'standard.offpage:sym_5'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Output.

See Also

USE_OFFPAGE

LEFT_IN_OFFPAGE

LEFT_IN_ROT

LEFT_IO_OFFPAGE

LEFT_IO_ROT

LEFT_OUT_ROT

LEFT_OUT_ROT

Use this directive to specify the angle in degrees by which the offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by right_out_offpage as input offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
left_out_rot <angle>
```

Example

```
left_out_rot '180'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Output — Rotate By.

See Also

USE_OFFPAGE

LEFT_IN_OFFPAGE

LEFT_IN_ROT

LEFT_IO_OFFPAGE

LEFT_IO_ROT

LEFT_OUT_OFFPAGE

LIBRARY_BROWSER

Activates the Search Stack dialog box when you enter the lib command in the console window and then press Return. If off, the current search stack is displayed.

Syntax

```
LIBRARY_BROWSER 'ON'|'OFF'
```

Example

```
LIBRARY_BROWSER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Libraries Browser

See Also

- DRAWING_BROWSER

LINE_CAP_STYLE

A line cap style defines the ending of a line. This directive specifies the default cap style used when drawing a line on the canvas

Syntax

```
LINE_CAP_STYLE '<line_cap_style>'
```

OBJECT	Represents the object type for which the directive is set.
line_cap_style	A line cap style can be round-cap, square-cap, diamond-cap, or arrowhead-cap.

Example

```
LINE_CAP_STYLE 'square-cap'
```

LINE_COLOR

Sets the color of a line drawn on the canvas.

Syntax

```
LINE_COLOR '<line_color>'
```

line_color	The value is specified in hex color code. For example, `#000000`, `#FF0000`, and `#0022CC` represent black, red, and blue, respectively.

Example

```
LINE_COLOR '#CC0000'
LINE_COLOR '#FF0000'
LINE_COLOR '#0022CC'
```

LINE_JOIN_STYLE

Sets the type of corner created when two lines join.

Syntax

```
LINE_JOIN_STYLE '<line_join_style>'
```

line_join_style	A line join style can be `miter-join`, `round-join`, or `bevel-join`.

Example

```
LINE_JOIN_STYLE 'round-join'
LINE_JOIN_STYLE 'miter-join'
LINE_JOIN_STYLE 'bevel-join'
```

LINE_STYLE

Specifies the default style used when drawing a line on the canvas.

Syntax

```
LINE_STYLE '<line_style>'
```

line_style	The line style can be: ■ solid ■ dash ■ dot ■ dash-dot ■ dash-dot-dot The GUI equivalent of line style are: 

Example

```
LINE_STYLE 'SOLID'
```

LINE_WIDTH

Specifies the default width of the line object.

Syntax

```
LINE_WIDTH '<width>'
```

Example

```
LINE_WIDTH '1'
```

LIST_VALID_VERSIONS_METADATA

When this directive is ON, DE-HDL only displays valid versions to which a component version can be changed. Valid versions are based on the PACK_TYPE that was applied on the component. This directive is set to 'ON' by default.

Syntax

```
LIST_VALID_VERSIONS_METADATA 'ON'|'OFF'
```

Example

```
LIST_VALID_VERSIONS_METADATA 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

LOCK_FILE_PERM

DE-HDL creates `.lck` files during user operations and removes these files once the operations are complete. By default, DE HDL creates these lock files with 444 permissions. This marks the file as read only for all users.

You can modify the permission settings with which DE HDL creates lock files using the LOCK_FILE_PERM directive. The directive is applicable for all types of DE HDL lock files—page files, connectivity files (XCON), constraints and properties file (DCF).

Syntax

```
LOCK_FILE_PERM '444'|'666'|'777'
```

where

444	Sets read-only permissions for all users (owner, group, and others). Sets the permissions as 444 (--r-- r-- r--).
666	Sets read and write permissions for all users—owner, group, and others. Sets the permissions as 666 (--rw-- rw-- rw--)
777	Allows you to control the permissions. '777' gives full access to all users—owner, group, and others. Sets the permissions as 777 (rwx rwx rwx)

Example

```
LOCK_FILE_PERM '444'
```

Corresponding UI Option for Allegro Design Entry HDL

None

LOGIC_DOT_RADIUS

Adjusts the diameter of dots at wire connections in schematic drawings and published PDF.

Syntax

```
LOGIC_DOT_RADIUS '<value>'
```

The valid values range from 1 to 40. For any value greater than 40, DE-HDL retains the last valid value set by the user. In case of PDF Publisher, any value from 1 to 5 is rendered to 6 in the published PDF.

Example

```
LOGIC_DOT_RADIUS '13'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Dots — Logic Dot Radius

See Also

SYMBOL_DOT_RADIUS

LOGIC_GRID_MULTIPLE

Displays every nth grid line to define where objects can be placed so that pins do not fall off-grid. This ensures the correct connectivity of wires and symbols.

Syntax

```
LOGIC_GRID_MULTIPLE '<value>'
```

where

value	any number greater than 0.002.

Example

```
LOGIC_GRID_MULTIPLE '5'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Logic Grid — Multiple

See Also

- DOC_GRID_MULTIPLE
- SYMBOL_GRID_MULTIPLE

LOGIC_GRID_SIZE

Adjusts the logic grid size to be smaller or larger.

Syntax

```
LOGIC_GRID_SIZE '<value>'
```

where

value	any number greater than 0.002.

Example

```
LOGIC_GRID_SIZE '0.100'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Logic Grid— Size

See Also

- LOGIC_GRID_MULTIPLE
- LOGIC_GRID_TOGGLE
- LOGIC_GRID_TYPE

LOGIC_GRID_TOGGLE

Displays or hides the grid.

Syntax

```
LOGIC_GRID_TOGGLE 'ON'|'OFF'
```

Example

```
LOGIC_GRID_TOGGLE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Logic Grid— TOGGLE

See Also

- SYMBOL_GRID_TOGGLE
- DOC_GRID_TOGGLE
- LOGIC_GRID_TYPE
- LOGIC_GRID_MULTIPLE
- LOGIC_GRID_SIZE

LOGIC_GRID_TYPE

Displays the grid as Dots or dashed Lines.

Syntax

```
LOGIC_GRID_TYPE 'DOT'|'LINE'
```

Example

```
LOGIC_GRID_TYPE 'LINE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Logic Grid — Type

See Also

- SYMBOL_GRID_TOGGLE
- DOC_GRID_TOGGLE
- LOGIC_GRID_TOGGLE
- LOGIC_GRID_MULTIPLE
- LOGIC_GRID_SIZE

lrm_logfile

Specifies the name of the log file that is generated during the update operation. This file is created at the location defined by the `adwconfigdir` directive in the ADW section.

Syntax

```
lrm_logfile '<name>'
```

Example

```
lrm_logfile 'lrm.log'
```

Corresponding UI Option

None

LOWVOLTAGE_CLASS_PATTERN

Defines the patterns to identify low voltage net classes.

The nets of low voltage class with voltage greater than the threshold voltage are identified when the `Low-voltage class net having incorrect voltage` audit rule is run.

Syntax

```
LOWVOLTAGE_CLASS_PATTERN '<pattern>'
```

Example

```
LOWVOLTAGE_CLASS_PATTERN 'LV' 'LOW
```

Corresponding UI Option in Allegro System Capture

None

LOWVOLTAGE_THRESHOLD

Defines the maximum voltage for nets in low voltage net classes. There are no lower or upper limits.

The nets of voltage class with voltage greater than the threshold voltage are identified based on the patterns defined in this directive when the `Low-voltage class net having incorrect voltage` rule is run.

Syntax

```
LOWVOLTAGE_THRESHOLD '<Max Voltage Value>'
```

Example

```
LOWVOLTAGE_THRESHOLD '48V'
```

Corresponding UI Option in Allegro System Capture

None

13

M Directives

This chapter lists the CPM directives that start with `M` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

MAKE_PAGE_TITLE_INVISIBLE

By default, when a design is cross referenced using Crefer, the cross-referenced page title property, `CDS_XR_PAGE_TITLE`, is displayed above the page border.

The title can be made invisible by setting this directive to 'ON' in the `START_CREFERHDL` section of the `.cpm` file. Once set, the `CDS_XR_PAGE_TITLE` property will be invisible on all subsequent cross referencing runs.

Syntax

```
make_page_title_invisible 'ON'|'OFF'
```

Example

```
make_page_title_invisible 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Content page — Write Options — Make Page Title Invisible

MAX_COL_IMPORT_TABLE

Sets the maximum number of columns which can be imported from the csv file to display in the table (object) in a design.

Syntax

```
MAX_COL_IMPORT_TABLE 'max_num_of_columns_imported'
```

Example

```
MAX_COL_IMPORT_TABLE 20
```

Corresponding UI Option

None

See Also

- MAX_ROW_IMPORT_TABLE
- EDITABLE_IMPORT_TABLE

MAX_DRAWINGS

Specifies the maximum number of viewports that you can open in a session of Design Entry HDL.

Syntax

```
MAX_DRAWINGS '<value>'
```

where

value	any number greater than 0.

Example

```
MAX_DRAWINGS '50'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Drawings — Maximum Drawings

MAX_ERRORS

The MAX_ERRORS directive specifies the maximum number of errors allowed before Packager-XL terminates operation.

Syntax

```
MAX_ERRORS number;
```

number	The number of errors allowed before Packager-XL terminates operation.

By default, Packager-XL terminates operation after 999 errors.

Example

```
MAX_ERRORS 500;
NET_NAME_CHARS
```

MAX_PINS_IN_NET

Defines the maximum number of pins in a net before it is considered a VOLTAGE net.

Syntax

```
MAX_PINS_IN_NET '<value>'
```

where

value any number greater than 0.

Note: If you want the new value of this directive to be automatically updated in the Constraint Manager database every time you launch DE-HDL, use `REFRESH_` as a prefix with the directive.

Example

```
MAX_PINS_IN_NET '100'
```

```
REFRESH_MAX_PINS_IN_NET '100'
```

Corresponding UI Option for Allegro Design Entry HDL

None

MAX_ROW_IMPORT_TABLE

Sets the maximum number of rows which can be imported from the csv file to display in the *Table* object. This directive along with MAX_COL_IMPORT_TABLE ensures that the table fits into the page on which it is displayed.

Syntax

```
MAX_ROW_IMPORT_TABLE 'max_num_of_rows_imported'
```

Example

```
MAX_ROW_IMPORT_TABLE 20
```

Corresponding UI Option

None

See Also

- MAX_COL_IMPORT_TABLE
- EDITABLE_IMPORT_TABLE

Max_Search_Rows

Defines the maximum number of search results that can appear on a single page.

Using the Search Count slider in the user interface, you can set up to a maximum of 2500. To specify a number greater than 2500, use this directive. Max_Search_Rows, a `COMP_BROWSER` directive, allows you to specify up to a maximum of 32767 rows.

Note: Specifying a number greater than 500 can impact search performance.

Important

This directive is applicable only to the database mode.

Syntax

```
Max_Search_Rows 'value'
```

Example

```
Max_Search_Rows '500'
```

Corresponding UI Option

Configuration – Setup – Search – Search Count

Minimize_On_Add

Defines whether the Part Information Manager window should automatically minimize when you choose a part to add on the schematic.

Syntax

```
Minimize_On_Add 'True'| 'False'
```

Example

```
Minimize_On_Add 'True'
```

Corresponding UI Option

Configuration – Setup – Details - Minimize on Add

MISO_NET_PATTERN

Defines the patterns used for identifying nets as Master Input Slave Output (MISO) nets.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `MISO and MOSI pins of ICs not connected to the same set of ICs or connectors`
- `MISO and MOSI pins of ICs incorrectly connected to non-MISO or non-MOSI pins`
- `Unconnected Chip Select pins of ICs when MISO or MOSI pins are connected`
- `Unconnected Clock pins of ICs when MISO or MOSI pins are connected`
- `Unconnected MISO and MOSI pins of ICs`
- `Clock pins of ICs not connected to same set of ICs or connectors as MISO and MOSI pins`

These pin rules are available with the following licenses:

- `Allegro System Capture Designer`
- `Allegro PCB Venture`
- `Allegro System Capture Venture`
- `Allegro Enterprise System Design Authoring`
- `Allegro Enterprise Authoring Solution`

To detect a pin as MISO, the following two conditions must be met:

- The net name must match the MISO net pattern
- The pin name must match the MISO pin pattern

Syntax

```
MISO_PIN_PATTERN '<pin pattern>'
```

Example

```
MISO_NET_PATTERN 'MISO' 'SPI'
```

Corresponding UI Option in Allegro System Capture

None

See Also

- MISO_PIN_PATTERN
- MOSI_NET_PATTERN
- MOSI_PIN_PATTERN

MISO_PIN_PATTERN

Defines the patterns used for identifying pins as Master Input Slave Output (MISO) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `MISO and MOSI pins of ICs not connected to the same set of ICs or connectors`
- `MISO and MOSI pins of ICs incorrectly connected to non-MISO or non-MOSI pins`
- `Unconnected Chip Select pins of ICs when MISO or MOSI pins are connected`
- `Unconnected Clock pins of ICs when MISO or MOSI pins are connected`
- `Unconnected MISO and MOSI pins of ICs`
- `Clock pins of ICs not connected to same set of ICs or connectors as MISO and MOSI pins`

These pin rules are available with the following licenses:

- `Allegro System Capture Designer`
- `Allegro PCB Venture`
- `Allegro System Capture Venture`
- `Allegro Enterprise System Design Authoring`
- `Allegro Enterprise Authoring Solution`

Syntax

```
MISO_PIN_PATTERN '<pin pattern>'
```

The following default MISO pin patterns are supported:

```
MISO, SOMI, DI, DIN, SDI, DO, SDO, DOUT
```

Example

```
MISO_PIN_PATTERN 'MISO' 'SOMI' 'DI'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *MISO Pin Patterns*

See Also

- MISO_NET_PATTERN
- MOSI_NET_PATTERN
- MOSI_PIN_PATTERN

MONOCHROME_PRINTING_THRESHOLD

Enhances the printing of black and white images by customizing the threshold values. If you want an image to be darker then set higher value of threshold and for a brighter image, set the threshold value lower than the pixels value. Is an alternative for the `sch::setMonochromePrintingThreshold` Tcl command.

Syntax

```
MONOCHROME_PRINTING_THRESHOLD = 'value'
```

where threshold value can be 0-255. Default value is 127.

Example

```
MONOCHROME_PRINTING_THRESHOLD = 180
```

Corresponding UI Option

None

See Also

IMAGE_COLOR_MODE_PRINT

MOSI_NET_PATTERN

Defines the patterns used for identifying nets as Master Output Slave Input (MOSI) nets.

This directive is used when the following Connectivity Checks audit rules are run:

- `MISO and MOSI pins of ICs not connected to the same set of ICs or connectors`
- `MISO and MOSI pins of ICs incorrectly connected to non-MISO or non-MOSI pins`
- `Unconnected Chip Select pins of ICs when MISO or MOSI pins are connected`
- `Unconnected Clock pins of ICs when MISO or MOSI pins are connected`
- `Unconnected MISO and MOSI pins of ICs`
- `Clock pins of ICs not connected to same set of ICs or connectors as MISO and MOSI pins`

These pin rules are available with the following licenses:

- `Allegro System Capture Designer`
- `Allegro PCB Venture`
- `Allegro System Capture Venture`
- `Allegro Enterprise System Design Authoring`
- `Allegro Enterprise Authoring Solution`

To detect a pin as MOSI, the following two conditions must be met:

- The net name must match the MOSI net pattern
- The pin name must match the MOSI pin pattern

Syntax

```
MOSI_PIN_PATTERN '<pin pattern>'
```

Example

```
MOSI_NET_PATTERN 'MOSI' 'SPI'
```

Corresponding UI Option in Allegro System Capture

None

See Also

- MISO_NET_PATTERN
- MISO_PIN_PATTERN
- MOSI_PIN_PATTERN

MOSI_PIN_PATTERN

Defines the patterns used for identifying pins as Master Output Slave Input (MOSI) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `MISO and MOSI pins of ICs not connected to the same set of ICs or connectors`
- `MISO and MOSI pins of ICs incorrectly connected to non-MISO or non-MOSI pins`
- `Unconnected Chip Select pins of ICs when MISO or MOSI pins are connected`
- `Unconnected Clock pins of ICs when MISO or MOSI pins are connected`
- `Unconnected MISO and MOSI pins of ICs`
- `Clock pins of ICs not connected to same set of ICs or connectors as MISO and MOSI pins`

These pin rules are available with the following licenses:

- `Allegro System Capture Designer`
- `Allegro PCB Venture`
- `Allegro System Capture Venture`
- `Allegro Enterprise System Design Authoring`
- `Allegro Enterprise Authoring Solution`

Syntax

```
MOSI_PIN_PATTERN '<pin pattern>'
```

The following default MOSI pin patterns are supported:

```
SIMO, MOSI, DO, SDO, DOUT, DI, DIN, SDI
```

Example

```
MOSI_PIN_PATTERN 'SIMO' 'MOSI'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *MOSI Pin Patterns*

See Also

- MOSI_NET_PATTERN
- MISO_NET_PATTERN
- MISO_PIN_PATTERN

MOVE

Draws wires that you move as Orthogonal or Direct

Syntax

```
MOVE 'ORTHOG'|'DIRECT'
```

Example

```
MOVE 'ORTHOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Wire — Add radio button

MULTI_FORMAT

Allows multiple-format signal names in the design.

Syntax

```
MULTI_FORMAT 'ON'|
```

where

ON	'( )', '<>', and '[]' are considered special characters that designate a vectored signal. They cannot be used anywhere else in the signal name. The parentheses must be matched correctly and must contain either an integer or a parameter. Colon (:), comma (,) and ampersand (&) are considered special characters that represent concatenation. They cannot be used anywhere else in the signal name. **Note:** The default value of the MULTI_FORMAT directive is set to 'ON'. The directive can now be modified in the project and site cpm files.

Example

```
MULTI_FORMAT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

MULTIPAGEBODERCREF

Use this directive to generate cross-reference links for designs that have more than one type of page border used in the design.

Syntax

```
MULTIPAGEBODERCREF '1|0'
```

Example

```
MULTIPAGEBODERCREF '0'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — Advanced — Cref Links— Generate cref links for a design with multiple page borders

14

N Directives

This chapter lists the CPM directives that start with `N` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

NAVIGATION_OPTION

Sets the default view of the Schematic Cells pane in Component Revision Manager.

Syntax

```
NAVIGATION_OPTION 'PAGE'|'INSTANCE'|'BLOCK'
```

where

PAGE	Shows the occurrences of schematic cells (in the Schematic Cells pane) on a per-page basis. When you specify this directive, only the page number (where the cell is located) of the design appears under the Page(s) Impacted header column.
INSTANCE	Shows the occurrences of schematic cells (in the Schematic Cells pane) on a per-instance basis.When you specify this directive, the page number and the instance name of the cell appears under the Instance(s) Impacted header column.
BLOCK	Shows the occurrences of schematic cells (in the Schematic Cells pane) on a per-block basis. For example, you schematic may contain blocks such as top, bottom, or low. When you specify this directive, the block name (to which the schematic cell belongs) appears under the Block(s) Impacted header column.

Example

```
NAVIGATION_OPTION 'PAGE'
```

Corresponding UI Option for Allegro Design Entry HDL

None

NAVLINKS_TEXT_FONT_BOLD

Specifies whether the text for navigation links appear in **bold** face.

Syntax

```
NAVLINKS_TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
NAVLINKS_TEXT_FONT_BOLD 'TRUE'
```

NAVLINKS_TEXT_FONT_COLOR

Sets the color of the text for navigation links.

Syntax

```
NAVLINKS_TEXT_FONT_COLOR '<font_color>'
```

Where

font_color	The default color of the text for navigation links. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#0000FF'` (Blue).

Examples

```
NAVLINKS_TEXT_FONT_COLOR '#0000FF'
```

NAVLINKS_TEXT_FONT_ITALIC

Specifies whether the text for navigation links appear in *italics*.

Syntax

```
NAVLINKS_TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
NAVLINKS_TEXT_FONT_ITALIC 'TRUE'
```

NAVLINKS_TEXT_FONT_NAME

Sets the font face used to display the text for navigation links.

Syntax

```
NAVLINKS_TEXT_FONT_NAME '<font_name>'
```

Where

font_name	The default font face used to display the text for navigation links. The value can be any of the system-supported fonts. The default value is `'ARIAL'`.

Examples

```
NAVLINKS_TEXT_FONT_NAME 'ARIAL'
```

NAVLINKS_TEXT_FONT_SIZE

Sets the size of the text for navigation links.

Syntax

```
NAVLINKS_TEXT_FONT_SIZE '<font_size>'
```

Where

`font_size`	The default font size used to display the text for navigation links. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
NAVLINKS_TEXT_FONT_SIZE '10'
```

NAVLINKS_TEXT_FONT_UNDERLINE

Specifies whether the text for navigation links appear underlined.

Syntax

```
NAVLINKS_TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
NAVLINKS_TEXT_FONT_UNDERLINE 'TRUE'
```

NAVLINKS_TEXT_MARGIN

Sets the margin or the space around navigation links.

Syntax

```
NAVLINKS_TEXT_MARGIN '<value>'
```

Where

`value`	Any positive integer value. The default value is `1`.

Examples

```
NAVLINKS_TEXT_MARGIN '1'
NAVLINKS_TEXT_MARGIN '0.5'
```

NET_NAME_CHARS

The NET_NAME_CHARS Directive specifies special (non-alphanumeric) characters permitted in physical net names in Allegro System Capture and Packager-XL.

The NET_NAME_CHARS Directive does not has impact on physical net names in the state file. It impacts only new net names. If you want to use nets that are already assigned, use the REPACKAGE directive.

The NET_NAME_CHARS directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
NET_NAME _CHARS character[,character]...;
```

where

character	are the special character(s) allowed in physical net names. More than one character can be specified by separating each character with a comma (to include a comma as a special character, enclose the comma in single quotes).

The default is the PCB Editor *legal* character set which, is listed below:

#, %, &, ( ), +, -, _, /, =, >, ., :, ?, [ ], ^, ‘, |, and 0-9

Example

```
NET_NAME_CHARS _, +;
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

NET_NAME_LENGTH

The NET_NAME_LENGTH controls the maximum length of physical net names generated by the packager in Allegro System Capture and Packager-XL. If a physical net name is already in the state file and its length is longer than the value specified, it generates an error message.

The NET_NAME_LENGTH directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
NET_NAME_LENGTH number;
```

where

number	is the maximum number of characters for a physical net name.

The default value for the NET_NAME_LENGTH is 31.

Note: If you change the default value of this directive, it would take effect only when you repackage the design keeping the *Regenerate Physical Net Names* option on.

Example

```
NET_NAME_LENGTH 18;
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Physical Net Name Max Length

Corresponding UI Option for Design Entry HDL

None

NETGROUP_FILL_COLOR

Sets the default fill color for netgroups.

Syntax

```
NETGROUP_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color for netgroups. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FFFFFF'` (White).

Examples

```
NETGROUP_FILL_COLOR '#E8EFF7'
```

NETGROUP_FILL_STYLE

Sets the style that is used to fill netgroups.

Syntax

```
NETGROUP_FILL_STYLE '<fill_style>'
```

Where

`fill_style`	The default fill style for netgroups. Valid values: `none`, `solid` (default).

Examples

```
NETGROUP_FILL_STYLE 'none'
NETGROUP_FILL_STYLE 'solid'
```

NETGROUP_LINE_CAP_STYLE

A line cap style defines the ending of a line. This directive defines the line cap style for netgroups.

Syntax

```
NETGROUP_LINE_CAP_STYLE '<cap_style>'
```

Where

`cap_style`	The default line cap style for netgroups. Valid values: `round-cap` (default), `square-cap`, `diamond-cap`, `arrowhead-cap`.

Examples

```
NETGROUP_LINE_CAP_STYLE 'round-cap'
NETGROUP_LINE_CAP_STYLE 'diamond-cap'
```

NETGROUP_LINE_COLOR

Sets the line color for netgroups.

Syntax

```
NETGROUP_LINE_COLOR '<line_color>'
```

Where

`line_color`	The default line color for netgroups. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#33CC00'` (Green).

Examples

```
NETGROUP_LINE_COLOR '#003C77'
```

NETGROUP_LINE_JOIN_STYLE

Sets the type of corner created when two lines of a netgroup join or meet.

Syntax

```
NETGROUP_LINE_JOIN_STYLE '<join_style>'
```

Where

`join_style`	The default line join style for netgroup lines. Valid values: `miter-join`, `round-join` (default), `bevel-join`.

Examples

```
NETGROUP_LINE_JOIN_STYLE 'round-join'
NETGROUP_LINE_JOIN_STYLE 'bevel-join'
```

NETGROUP_LINE_STYLE

Specifies the default line style for netgroups.

Syntax

```
NETGROUP_LINE_STYLE '<line_style>'
```

Where

`line_style`	The default line style for netgroups. Valid values: `solid` (default), `dash`, `dot`, `dash-dot`, `dash-dot-dot`.

Examples

```
NETGROUP_LINE_STYLE 'dot'
NETGROUP_LINE_STYLE 'solid'
```

NETGROUP_LINE_WIDTH

Sets the default line width for netgroups.

Syntax

```
NETGROUP_LINE_WIDTH '<line_width>'
```

Where

`line_width`	The default line width for netgroups. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
NETGROUP_LINE_WIDTH '10'
```

NETGROUP_TEXT_FONT_BOLD

Specifies whether the netgroup text appears in **bold** face.

Syntax

```
NETGROUP_TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
NETGROUP_TEXT_FONT_BOLD 'FALSE'
```

NETGROUP_TEXT_FONT_COLOR

Sets the font color of the text for netgroups.

Syntax

```
NETGROUP_TEXT_FONT_COLOR '<font_color>'
```

Where

font_color	The default font color of the text for netgroups. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#000000'` (Black).

Examples

```
NETGROUP_TEXT_FONT_COLOR '#000000'
```

NETGROUP_TEXT_FONT_ITALIC

Specifies whether the netgroup text appears in *italics*.

Syntax

```
NETGROUP_TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
NETGROUP_TEXT_FONT_ITALIC 'FALSE'
```

NETGROUP_TEXT_FONT_NAME

Sets the font face used to display netgroup text.

Syntax

```
NETGROUP_TEXT_FONT_NAME '<font_name>'
```

Where

font_name	The default font face used to display netgroup text. The value can be any of the system-supported fonts. The default value is `'ARIAL'`.

Examples

```
NETGROUP_TEXT_FONT_NAME 'ARIAL BLACK'
```

NETGROUP_TEXT_FONT_SIZE

Sets the font size of the netgroup text.

Syntax

```
NETGROUP_TEXT_FONT_SIZE '<font_size>'
```

Where

`font_size`	The default font size used to display netgroup text. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
NETGROUP_TEXT_FONT_SIZE '10'
```

NETGROUP_TEXT_FONT_UNDERLINE

Specifies whether netgroup text appears underlined.

Syntax

```
NETGROUP_TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
NETGROUP_TEXT_FONT_UNDERLINE 'FALSE'
```

NETGROUP_TEXT_MARGIN

Sets the margin or the space around text in tables.

Syntax

```
NETGROUP_TEXT_MARGIN '<value>'
```

Where

`value`	Any positive integer value. The default value is `1`.

Examples

```
NETGROUP_TEXT_MARGIN '1'
NETGROUP_TEXT_MARGIN '0.5'
```

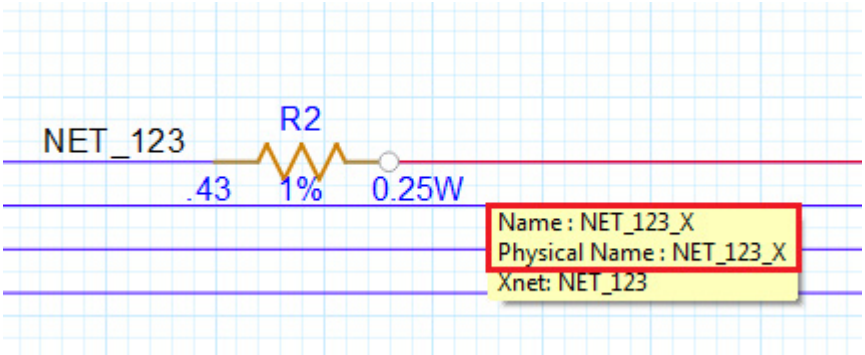
NETSPLIT_SUFFIX

Applies a suffix to the name of a net that is automatically created on dropping a two-pin discrete on a wire.

Syntax

```
NETSPLIT_SUFFIX '<suffix>'
```

suffix A single letter which is added as a suffix to the netname of a split net. The default value is 'X'. The following image illustrates that after a resistor is placed on a net named NET_123, the suffix 'X' is added to the netname:



Example

```
NETSPLIT_SUFFIX 'X'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - Net-Split Suffix

NEW_PROPERTY_VISIBILITY

Using this directive, you can set the default visibility to be used for the newly added properties on canvas.

Syntax

```
NEW_PROPERTY_VISIBILITY 'OFF'|'ON'
```

OFF	New properties are visible on the canvas.
ON	New properties are not visible on the canvas.

Example

```
NEW_PROPERTY_VISIBILITY 'OFF'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - General - Display value for newly added attributes

NO_CONNECT

Using this directive, you can add a symbol to the *No connect* section of the *Special Symbols* bucket that is to be used as an offpage connector for input signals.

Syntax

```
NO_CONNECT '<library:cell:view>'
```

where

library	Enter the library name in which you want the symbol to be added.
cell	Enter the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	Enter the view name in which you want the symbol to be added.

Example

```
NO_CONNECT 'standard.nc:sym_1' 'standard.nc:sym_2' 'standard.nc:sym_3'
     'standard.nc:sym_4'
```

Corresponding UI Option

None

NO_FEEDBACK

The NO_FEEDBACK directive disables property feedback from PCB Editor to preserve the value in the state file. Properties that are fed back from an PCB Editor board get to the schematic in three steps:

1. From the board to `*view.dat` files.
2. From `*view.dat` files to `pst*.dat` files.
3. From the `pstback.dat` file to the schematic.

Properties that are fed back from an PCB Editor board to `*view.dat` files are controlled through the `pxlBA.txt` file. This file lists the properties that are extracted from the board and stored in `*view.dat` files.

The NO_FEEDBACK directive is used for the properties that are fed back from `*view.dat` files to `pst*.dat` file. Properties specified through the NO_FEEDBACK directive, although present in the `*view.dat` files, are not updated or included in the `pst*.dat` files.

The NO_BACKANNOTATE property is used for the properties that are fed back from the `pstback.dat` file to the schematic. This property used on a per instance basis allows the prevention of the updated values appearing in the `pstback.dat` file and therefore does not update the values on the schematic. For more details, see Allegro Platform Properties Reference guide.

Syntax

```
NO_FEEDBACK property [,property] ... ;
```

property	Represents a property name in the schematic.

The default value for the NO_FEEDBACK directive is none, or feedback is allowed for all properties

Example

```
NO_FEEDBACK LOCATION;
```

NON_GRAPHIC_MODE_FOR_CM

Set this directive to launch Constraint Manager in the non-GUI mode. Constraint Manager will launch in the console mode.

Syntax

```
NON_GRAPHIC_MODE_FOR_CM 'ON'|'OFF'
```

Example

```
NON_GRAPHIC_MODE_FOR_CM 'ON'
```

Corresponding UI Option

None

NOTE_COLOR

Changes the default note color.

Syntax

```
NOTE_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
NOTE_COLOR 'PURPLE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Note

See Also

- BACKGROUND_COLOR
- DOT_COLOR
- OCCPROP_COLOR
- HIGHLIGHT_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

NUM_OLD_VERSIONS

The NUM_OLD_VERSIONS directive specifies the maximum number of old versions retained for each output file generated by the packager.

Syntax

```
NUM_OLD_VERSIONS number;
```

number	Number of versions

The default value for the NUM_OLD_VERSIONS directive is three.

Example

```
NUM_OLD_VERSIONS 3;
```

Packager-XL always produces the `pstchip.dat` file. If `pstchip.dat` files exist, Packager-XL starts numbering them as `pstchip.dat,1`, `pstchip.dat,2`, and `pstchip.dat,3`. The highest number represents the most recent file.

Note: The version number is kept consistent across the entire set of output files.

15

O Directives

This chapter lists the CPM directives that start with o and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

Text Object Types

This section explains the different types of text objects that <OBJECT> directives work on.

There are two categories of text objects in System Capture. The first category supports full Unicode characters and extensive formatting capabilities. These are purely for documentation and are not passed in the front to back or back to front flows.

The second category of text is applicable to component instances, signals, and properties. The values of these objects are passed in the flow, for example netlist.

- Documentation text objects:
  - Support printable Unicode characters
  - Support extensive text formatting, such as bold, italics, underline, or color, including the ability to format a portion of the selected text.
  - Include RICH_NOTEs, such as notes, text added to blocks, connectors, and tables.
- Flow text

  Printable ASCII characters that form valid signal names are supported. Formatting is applied to the entire text and not to a partial selection. Flow text include:
  - SIMPLE_NOTE

    Signal names on wires, buses, and NetGroups.
  - INST_TEXT

    Properties on primitives and block instances
  - ROUTE_TEXT

    Represents properties on wires and buses

<OBJECT>_FILL_STYLE

Sets the style that is used to fill shapes of the specified object type.

Syntax

```
<OBJECT>_STYLE '<fill_style>'
```

OBJECT	Represents the object type for which the directive is set.
fill_style	The default fill style for the specified object type. The fill style can be `solid` or `none`.

Examples

```
INST_FILL_STYLE 'solid'
PIN_FILL_STYLE 'none'
RAT_FILL_STYLE 'solid'
BUS_FILL_STYLE 'solid'
RICH_NOTE_FILL_STYLE 'none'
ROUTE_FILL_STYLE 'solid'
GRAPHIC_BLOCK_FILL_STYLE 'solid'
GRAPHIC_CONNECTOR_FILL_STYLE 'solid'
PROPERTY_FILL_STYLE 'none'
NOTE_FILL_STYLE 'none'
OCC_PROPERTY_FILL_STYLE  'none'
SIMPLE_NOTE_FILL_STYLE 'none'
```

<OBJECT>_LINE_CAP_STYLE

A line cap style defines the ending of a line. This directive defines the line cap style for the specified object type.

Syntax

```
<OBJECT>_LINE_CAP_STYLE '<line_cap_style>'
```

OBJECT	Represents the object type for which the directive is set.
line_cap_style	The default line cap style for the specified object type. A line cap style can be round-cap, square-cap, diamond-cap, or arrowhead-cap.

Example

```
INST_LINE_CAP_STYLE 'round-cap'
RAT_LINE_CAP_STYLE 'round-cap'
RICH_NOTE_LINE_CAP_STYLE 'diamond-cap'
OCC_PROPERTY_LINE_CAP_STYLE 'square-cap'
PIN_LINE_CAP_STYLE 'round-cap'
BUS_LINE_CAP_STYLE 'round-cap'
ROUTE_LINE_CAP_STYLE 'diamond-cap'
NOTE_LINE_CAP_STYLE 'square-cap'
PROPERTY_LINE_CAP_STYLE 'round-cap'
SIMPLE_NOTE_LINE_CAP_STYLE 'round-cap'
GRAPHIC_BLOCK_LINE_CAP_STYLE 'diamond-cap'
GRAPHIC_CONNECTOR_LINE_CAP_STYLE 'square-cap'
```

<OBJECT>_LINE_COLOR

Sets the line color for the specified object type.

Syntax

```
<OBJECT>_LINE_COLOR '<line_color>'
```

OBJECT	Represents the object type for which the directive is set.
line_color	The default line color for the specified object type. The value is specified in hex color code. For example, `#CC0000` and `#FF0000` represent shades of the color red, while `#0022CC` is a shade of blue.

Example

```
INST_LINE_COLOR '#CC0000'
RAT_LINE_COLOR '#FF0000'
RICH_NOTE_LINE_COLOR '#0022CC'
PIN_LINE_COLOR '#AA8844'
SIMPLE_NOTE_LINE_COLOR '#0000CC'
ROUTE_LINE_COLOR '#CC0000'
PROPERTY_LINE_COLOR '#FF0000'
NOTE_LINE_COLOR '#0022CC'
BUS_LINE_COLOR '#AA8844'
OCC_PROPERTY_LINE_COLOR  '#0000ff'
GRAPHIC_CONNECTOR_LINE_COLOR '#0000CC'
GRAPHIC_BLOCK_LINE_COLOR '#0000CC'
```

<OBJECT>_LINE_JOIN_STYLE

Sets the type of corner created when two lines of the specified object type join or meet.

Syntax

```
<OBJECT>_LINE_JOIN_STYLE '<line_join_style>'
```

OBJECT	Represents the object type for which the directive is set.
line_join_style	The default line join style for the specified object type. A line join style can be `miter-join`, `round-join`, or `bevel-join`.

Example

```
INST_LINE_JOIN_STYLE 'round-join'
PIN_LINE_JOIN_STYLE 'miter-join'
RAT_LINE_JOIN_STYLE 'bevel-join'
ROUTE_LINE_JOIN_STYLE 'round-join'
BUS_LINE_JOIN_STYLE 'round-join'
NOTE_LINE_JOIN_STYLE 'miter-join'
RICH_NOTE_LINE_JOIN_STYLE 'bevel-join'
SIMPLE_NOTE_LINE_JOIN_STYLE 'round-join'
PROPERTY_LINE_JOIN_STYLE 'round-join'
OCC_PROPERTY_LINE_JOIN_STYLE 'miter-join'
GRAPHIC_BLOCK_LINE_JOIN_STYLE 'bevel-join'
GRAPHIC_CONNECTOR_LINE_JOIN_STYLE 'round-join'
```

<OBJECT>_LINE_STYLE

Specifies the default line style of the specified object type.

Syntax

```
<OBJECT>_LINE_STYLE '<line_style>'
```

OBJECT	Represents the object type for which the directive is set.
line_style	The default line style of the objects of the specified type. The line style can be: ■ solid ■ dash ■ dot ■ dash-dot ■ dash-dot-dot GUI equivalent of line style: 

Example

```
PIN_LINE_STYLE 'SOLID'
RAT_LINE_STYLE 'DASH'
RICH_NOTE_LINE_STYLE 'DOT'
OCC_PROPERTY_LINE_STYLE 'DASH-DOT'
INST_LINE_STYLE 'DASH-DOT-DOT'
BUS_LINE_STYLE 'DOT'
ROUTE_LINE_STYLE 'SOLID'
NOTE_LINE_STYLE 'DASH'
PROPERTY_LINE_STYLE 'SOLID'
SIMPLE_NOTE_LINE_STYLE 'DASH'
```

```
GRAPHIC_BLOCK_STYLE_LINE_STYLE 'SOLID'
GRAPHIC_CONNECTOR_STYLE_LINE_STYLE 'DASH'
```

<OBJECT>_LINE_WIDTH

Specifies the default line width of the specified object type.

Syntax

```
<OBJECT>_LINE_WIDTH '<line_width>'
```

OBJECT	Represents the object type for which the directive is set.
line_width	The default line width of the objects of the specified type. The value can range from 0 to 100.

Example

```
INST_LINE_WIDTH '2'
PIN_LINE_WIDTH '1'
RAT_LINE_WIDTH '2'
BUS_LINE_WIDTH '1'
ROUTE_LINE_WIDTH '2'
NOTE_LINE_WIDTH '1'
PROPERTY_LINE_WIDTH '2'
OCC_PROPERTY_LINE_WIDTH '1'
OUTLINE_LINE_WIDTH '3'
RICH_NOTE_LINE_WIDTH '2'
SIMPLE_NOTE_LINE_WIDTH '1'
GRAPHIC_BLOCK_STYLE_LINE_WIDTH '2'
GRAPHIC_CONNECTOR_STYLE_LINE_WIDTH '1'
```

<OBJECT>_TEXT_FONT_BOLD

Specifies whether the text for the object appears in **bold** face.

Syntax

```
<OBJECT>_TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

OBJECT	Represents the object type for which the directive is set.

Examples

```
INST_TEXT_FONT_BOLD 'FALSE'
PIN_TEXT_FONT_BOLD 'FALSE'
BUS_TEXT_FONT_BOLD 'FALSE'
NOTE_TEXT_FONT_BOLD 'FALSE'
RAT_TEXT_FONT_BOLD 'FALSE'
ROUTE_TEXT_FONT_BOLD 'FALSE'
RICH_NOTE_TEXT_FONT_BOLD 'TRUE'
PROPERTY_TEXT_FONT_BOLD 'FALSE'
OCC_PROPERTY_TEXT_FONT_BOLD 'FALSE'
SIMPLE_NOTE_TEXT_FONT_BOLD 'FALSE'
GRAPHIC_CONNECTOR_TEXT_FONT_BOLD 'FALSE'
```

<OBJECT>_TEXT_FONT_COLOR

Sets the color of the text for the specified object type.

Syntax

```
<OBJECT>_TEXT_FONT_COLOR 'font_color'
```

OBJECT	Represents the object type for which the directive is set.
font_color	The default color of the text for the specified object type. The value can be expressed as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red.

Examples

```
INST_TEXT_FONT_COLOR '#000000'
PIN_TEXT_FONT_COLOR '#FF0000'
RAT_TEXT_FONT_COLOR '#0000FF'
ROUTE_TEXT_FONT_COLOR '#0000FF'
BUS_TEXT_FONT_COLOR '#00FF00'
RICH_NOTE_TEXT_FONT_COLOR '#00FF00'
NOTE_TEXT_FONT_COLOR '#00FF00'
SIMPLE_NOTE_TEXT_FONT_COLOR '#00FF00'
GRAPHIC_BLOCK_TEXT_FONT_COLOR 'BLUE'
GRAPHIC_CONNECTOR_TEXT_FONT_COLOR '#000000'
PROPERTY_TEXT_FONT_COLOR '#000000'
OCC_PROPERTY_TEXT_FONT_COLOR '#000000'
```

<OBJECT>_TEXT_FONT_ITALIC

Specifies whether the text for the specified object appears in *italics*.

Syntax

```
<OBJECT>_TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

OBJECT	Represents the object type for which the directive is set.

Examples

```
BUS_TEXT_FONT_ITALIC 'FALSE'
NOTE_TEXT_FONT_ITALIC 'TRUE'
INST_TEXT_FONT_ITALIC 'FALSE'
PIN_TEXT_FONT_ITALIC 'TRUE'
RAT_TEXT_FONT_ITALIC 'FALSE'
ROUTE_TEXT_FONT_ITALIC 'FALSE'
RICH_NOTE_TEXT_FONT_ITALIC 'TRUE'
PROPERTY_TEXT_FONT_ITALIC 'FALSE'
OCC_PROPERTY_TEXT_FONT_ITALIC 'FALSE'
SIMPLE_NOTE_TEXT_FONT_ITALIC 'FALSE'
GRAPHIC_CONNECTOR_TEXT_FONT_ITALIC 'FALSE'
```

<OBJECT>_TEXT_FONT_NAME

Sets the font face used to display the text for the specified object type.

Syntax

```
<OBJECT>_TEXT_FONT_NAME '<font_name>'
```

OBJECT	Represents the object type for which the directive is set.
font_name	The default font face used to display the text for the specified object type. The value can be any of the system-supported fonts.

Examples

```
INST_TEXT_FONT_NAME 'Arial'
PIN_TEXT_FONT_NAME 'Helvetica'
RAT_TEXT_FONT_NAME 'Courier'
ROUTE_TEXT_FONT_NAME 'Courier'
RICH_NOTE_TEXT_FONT_NAME 'Helvetica'
BUS_TEXT_FONT_NAME 'Segoe UI'
NOTE_TEXT_FONT_NAME 'Times'
PROPERTY_TEXT_FONT_NAME 'Tahoma'
OCC_PROPERTY_TEXT_FONT_NAME 'Courier New'
SIMPLE_NOTE_TEXT_FONT_NAME 'Arial'
GRAPHIC_BLOCK_TEXT_FONT_NAME 'Tahoma'
GRAPHIC_CONNECTOR_TEXT_FONT_NAME 'Arial'
```

<OBJECT>_TEXT_FONT_SIZE

Sets the size of the text for the specified object type.

Syntax

```
<OBJECT>_TEXT_FONT_SIZE '<font_size>'
```

OBJECT	Represents the object type for which the directive is set.
font_size	The default font size used to display the text for the specified object type.

Examples

```
INST_TEXT_FONT_SIZE '5'
PIN_TEXT_FONT_SIZE '5'
OCC_PROPERTY_TEXT_FONT_SIZE '5'
BUS_TEXT_FONT_SIZE '5'
NOTE_TEXT_FONT_SIZE '10'
RAT_TEXT_FONT_SIZE '5'
ROUTE_TEXT_FONT_SIZE '5'
RICH_NOTE_TEXT_FONT_SIZE '20'
PROPERTY_TEXT_FONT_SIZE '3'
SIMPLE_NOTE_TEXT_FONT_SIZE '10'
GRAPHIC_BLOCK_TEXT_FONT_SIZE '30'
GRAPHIC_CONNECTOR_TEXT_FONT_SIZE '15'
```

<OBJECT>_TEXT_FONT_UNDERLINE

Specifies whether the text for the object appears underlined.

Syntax

```
<OBJECT>_TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

OBJECT	Represents the object type for which the directive is set.

Examples

```
INST_TEXT_FONT_UNDERLINE 'FALSE'
PIN_TEXT_FONT_UNDERLINE 'FALSE'
BUS_TEXT_FONT_UNDERLINE 'FALSE'
NOTE_TEXT_FONT_UNDERLINE 'FALSE'
RAT_TEXT_FONT_UNDERLINE 'FALSE'
ROUTE_TEXT_FONT_UNDERLINE 'FALSE'
RICH_NOTE_TEXT_FONT_UNDERLINE 'FALSE'
PROPERTY_TEXT_FONT_UNDERLINE 'FALSE'
OCC_PROPERTY_TEXT_FONT_UNDERLINE 'FALSE'
SIMPLE_NOTE_TEXT_FONT_UNDERLINE 'FALSE'
GRAPHIC_CONNECTOR_TEXT_FONT_UNDERLINE 'FALSE'
```

OCCPROP_COLOR

Changes the default occurrence property color.

Syntax

```
OCCPROP_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
OCCPROP_COLOR 'PURPLE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Occurrence Property

See Also

- BACKGROUND_COLOR
- DOT_COLOR
- NOTE_COLOR
- HIGHLIGHT_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

OFFPAGE

Use this directive to specify the library:cell:view of the symbol to be used as off page connector in the generated document schematic.

Syntax

```
offpage <library.cell:view>
```

Example

```
offpage 'standard.offpage:sym_1'
```

Corresponding UI Option for System Connectivity Manager

None

OFFPAGE_INPUT

Using this directive you can add a symbol to the *Offpage* section of the *Special Symbols* bucket that is to be used as an offpage connector for input signals.

Syntax

```
OFFPAGE_INPUT '<library:cell:view>'
```

where

library	Enter the library name in which you want the symbol to be added.
cell	Enter the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	Enter the view name in which you want the symbol to be added.

Example

```
OFFPAGE_INPUT 'standard.offpage:sym_1'
```

Corresponding UI Option

None

OFFPAGE_IO

Using this directive you can add a symbol to the *Offpage* section of the *Special Symbols* bucket that is to be used as an offpage connector for InOut signals.

Syntax

```
OFFPAGE_IO '<library:cell:view>'
```

where

library	Enter the library name in which you want the symbol to be added.
cell	Enter the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	Enter the view name in which you want the symbol to be added.

Example

```
OFFPAGE_IO 'standard.offpage:sym_3'
```

Corresponding UI Option

None

OFFPAGE_OUTPUT

Using this directive you can add a symbol to the *Offpage* section of the *Special Symbols* bucket that is to be used as an offpage connector for output signals.

Syntax

```
OFFPAGE_OUTPUT '<library:cell:view>'
```

where

library	Enter the library name in which you want the symbol to be added.
cell	Enter the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	Enter the view name in which you want the symbol to be added.

Example

```
OFFPAGE_OUTPUT 'standard.offpage:sym_2'
```

Corresponding UI Option

None

old_versions_count

This directive can be used to limit the number of backup PTF files that are created for the `part_table.ptf` file in `flatlib`. This directive can be set at the site or project level.

Syntax

```
old_versions_count '<integer count >'
```

Example

```
old_versions_count '10'
```

Corresponding UI Option

None

OMIT_CELL_FROM_CREF_PARTS

Suppresses the design name and the white space from the crefparts report. Note that this directive eliminates the extra white spaces due to the removal of the design name.

Syntax

```
OMIT_CELL_FROM_CREF_PARTS 'ON'|'OFF'
```

Example

```
OMIT_CELL_FROM_CREF_PARTS 'ON'
```

Corresponding UI Option

None

See Also

OMIT_CELL_FROM_CREF_PARTS

OMIT_CREFPARTS_HIERARCHY

Use the OMIT_CREFPARTS_HIERARCHY directive to omit higher level cells from the parts by page report (crefparts.txt file), when the flattened schematic (schcref view) is generated.

Value

```
OMIT_CREFPARTS_HIERARCHY 'ON'|'OFF'
```

Example

```
OMIT_CREFPARTS_HIERARCHY 'ON'
```

Corresponding UI Option

None

See Also

OMIT_CELL_FROM_CREF_PARTS

OMIT_DOWN_HIERARCHY

Omits cross-references down the hierarchy.

Value

```
OMIT_DOWN_HIERARCHY 'ON'|'OFF'
```

Example

```
OMIT_DOWN_HIERARCHY 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Format page — Formatting Options — Omit Xrefs Down Hierarchy

See Also

OMIT_CELL_FROM_CREF_PARTS

OMIT_CREFPARTS_HIERARCHY

OMIT_ZONE_INFO

OMIT_ZONE_INFO

Creates cross reference by page number only and omits the zone (page grid) information.

Note: By default, CRefer includes information about the page border zones in the cross references. For example, when this directive is set to OFF, a signal may have the following cross reference 1C7^. However, when the directive is set to ON, CRefer records the cross reference as 1^.

Value

```
OMIT_ZONE_INFO 'ON'|'OFF'
```

Example

```
OMIT_ZONE_INFO 'ON'
```

Corresponding UI Option

Project Manager: Tools — CRefer — Options — Cross Referencer Options dialog box — Content page — Write Options — Omit Zone Information

See Also

OMIT_CELL_FROM_CREF_PARTS

OMIT_CREFPARTS_HIERARCHY

OMIT_DOWN_HIERARCHY

Online_Mode

Lets you specify the mode for the subsequent launch of Part Information Manager:

- Database
- Cache

Syntax

```
Online_Mode 'False'| 'True'
```

Examples

```
Online_Mode 'False'
```

This will launch Part Information Manager in the cache mode.

```
Online_Mode 'True'
```

This will launch Part Information Manager in the database mode.

Corresponding UI Option

Configuration – Setup – General – Launch Mode for Next Invocation

OPEN_ONLY_ACTIVE_TAB

When opening a design, only first tab is opened and the other open tabs from last tool invocation are ignored.

Syntax

```
OPEN_ONLY_ACTIVE_TAB 'TRUE'|'FALSE'
```

Example

```
OPEN_ONLY_ACTIVE_TAB 'TRUE'
```

OPTIMIZE

The OPTIMIZE directive specifies that existing assignments can be modified in order to optimize the packaged design.

Optimization operates as follows:

- Minimizes the number of packages used in a design by condensing free slots.
- Affects only instances with multiple slots to minimize the number of slots swapped.
- Removes unused packages from the design.

  No new packages are created during optimization.

To use the OPTIMIZE directive, package the design, create a state file containing the packaging for the design, and make changes to the schematic or layout.

Since Packager-XL by default attempts to preserve the existing packaging assignments, changes to the design can result in less than optimal packaging.

Syntax

```
OPTIMIZE on|off ;
```

The default value for the OPTIMIZE directive is off.

Example

```
OPTIMIZE on;
```

ORIENT_NET_NAME_DISPLAY

Controls whether the net names are displayed vertically along the vertical wire and bus segments.

Syntax

```
ORIENT_NET_NAME_DISPLAY 'YES'|'NO'
```

YES	The orientation of netnames is changed to vertical for vertical wire and bus segments.
NO	The orientation of all netnames is horizontal.

Example

```
ORIENT_NET_NAME_DISPLAY 'YES'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - Display Net Name on Vertical Segment

OUT_PORT_PIN_SIDE

This directive defines the default placement location of the output ports on the symbol outline.

Syntax

```
OUT_PORT_PIN_SIDE 'LEFT'|'RIGHT'|'UP'|'DOWN'
```

LEFT	Output ports are automatically placed on the left side of the symbol outline.
RIGHT	Output ports are automatically placed on the right side of the symbol outline. This is the default value.
UP	Output ports are automatically placed on the top side of the symbol outline.
DOWN	Output ports are automatically placed on the bottom side of the symbol outline.

Example

```
OUT_PORT_PIN_SIDE 'LEFT'
OUT_PORT_PIN_SIDE 'UP'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Wiring/Ports - Out

OUTPORT

The OUTPORT directive is used in Allegro System Capture and Schgen. This directive is used to add output port to the *Port* section of the *Special Symbols* bucket.

Syntax

```
OUTPORT '<library:cell:view>'
```

where

library	is the library name in which you want the symbol to be added.
cell	is the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	is the view name in which you want the symbol to be added.

Example

```
OUTPORT 'standard.outport:sym_1'
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for System Connectivity Manager

None

See Also

INPORT

IOPORT

OUTPUT

The OUTPUT directive specifies the output files that Packager-XL writes. If you omit this directive, Packager-XL writes the `netlist`, `report`, `xref`, `and pinlist` files.

The OUTPUT directive can appear more than once in the project file.

Syntax

```
OUTPUT on|off|output_file[,output_file]...;
```

`on`	Generates all output files.
`off`	Prevents the generation of output files.
`output_file`	Possible output files include: ■ `netlist` - Generates the `pstxprt`, `pstxnet`, and `pstchip` files. ■ `changes` - Generates the `pxl.chg` file. ■ `report` - Generates the `pstrprt.dat` file. ■ `xref` - Generates the `pstxref.dat` file. ■ `pinlist` - Generates the `pstpin.dat` file.

The default value for the OUTPUT directive is on.

Example

```
OUTPUT netlist;
```

OUTPUT_ASCII

Saves an ASCII representation of the logic.

Syntax

```
OUTPUT_ASCII 'ON'|'OFF'
```

Example

```
OUTPUT_ASCII 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write— ASCII File

See Also

- OUTPUT_BINARY
- OUTPUT_DEPENDENCY

OUTPUT_BINARY

Saves a binary representation of the logic.

Syntax

```
OUTPUT_BINARY 'ON'|'OFF'
```

Example

```
OUTPUT_BINARY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write — Binary File

See Also

- OUTPUT_ASCII
- OUTPUT_DEPENDENCY

OUTPUT_DEPENDENCY

Saves an ASCII file with dependency information.

Syntax

```
OUTPUT_DEPENDENCY 'ON'|'OFF'
```

Example

```
OUTPUT_DEPENDENCY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Schematic Write — Dependency File

See Also

- OUTPUT_BINARY
- OUTPUT_ASCII

OUTPUT_VERILOG

Creates a Verilog text representation of the design when it is saved. This is always created when the Create Netlist option is on.

Syntax

```
OUTPUT_VERILOG 'ON'|'OFF'
```

Example

```
OUTPUT_VERILOG 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Create Netlist — Verilog

See Also

- OUTPUT_BINARY
- OUTPUT_ASCII
- OUTPUT_VHDL

OUTPUT_VHDL

Creates a VHDL text representation of the design when it is saved.

Syntax

```
OUTPUT_VHDL 'ON'|'OFF'
```

Example

```
OUTPUT_VHDL 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Output page — Create Netlist — VHDL

See Also

- OUTPUT_VERILOG
- OUTPUT_BINARY
- OUTPUT_ASCII

OVERWRITE_CONSTRAINTS

Use this directive to determine if all electrical constraints in the board file must be overwritten using values in the logical design or only those constraints in the logical design that have changed since the previous export.

Syntax

```
overwrite_constraints 'ON'|'OFF'
```

Example

```
overwrite_constraints 'OFF'
```

Corresponding UI Option for System Connectivity Manager

Project — Export — PCB Board — Constraint Manager Data

See Also

- ETCH_REMOVAL
- GEN_PSTFILES
- IGNORE_FIXED

16

P Directives

This chapter lists the CPM directives that start with `P` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

Package_Class

The `Package_Class` directive specifies the default `CLASS` value of a part.

Syntax

```
Package_Class '<class_name>'
```

Example

```
Package_Class 'IC'
```

Corresponding UI Option for Part Developer

Class list on the *General* page in the Package Editor

Package_JedecType

The `Package_JedecType` directive specifies the default `JEDEC_TYPE` value of a part.

Syntax

```
Package_JedecType '<VALUE>'
```

Example

```
Package_JedecType ''
```

Corresponding UI Option for Part Developer

Jedec Type list on the *General* page in the Package Editor

Package_PinDelayUnit

The `Package_PinDelayUnit` directive specifies the default unit for pin delay.

Syntax

```
Package_PinDelayUnit '<pin_delay_unit>'
```

Example

```
Package_PinDelayUnit 'ns'
```

Corresponding UI Option for Part Developer

PIN_DELAY Units list box on *Tools – Setup – Package Pins*

PACKAGE_PROP

The PACKAGE_PROP directive is used in Allegro System Capture and Packager-XL. This directive specifies the properties that control packaging, which cause Packager-XL to keep together schematic instances with properties of equal value. Packager-XL does not package together any instances that have different values for the same property. However, if spare slots are available, instances without the package properties can be added

The PACKAGE_PROP Directive is also used to specify the properties that need to be ignored during packaging in System Connectivity Manager.

The PACKAGE_PROP directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

For information about preventing the packaging together of schematic instances with property values and schematic instances without property values, see the "STRICT_PACKAGE_PROP" on page 469.

Syntax

```
PACKAGE_PROP property [,property] ...;
```

where

property	is a property name such as GROUP, ROOM, or COMPONENT_WEIGHT.

The default properties are GROUP and ROOM.

Example

```
PACKAGE_PROP group, room, component_weight;
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Packager-XL

None

Corresponding UI Option for System Connectivity Manager

None

Package_RefDesPrefix

The `Package_RefDesPrefix` directive specifies the default reference designator of a part.

Syntax

```
Package_RefDesPrefix '<reference_designator>'
```

Example

```
Package_RefDesPrefix 'U'
```

Corresponding UI Option for Part Developer

RefDes Prefix list on the *General* page in the Package Editor

PACKAGED_FOLDER

Specify the location where you want the packaged files to be generated.

Syntax

```
PACKAGED_FOLDER '<path>'
```

where

<path>	Specify the path where you want the packaged files to be generated.

Example

```
PACKAGED_FOLDER 'C:\Designs'
```

Corresponding UI Option

None

PackagePin_AbsentChar

The `PackagePin_AbsentChar` directive specifies the character that Part Developer uses to indicate that a pin is unmapped. The default value is -.

Important

It is recommended that you do not modify the default value of the `PackagePin_AbsentChar` directive.

Syntax

```
PackagePin_AbsentChar '<character>'
```

Example

```
PackagePin_AbsentChar '-'
```

Corresponding UI Option for Part Developer

None

PackagePin_Property_ANALOG_Assert

The `PackagePin_Property_ANALOG_Assert` directive determines if pins of type ANALOG are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_ANALOG_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_ANALOG_Assert 'False'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_ANALOG_Connect

The `PackagePin_Property_ANALOG_Connect` directive determines if pins of type ANALOG are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_ANALOG_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_ANALOG_Connect 'False'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_ANALOG_Dir

The `PackagePin_Property_ANALOG_Dir` determines if pins of type ANALOG are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_ANALOG_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_ANALOG_Dir 'False'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_ANALOG_IO

The `PackagePin_Property_ANALOG_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type ANALOG. The directive controls the value of the `NO_IO_CHECK` property in chips.prt.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_ANALOG_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_ANALOG_IO 'Off'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_ANALOG_Load

The `PackagePin_Property_ANALOG_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type ANALOG.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_ANALOG_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_ANALOG_Load 'Off'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_ANALOG_UnknownLoading

The `PackagePin_Property_ANALOG_UnknownLoading` directive specifies if load checking is to be done for pins of type ANALOG. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_ANALOG_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_ANALOG_UnknownLoading 'False'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_BIDIR_Assert

The `PackagePin_Property_BIDIR_Assert` directive determines if pins of type BIDIR are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_BIDIR_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_BIDIR_Assert 'True'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_BIDIR_Connect

The `PackagePin_Property_BIDIR_Connect` directive determines if pins of type BIDIR are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_BIDIR_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_BIDIR_Connect 'True'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_BIDIR_Dir

The `PackagePin_Property_BIDIR_Dir` determines if pins of type BIDIR are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_BIDIR_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_BIDIR_Dir 'True'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_BIDIR_IO

The `PackagePin_Property_BIDIR_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type BIDIR. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_BIDIR_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_BIDIR_IO 'Both'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_BIDIR_Load

The `PackagePin_Property_BIDIR_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type BIDIR.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_BIDIR_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_BIDIR_Load 'Both'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_BIDIR_UnknownLoading

The `PackagePin_Property_BIDIR_UnknownLoading` directive specifies if load checking is to be done for pins of type BIDIR. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_BIDIR_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_BIDIR_UnknownLoading 'False'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_GROUND_Assert

The `PackagePin_Property_GROUND_Assert` directive determines if pins of type GROUND are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_GROUND_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_GROUND_Assert 'False'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_GROUND_Connect

The `PackagePin_Property_GROUND_Connect` directive determines if pins of type GROUND are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_GROUND_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_GROUND_Connect 'False'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_GROUND_Dir

The `PackagePin_Property_GROUND_Dir` determines if pins of type GROUND are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_GROUND_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_GROUND_Dir 'False'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_GROUND_IO

The `PackagePin_Property_GROUND_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type GROUND. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_GROUND_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_GROUND_IO 'Off'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_GROUND_Load

The `PackagePin_Property_GROUND_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type GROUND.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_GROUND_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_GROUND_Load 'Off'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_GROUND_UnknownLoading

The `PackagePin_Property_GROUND_UnknownLoading` directive specifies if load checking is to be done for pins of type GROUND. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_GROUND_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_GROUND_UnknownLoading 'False'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_INPUT_Assert

The `PackagePin_Property_INPUT_Assert` directive determines if pins of type INPUT are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_INPUT_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_INPUT_Assert 'True'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_INPUT_Connect

The `PackagePin_Property_INPUT_Connect` directive determines if pins of type INPUT are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_INPUT_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_INPUT_Connect 'True'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_INPUT_Dir

The `PackagePin_Property_INPUT_Dir` determines if pins of type INPUT are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_INPUT_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_INPUT_Dir 'True'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_INPUT_IO

The `PackagePin_Property_INPUT_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type INPUT. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_INPUT_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_INPUT_IO 'Both'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_INPUT_Load

The `PackagePin_Property_INPUT_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type INPUT.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_INPUT_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_INPUT_Load 'Both'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_INPUT_UnknownLoading

The `PackagePin_Property_INPUT_UnknownLoading` directive specifies if load checking is to be done for pins of type INPUT. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_INPUT_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_INPUT_UnknownLoading 'False'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_NC_Assert

The `PackagePin_Property_NC_Assert` directive determines if pins of type NC are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_NC_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_NC_Assert 'False'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_NC_Connect

The `PackagePin_Property_NC_Connect` directive determines if pins of type NC are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_NC_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_NC_Connect 'False'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_NC_Dir

The `PackagePin_Property_NC_Dir` determines if pins of type NC are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_NC_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_NC_Dir 'FALSE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_NC_IO

The `PackagePin_Property_NC_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type NC. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_NC_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_NC_IO 'Off'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_NC_Load

The `PackagePin_Property_NC_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type NC.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_NC_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_NC_Load 'Off'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_NC_UnknownLoading

The `PackagePin_Property_NC_UnknownLoading` directive specifies if load checking is to be done for pins of type NC. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_NC_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_NC_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OC_Assert

The `PackagePin_Property_OC_Assert` directive determines if pins of type OC are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_OC_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OC_Assert 'TRUE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OC_Connect

The `PackagePin_Property_OC_Connect` directive determines if pins of type OC are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OC_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OC_Connect 'TRUE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_OC_Dir

The `PackagePin_Property_OC_Dir` determines if pins of type OC are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_OC_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OC_Dir 'TRUE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OC_IO

The `PackagePin_Property_OC_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type OC. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OC_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OC_IO 'BOTH'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OC_Load

The `PackagePin_Property_OC_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type OC.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OC_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OC_Load 'BOTH'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OC_UnknownLoading

The `PackagePin_Property_OC_UnknownLoading` directive specifies if load checking is to be done for pins of type OC. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_OC_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OC_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OCBIDIR_Assert

The `PackagePin_Property_OCBIDIR_Assert` directive determines if pins of type OCBIDIR are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_OCBIDIR_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OCBIDIR_Assert 'TRUE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OCBIDIR_Connect

The `PackagePin_Property_OCBIDIR_Connect` directive determines if pins of type OCBIDIR are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OCBIDIR_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OCBIDIR_Connect 'TRUE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_OCBIDIR_Dir

The `PackagePin_Property_OCBIDIR_Dir` determines if pins of type OCBIDIR are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_OCBIDIR_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OCBIDIR_Dir 'TRUE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OCBIDIR_IO

The `PackagePin_Property_OCBIDIR_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type OCBIDIR. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OCBIDIR_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OCBIDIR_IO 'BOTH'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OCBIDIR_Load

The `PackagePin_Property_OCBIDIR_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type OCBIDIR.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OCBIDIR_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OCBIDIR_Load 'BOTH'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OCBIDIR_UnknownLoading

The `PackagePin_Property_OCBIDIR_UnknownLoading` directive specifies if load checking is to be done for pins of type OCBIDIR. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_OCBIDIR_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OCBIDIR_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OE_Assert

The `PackagePin_Property_OE_Assert` directive determines if pins of type OE are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_OE_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OE_Assert 'TRUE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OE_Connect

The `PackagePin_Property_OE_Connect` directive determines if pins of type OE are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OE_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OE_Connect 'TRUE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_OE_Dir

The `PackagePin_Property_OE_Dir` determines if pins of type OE are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_OE_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OE_Dir 'TRUE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OE_IO

The `PackagePin_Property_OE_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type OE. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OE_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OE_IO 'BOTH'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OE_Load

The `PackagePin_Property_OE_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type OE.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OE_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OE_Load 'BOTH'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OE_UnknownLoading

The `PackagePin_Property_OE_UnknownLoading` directive specifies if load checking is to be done for pins of type OE. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_OE_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OE_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OEBIDIR_Assert

The `PackagePin_Property_OEBIDIR_Assert` directive determines if pins of type OEBIDIR are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_OEBIDIR_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OEBIDIR_Assert 'TRUE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OEBIDIR_Connect

The `PackagePin_Property_OEBIDIR_Connect` directive determines if pins of type OEBIDIR are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OEBIDIR_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OEBIDIR_Connect 'TRUE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_OEBIDIR_Dir

The `PackagePin_Property_OEBIDIR_Dir` determines if pins of type OEBIDIR are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PsackagePin_Property_OEBIDIR_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OEBIDIR_Dir 'TRUE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OEBIDIR_IO

The `PackagePin_Property_OEBIDIR_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type OEBIDIR. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OEBIDIR_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OEBIDIR_IO 'BOTH'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OEBIDIR_Load

The `PackagePin_Property_OEBIDIR_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type OEBIDIR.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OEBIDIR_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OEBIDIR_Load 'BOTH'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OEBIDIR_UnknownLoading

The `PackagePin_Property_OEBIDIR_UnknownLoading` directive specifies if load checking is to be done for pins of type OEBIDIR. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_OEBIDIR_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OEBIDIR_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OUTPUT_Assert

The `PackagePin_Property_OUTPUT_Assert` directive determines if pins of type OUTPUT are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_OUTPUT_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OUTPUT_Assert 'TRUE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OUTPUT_Connect

The `PackagePin_Property_OUTPUT_Connect` directive determines if pins of type OUTPUT are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OUTPUT_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OUTPUT_Connect 'TRUE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_OUTPUT_Dir

The `PackagePin_Property_OUTPUT_Dir` determines if pins of type OUTPUT are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_OUTPUT_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OUTPUT_Dir 'TRUE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OUTPUT_IO

The `PackagePin_Property_OUTPUT_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type OUTPUT. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OUTPUT_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OUTPUT_IO 'BOTH'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OUTPUT_Load

The `PackagePin_Property_OUTPUT_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type OUTPUT.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_OUTPUT_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_OUTPUT_Load 'BOTH'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_OUTPUT_UnknownLoading

The `PackagePin_Property_OUTPUT_UnknownLoading` directive specifies if load checking is to be done for pins of type OUTPUT. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_OUTPUT_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_OUTPUT_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_POWER_Assert

The `PackagePin_Property_POWER_Assert` directive determines if pins of type POWER are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_POWER_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_POWER_Assert 'FALSE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_POWER_Connect

The `PackagePin_Property_POWER_Connect` directive determines if pins of type POWER are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_POWER_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_POWER_Connect 'FALSE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_POWER_Dir

The `PackagePin_Property_POWER_Dir` determines if pins of type POWER are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_POWER_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_POWER_Dir 'FALSE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_POWER_IO

The `PackagePin_Property_POWER_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type POWER. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_POWER_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_POWER_IO 'Off'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_POWER_Load

The `PackagePin_Property_POWER_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type POWER.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_POWER_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_POWER_Load 'Off'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_POWER_UnknownLoading

The `PackagePin_Property_POWER_UnknownLoading` directive specifies if load checking is to be done for pins of type POWER. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_POWER_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_POWER_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_UNSPEC_Assert

The `PackagePin_Property_UNSPEC_Assert` directive determines if pins of type UNSPEC are to have an assertion check. The directive controls the value of the `NO_ASSERT_CHECK` property in chips.

Syntax

```
PackagePin_Property_UNSPEC_Assert 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_UNSPEC_Assert 'FALSE'
```

Corresponding UI Option for Part Developer

Check Assert column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_UNSPEC_Connect

The `PackagePin_Property_UNSPEC_Connect` directive determines if pins of type UNSPEC are to have an output check. The directive controls the value of the `ALLOW_CONNECT` property in chips.

For more information, see the *Check Output* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_UNSPEC_Connect 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_UNSPEC_Connect 'FALSE'
```

Corresponding UI Option for Part Developer

Check Output column in the *Package Pin Properties* grid on *Tools* – Setup – Package Pins

PackagePin_Property_UNSPEC_Dir

The `PackagePin_Property_UNSPEC_Dir` determines if pins of type UNSPEC are to have a direction check. The directive controls the value of the `NO_DIR_CHECK` property in chips.

Syntax

```
PackagePin_Property_UNSPEC_Dir 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_UNSPEC_Dir 'FALSE'
```

Corresponding UI Option for Part Developer

Check Dir column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_UNSPEC_IO

The `PackagePin_Property_UNSPEC_IO` directive specifies how the inputio_check Rules Checker rule is to be run for pins of type UNSPEC. The directive controls the value of the `NO_IO_CHECK` property in chips.

For more information, see the *Check IO* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_UNSPEC_IO 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_UNSPEC_IO 'Off'
```

Corresponding UI Option for Part Developer

Check IO column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_UNSPEC_Load

The `PackagePin_Property_UNSPEC_Load` directive specifies how the loading_check Rules Checker rule is to be run for pins of type UNSPEC.

For more information, see the *Check Load* section in the Configuring Part Developer chapter of Part Developer User Guide.

Syntax

```
PackagePin_Property_UNSPEC_Load 'Both'|'High'|'Low'|'Off'
```

Example

```
PackagePin_Property_UNSPEC_Load 'Off'
```

Corresponding UI Option for Part Developer

Check Load column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PackagePin_Property_UNSPEC_UnknownLoading

The `PackagePin_Property_UNSPEC_UnknownLoading` directive specifies if load checking is to be done for pins of type UNSPEC. The directive controls the value of the `UNKNOWN_LOADING` property in chips.

Syntax

```
PackagePin_Property_UNSPEC_UnknownLoading 'TRUE'|'FALSE'
```

Example

```
PackagePin_Property_UNSPEC_UnknownLoading 'FALSE'
```

Corresponding UI Option for Part Developer

Unknown Loading column in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

PAGE_BORDER

Use this directive to specify the lib:cell:view of the symbol to be used as page border during the document schematic generation process.

Syntax

```
page_border <library.cell:view>
```

Example

```
page_border 'standard.b size page:sym_1'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Page Border.

PAGE_DEFAULT_SIZE

Using this directive, you can set the default page size for the schematic.

Syntax

```
PAGE_DEFAULT_SIZE 'A'|'B'|'C'|'D'|'E'
```

Following table describes the dimensions of the available pages:

A	Height: 7.2 inches
	Width: 9.7 inches
B	Height: 9.2 inches
	Width: 15.2 inches
C	Height: 15.2 inches
	Width: 20.2 inches
D	Height: 32.2 inches
	Width: 20.2 inches
E	Height: 42.2 inches
	Width: 32.7 inches

Example

```
PAGE_DEFAULT_SIZE 'A'
```

PAGE_DOUBLE_WIDTH

Use this directive to specify the width to be used for thicker wires while generating the PDF output file.

Syntax

```
PAGE_DOUBLE_WIDTH '<numerical value>'
```

Example

```
PAGE_DOUBLE_WIDTH '3'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Line Width — Double

See Also

PAGE_SINGLE_WIDTH

PAGE_HEIGHT

If you choose a custom page size to publish your PDF files, you can specify the height of the page using this directive.

Syntax

```
PAGE_HEIGHT '<numerical value>'
```

Example

```
PAGE_ HEIGHT '10.20'
```

Corresponding UI Option for Design Entry HDL

File —Publish PDF — Setup — PDF — PageSetup — Paper Size — Height

See Also

PAGE_WIDTH

PAGE_MARGIN_BOTTOM

Controls the bottom margin of the PDF page in the specified unit (inch or millimeter).

Syntax

```
PAGE_MARGIN_ BOTTOM '<numerical value>'
```

Example

PAGE_MARGIN_ BOTTOM '1'

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Margins — Bottom

See Also

PAGE_MARGIN_TOP

PAGE_MARGIN_LEFT

PAGE_MARGIN_RIGHT

PAGE_MARGIN_LEFT

Controls the left margin of the PDF page in the specified unit (inch or millimeter).

Syntax

```
PAGE_MARGIN_LEFT '<numerical value>'
```

Example

```
PAGE_MARGIN_LEFT '1.2'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Margins — Left

See Also

PAGE_MARGIN_RIGHT

PAGE_MARGIN_TOP

PAGE_MARGIN_BOTTOM

PAGE_MARGIN_RIGHT

Controls the right margin of the PDF page in the specified unit (inch or millimeter).

Syntax

PAGE_MARGIN_RIGHT '< numerical value>'

Example

PAGE_MARGIN_ RIGHT '1.2'

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Margins — Right

See Also

PAGE_MARGIN_LEFT

PAGE_MARGIN_TOP

PAGE_MARGIN_BOTTOM

PAGE_MARGIN_TOP

Controls the top margin of the PDF page in the specified unit (inch or millimeter).

Syntax

```
PAGE_MARGIN_ TOP '<numerical value>'
```

Example

```
PAGE_MARGIN_ TOP '1'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Margins — Top

See Also

PAGE_MARGIN_BOTTOM

PAGE_MARGIN_LEFT

PAGE_MARGIN_TOP

PAGE_NAME_CASE

Specifies whether the case of the `PAGE_NAME_PROP` property value will be preserved, upper cased, or lower cased while showing in the hierarchy viewer. By default, the case is preserved.

Syntax

```
PAGE_NAME_CASE 'UPCASE'|'LOWCASE'
```

Example

```
PAGE_NAME_PROP 'UPCASE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Design Navigation page — For page name property value — Preserve Case|All Lower Case|All Upper Case

PAGE_NAME_PROP

Sets the property name to be picked from the page border to obtain the page name. This directive sets the property on the page border for page name. By default, the property name is TITLE. This property name is configured in the `site.cpm` file.

Syntax

```
PAGE_NAME_PROP '<property_name>'
```

Example

```
PAGE_NAME_PROP 'TITLE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Design Navigation page — Property on page border for page name

PAGE_ORIENTATION

Defines whether the PDF page will be printed with portrait or landscape orientation,

Syntax

```
PAGE_ORIENTATION 'Portrait|Landscape'
```

Example

```
PAGE_ORIENTATION 'Landscape'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Orientation

PAGE_SCALE

Use this directive to define the page scaling (shrink or enlarge pages when you print.) You can automatically scale to fit the paper or you can specify the scaling factor, in percentage, by which the page is to be scaled (reduced or enlarged).

Syntax

```
PAGE_SCALE ‘percentage value|fit_to_page'
```

Example

```
PAGE_SCALE 'fit_to_page'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Scaling

PAGE_SINGLE_WIDTH

Use this directive to specify the line width of thin wires in the PDF output file.

Syntax

```
PAGE_SINGLE_WIDTH '<numerical value>'
```

Example

```
PAGE_SINGLE_WIDTH '1'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Line Width — Single

See Also

PAGE_DOUBLE_WIDTH

PAGE_UNIT

Defines the unit in which the PDF page size and margins are controlled.

Syntax

```
PAGE_UNIT 'Inch|Millimeter'
```

Example

```
PAGE_ UNIT 'Inch'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — PDF — PageSetup — Unit

PAGE_WIDTH

If you choose a custom page size to publish your PDF files, you can specify the width of the page using this directive.

Syntax

```
PAGE_WIDTH '<numerical value>'
```

Example

```
PAGE_ WIDTH '12'
```

Corresponding UI Option for Design Entry HDL

File —Publish PDF — Setup — PDF — PageSetup — Paper Size — Width

See Also

PAGE_HEIGHT

PAPER_ORIENTATION

Sets the orientation of the plot output. You can set it to 1 for Portrait or 2 for Landscape.

Syntax

```
PAPER_ORIENTATION '1'|'2'
```

where

1	Portrait
2	Landscape

Example

```
PAPER_ORIENTATION '1'
```

Corresponding UI Option for Allegro Design Entry HDL

Plotter Setup dialog box

PAPER_SIZE

Helps you plot in one paper size only and retain this setting over multiple Design Entry HDL sessions.

Syntax

```
PAPER_SIZE '<index>'
```

where

index	This 0-based index is the number of the paper size selected in the combo box in the Plot Setup dialog box.

Important

It is recommended that you set the paper size in the Plot Setup dialog box. If you need to change it manually, ensure that you enter the correct index to map to the correct paper size in the list displayed in the combo box. All plotters may not support all the available sizes

Note: For more information on the paper sizes, refer to the *Paper Sizes Supported by Design Entry HDL* topic in the *Plotting Your Design* section of *Allegro Design Entry HDL User Guide*.

Example

```
PAPER_SIZE '9'
```

Corresponding UI Option for Allegro Design Entry HDL

Print Setup dialog box — *Paper* — *Size*

PAPER_SOURCE

Helps you use one paper source throughout the site or for all your designs.

Syntax

```
PAPER_SOURCe '<index>'
```

where

index	This 0-based index is the number of the paper source selected in the combo box in the Plot Setup dialog box.

Note: All plotters may not support all the available paper sources.

Example

```
PAPER_SOURCE '4'
```

Corresponding UI Option for Allegro Design Entry HDL

Print Setup dialog box — *Paper* — *Source*

PART_TYPE_LENGTH

The PART_TYPE_LENGTH directive is used in Allegro System Capture and Packager-XL. This directive limits the length of the synthesized part names that are generated by Packager-XL when you use physical part tables or component definition properties.

Packager-XL shortens only physical part names synthesized by concatenating property values. The following part names are not shortened:

- Part names from the *chips* file.
- Part names in the PPT specified using exact part names (`~name`).
- Part names in the PPT synthesized by concatenating a string to the table name.
- Part names in the schematic specified using the COMP_NAME and COMP_NAME_SUFFIX properties.

However, if the part name length synthesized in any of the above mentioned ways, exceeds this maximum length, an error message is generated.

The PART_TYPE_LENGTH directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
PART_TYPE_LENGTH number;
```

where

number	is the maximum part type length. The number must be between 1 and 255.

Note: The default value for the PART_TYPE_LENGTH is 31.

Example

```
PART_TYPE_LENGTH 25;
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Physical Part Name Max Length

Corresponding UI Option for Design Entry HDL

None

PASS_PROPERTY

The PASS_PROPERTY directive specifies properties that are passed to the *pst* output files. You can pass any number of properties to the *pst* output files, and you can enter the PASS_PROPERTY directive as many times as needed in the project file.

If you specify any property with the PASS_PROPERTY directive, all other properties will automatically be filtered.

To omit specific properties, use the FILTER_PROPERTY directive.

If you specify a property with both, the PASS_PROPERTY, and the FILTER_PROPERTY directives, the property is passed.

Syntax

```
PASS_PROPERTY off|on|property[,property]...;
```

off	Prevents any properties from passing to the *pst* output files.
on	Packager-XL passes all properties to the *pst* output files.
property	Specifies the property name.

Note: The default value for the PASS_PROPERTY is on.

Examples

```
PASS_PROPERTY on;
```

PASTE_REPEATEDLY

Keeps the copied object attached to the cursor for repeated pasting after it is pasted on the canvas. By default, the directive is set to ON.

Syntax

```
PASTE_REPEATEDLY 'ON'|'OFF'
```

Examples

```
PASTE_REPEATEDLY 'ON'
PASTE_REPEATEDLY 'OFF'
```

PATHPROP_INVISIBLE

When you instantiate a component, the value of its PATH property is visible by default. Use this directive to hide the PATH property of components when they are instantiated. The visibility of the existing components is not affected by setting this directive.

Syntax

```
PATHPROP_INVISIBLE 'ON'|'OFF'
```

Example

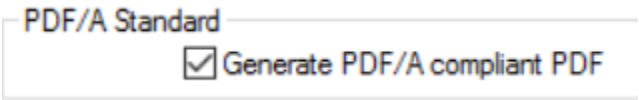
```
PATHPROP_INVISIBLE 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Set PATH Property Invisible

PDFA

This directive is set if you choose to generate a PDF/A compliant PDF.



Syntax

```
PDFA '0|1'
```

Example

```
PDFA '0'
```

Corresponding UI Option for Design Entry HDL

File —Publish PDF — Setup — PDF — Advanced — PDF/A Standard — Generate PDF/A Compliant PDF

PDFFont

Set this directive to fix text alignment issues when you print a schematic to PDF. When set, the height and width of a text object is stretched to align the object properly as it appears in the actual design. The range of values you can assign to this directive is between 0 to 4. Republishing a design after setting this directive fixes the text alignment issues.

Important

The recommended value for this directive is 0.87.

Syntax

```
PDFFont '<numerical value>'
```

Example

```
PDFFont '0.87'
```

Corresponding UI Option for Design Entry HDL

None

PHYSICAL_FOLDER

Using this directive, you can specify the location where you want the physical layout files to be generated.

Syntax

```
PHYSICAL_FOLDER '<directory>'
```

where

<directory>	Specify the path where you want the physical layout files to be generated.

Example

```
PHYSICAL_FOLDER 'C:\Designs'
```

Corresponding UI Option

None

PHYSICAL_PATH

The `PHYSICAL_PATH` directive lets you set the path of the input board (.brd) file for Export Physical. This directive is defined in the GLOBAL section of the project (.cpm) file.

If you have defined the `PHYSICAL_PATH` directive, then the following occurs:

- The Export Physical dialog box uses the path specified by the `PHYSICAL_PATH` directive to read the input board file.

  If you click the *Browse* button for selecting the input board file in the Export Physical dialog box, the resulting dialog box will display the board files in the path specified by the `PHYSICAL_PATH` directive. The value of the `PHYSICAL_PATH` directive in the .cpm file is also updated to the new path that you select.

If the `PHYSICAL_PATH` directive is not specified, then the `physical` directory under the root design is used to read or store the board files.

To set the `PHYSICAL_PATH` directive, perform the following steps:

1. Choose the *Tools* tabbed page in Project Setup.
2. The default entry in the *Physical* field is null, which means the physical directory in the root design. You can browse and choose a directory from where the input board file has to be opened.

Note: Cadence recommends that the path that you specify in the `PHYSICAL_PATH` directive be the same as that specified in the `VIEW_PCB` directive.

PINALIAS_<PIN_TYPE>

Set this directive to map the copied pin type to the System Capture supported pin types when pasting pin data copied from a CSV file to *Table* view.

Syntax

```
PINALIAS_<PIN_TYPE> '<copied_pin_type>'
```

where

`PIN_TYPE` Pin types supported by System Capture:

- INPUT
- TS
- TSBIDIR
- OCBIDIR
- OEBIDIR
- OUTPUT
- OC
- OE
- BIDIR
- ANALOG
- GROUND
- POWER
- NC

`copied_pin_type` External pin type (copied from CSV) which will be mapped to the System Capture pin types. Copied pin types are not case-sensitive.

Following table describes the mapping between the external pin types and System Capture pin types:

If Specified	External pin type will be mapped to
PinShape_INPUT	Input
PinShape_TS	Ts
PinShape_TSBIDIR	Ts_inout
PinShape_OCBIDIR	Oc_inout
PinShape_OEBIDIR	Oe_inout
PinShape_OUTPUT	Output
PinShape_OC	Oc
PinShape_OE	Oe
PinShape_BIDIR	Inout
PinShape_ANALOG	Analog
PinShape_GROUND	Ground
PinShape_POWER	Power
PinShape_NC	Nc

Example

```
PINALIAS_BIDIR 'Input-Output'
```

If the above value is specified, all the pins copied from the CSV file, which have pin type as `Input-Output` will be mapped to *Inout* pins in System Capture, when pasted in the *Table* view.

PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH

This directive applies to system-level designs created in Allegro System capture. You can quickly identify the nets that are perfect matches or partial matches in the *Port/Pin Assignment* dialog box by setting this directive to `True`. The `PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT` directive controls how many characters are compared.

Syntax

```
PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH 'YES'|'NO'
```

Example

```
PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH 'YES'
```

Corresponding UI Option for Allegro System Capture

None

Related Commands

PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT

PIN_ASSIGNMENT_DIALOG_SHOW_BLOCK_NET_NAME

Controls the display of block net names in the Port/Pin Assignment dialog box.

Syntax

```
PIN_ASSIGNMENT_DIALOG_SHOW_BLOCK_NET_NAME 'YES'|'NO'
```

where,

`YES`	Block net names are displayed in the *Port/Pin Assignment* dialog box.
`NO`	Block net names are not displayed in the *Port/Pin Assignment* dialog box. This is the default value

Example

```
PIN_ASSIGNMENT_DIALOG_SHOW_BLOCK_NET_NAME 'NO'
```

PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER

Controls the display of pin numbers in the *Port/Pin Assignment* dialog box.

Syntax

```
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 'YES'|'NO'
```

where

`YES`	Pin numbers are displayed in the *Port/Pin Assignment* dialog box. This is the default value.
`NO`	Pin numbers are not displayed in the *Port/Pin Assignment* dialog box.

Example

```
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 'YES'
```

PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER

Controls the display of pin numbers in the *Port/Pin Assignment* dialog box.

Syntax

```
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 'YES'|'NO'
```

where

`YES`	Pin numbers are displayed in the *Port/Pin Assignment* dialog box. This is the default value.
`NO`	Pin numbers are not displayed in the *Port/Pin Assignment* dialog box.

Example

```
PIN_ASSIGNMENT_DIALOG_SHOW_PIN_NUMBER 'YES'
```

PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT

This directive applies to system-level designs created in Allegro System Capture. You can quickly identify the nets that are perfect matches or partial matches in the *Port/Pin Assignment* dialog box by setting the `PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH` directive to `True`. This directive controls how many characters are compared.

Here is how the colors are assigned:

- If all characters match, the net is highlighted in green
- If ‘N or more’ continuous characters match but not all, the net is highlighted in yellow.
- If less than ‘N’ characters match, the net is highlighted in red.

Syntax

```
PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT <number>
```

Example

```
PIN_ASSIGNMENT_DIALOG_MINIMUM_CHARACTER_MATCH_COUNT 3
```

Corresponding UI Option for Allegro System Capture

None

Related Commands

PIN_ASSIGNMENT_DIALOG_SHOW_COLORED_NET_MISMATCH

PINNUMBER_ROTATION

Automatically rotates pin numbers that are attached to vertical pins.

Syntax

```
PINNUMBER_ROTATION 'ON'|'OFF'
```

Example

```
PINNUMBER_ROTATION 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Properties — Rotate Vertical Pin Numbers During Backannotation

PINNUMBER_SIZE

Adjusts the size of the pin number displayed on the schematic to be larger or smaller. The unit is in inches. The pin number size is not related to Text Size you specify in this dialog box.

Syntax

```
PINNUMBER_SIZE '<value>'
```

where

value	a positive number

Note: All plotters may not support all the available paper sources.

Example

```
PINNUMBER_SIZE '0.805'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Properties — Pin Number Size

PINPROP_VISIBILITY

Controls the visibility of symbol pin properties when the symbol/component is instantiated on the schematic.

Syntax

```
PINPROP_VISIBILITY 'OFF'|'ON'
```

where

ON	Defined by the component makes pin properties visible or not depending on how property visibility is defined on the symbol.
OFF	Does not display the symbol pin properties

Example

```
PINPROP_VISIBILITY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Pin Property Visibility

PinType

The `PinType` directive specifies the default input and output load and pin location associated with each pin type supported in the Cadence Allegro flow.

For a list of pin types supported in the Cadence Allegro flow, see the *Pin Types* appendix in *Part Developer User Guide*.

Syntax

```
PinType_<pin_type>
     '<input_load_low>,<input_load_high>,<output_load_low>,<output_load_high>,<sy
     mbol_pin_location>'
```

Note: You can specify `x` in the place of a property to indicate that the property is not associated.

Example

```
PinType_INPUT '-0.01,0.01,x,x,Left'
```

Corresponding UI Option for Part Developer

PinType, *Input Load*, and *Output Load* columns in the *Package Pin Properties* grid on *Tools – Setup – Package Pins*

Pin Type and *Location* columns in the *Pin Location* grid on *Tools – Setup –Symbol Pins – Properties*

PLACEMENT_WITHIN_GROUP_USING_ORDER_IN_DESIGN

Use this directive to specify that in the generated document schematic, components in the same schematic group (specified using SCHEMATIC_GROUP property) are placed in the order in which they are added to the design.

Syntax

```
placement_within_group_using_order_in_design <0 or 1>
```

Example

```
placement_within_group_using_order_in_design '1'
```

Corresponding UI Option for System Connectivity Manager

None

PLOT_COLOR

Directs Design Entry HDL to plot the drawing in color if you are using a color plotter, and in gray scales if you are using a black and white printer.

Syntax

```
PLOT_COLOR 'OFF'|'ON'
```

Example

```
PLOT_COLOR 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Plot Method — Color

PLOT_DOUBLE_WIDTH

Specifies the width of thick wires and buses.

Syntax

```
PLOT_DOUBLE_WIDTH '<value>'
```

Example

```
PLOT_DOUBLE_WIDTH '10'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Scaling — Double Line Width

See Also

- PLOT_COLOR
- PLOT_FIT_TO_PAGE
- PLOT_FONT
- PLOT_SCALE
- PLOT_SCREEN
- PLOT_SINGLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_FILE_NAME

This directive is used in conjunction with the `PLOT_TO_FILE` directive to change the default file name for the plot output. You can also specify the full path of the file if you wish to direct output to a directory other than the project directory. By default the file name is output.ps and it is created in the project directory.

Note: This directive is for the `plot` console command. It is not read or written by plot Setup of Design Entry HDL.

Syntax

```
PLOT_FILE_NAME '<file_name>'
```

Example

```
PLOT_FILE_NAME 'output.ps'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PLOT_TO_FILE

PLOT_FIT_TO_PAGE

Directs Design Entry HDL to adjust the plot according to page size.

Syntax

```
PLOT_FIT_TO_PAGE 'OFF'|'ON'
```

Example

```
PLOT_FIT_TO_PAGE 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Scaling — Fit To Page

See Also

- PLOT_COLOR
- PLOT_FONT
- PLOT_SCALE
- PLOT_SCREEN
- PLOT_SINGLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_FONT

Specifies the font to be used when the schematic is plotted. You can specify Arial, Helvetica, Verdana, Trebuchet MS, or Default.

Syntax

```
PLOT_FONT '<value>'
```

Example

```
PLOT_FONT 'COURIER'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Plot Font

See Also

- PLOT_COLOR
- PLOT_SCALE
- PLOT_SCREEN
- PLOT_SINGLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_SCALE

Specifies the percentage by which to increase or decrease the plot size.

Syntax

```
PLOT_SCALE '<value_in_percentage>'
```

Example

```
PLOT_SCALE '100'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Scaling — Adjust To

See Also

- PLOT_COLOR
- PLOT_FONT
- PLOT_SCREEN
- PLOT_SINGLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_SCREEN

Directs Design Entry HDL to plot the portion of the schematic that is displayed on the screen.

Syntax

```
PLOT_SCREEN 'OFF'|'ON'
```

Example

```
PLOT_SCREEN 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Plot Method — Screen

See Also

- PLOT_COLOR
- PLOT_SCALE
- PLOT_FONT
- PLOT_SINGLE_WIDTH
- PLOT_DOUBLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_SINGLE_WIDTH

Specifies the width of thin wires and buses. You can also control the text width by this field. On some plotters, if the plot output is very thin and does not show the text clearly, this width can be increased making the whole design, along with the text, thicker.

Syntax

```
PLOT_SINGLE_WIDTH '<value>'
```

Example

```
PLOT_SINGLE_WIDTH '1'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Plotting page — Scaling — Single Line Width

See Also

- PLOT_COLOR
- PLOT_FIT_TO_PAGE
- PLOT_FONT
- PLOT_SCALE
- PLOT_SCREEN
- PLOT_DOUBLE_WIDTH
- PLOT_THIN_WIDTH
- PLOT_THICK_WIDTH

PLOT_THICK_WIDTH

Sets the width of thick lines in plots.

Syntax

```
PLOT_THICK_WIDTH '<value>'
```

Example

```
PLOT_THICK_WIDTH '10'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PLOT_THIN_WIDTH

PLOT_THIN_WIDTH

Sets the width of thin lines in plots.

Syntax

```
PLOT_THIN_WIDTH '<value>'
```

Example

```
PLOT_THIN_WIDTH '5'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PLOT_THICK_WIDTH

PLOT_TO_FILE

This directive tells the `plot` command to direct its plot output to a file. The default file name is `output.ps`.

Note: This directive is for the `plot` console command. It is not read or written by plot Setup of Design Entry HDL.

Syntax

```
PLOT_TO_FILE 'ON'|'OFF'
```

Example

```
PLOT_TO_FILE 'ON'
```

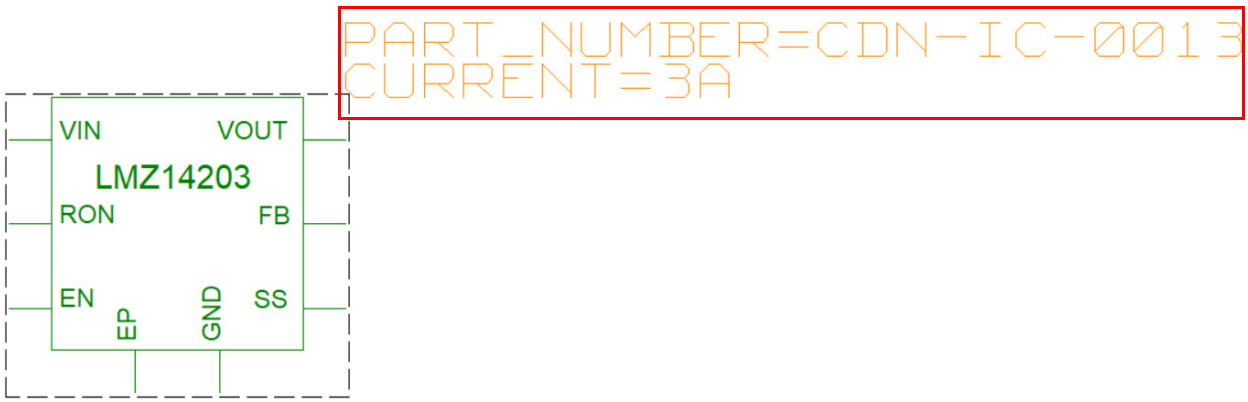
Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PLOT_FILE_NAME

- Stacks properties downwards.



See Also

- PROP_PLACEMENT_DEFAULT

- PROP_JUSTIFICATION
- PROP_STACKING

POWER_SYMBOLS

The POWER_SYMBOLS directive is used in Allegro System Capture and Schgen. This directive is used to add Power and Ground symbols to the *Special Symbols* bucket.

Syntax

```
POWER_SYMBOLS '<power:library:cell:view>'
```

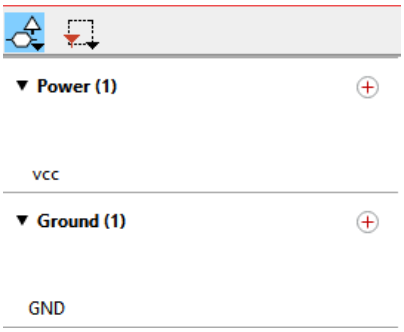
where

power	is `power` and the voltage of the symbol. For example, if you want to add the a power symbol of +5V, enter: `power!+5V`.
library	is the library name in which you want the symbol to be added.
cell	is the cell name of the library in which you want the symbol to be added. This name appears in the *Special Symbols* bucket.
view	is the view name in which you want the symbol to be added.

Example

```
POWER_SYMBOLS 'GND!0V:standard:GND:sym_1' 'VCC!+5V:standard:vcc:sym_1'
```

This adds two symbols to the *Special Symbols* bucket as shown in the following image:



Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for System Connectivity Manager

None

POWERPROP_VIS

Controls visibility when assigning power pins

Syntax

```
POWERPROP_VIS 'INVISIBLE'|'NAME'|'VALUE'|'BOTH'
```

Example

```
POWERPROP_VIS 'VALUE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Power Property Visibility

See Also

PROP_VISIBILITY

Ppl_Only

Defines whether Part Information Manager should allow components only from preferred parts lists in the design. This directive is always used in conjunction with the `project_ppl` directive.

If set to TRUE, Part Information Manager does not allow the addition of non-PPL parts to a design. If you try and add a part from lists other than the preferred parts lists, Part Information Manager displays an error. This is called the PPL Plus mode.

If set to FALSE, both PPL compliant parts, and other parts can be used in the design. This is called the PPL Only mode.

Syntax

```
ppl_only 'TRUE'|'FALSE'
```

Example

```
ppl_only 'FALSE'
```

Corresponding UI Option

Allegro EDM Project Wizard — Get Project Information — Select parts from PPLs only

See Also

Project_Ppl

PPT

The PPT directive lets you list paths to the files and directories that contain physical part tables (PPTs). If the path you specify is a directory, then Packager-XL loads all ptf files in that directory.

You can specify exceptions to this option with either the EXCLUDE_PPT or INCLUDE_PPT directives. Packager-XL first loads all files specified with the PPT directive and then loads the cell-level PPTs. For more information on cell-level PPTs, see USE_LIBRARY_PPT.

You can enter the PPT directive as many times as needed in the project file.

All ptf files located at the cell level must have a `.ptf` extension.

- You must specify the PPT directive in the Part Table section of the Project Setup dialog.
- You can use the PPT directive only for file names.

Syntax

```
PPT name[pathname, pathname]...;
```

pathname	The name of a ptf file or directory. If the path name is a directory, Packager-XL loads all files in the directory that have `.ptf` as the file extension.

The default value for the PPT directive is none.

Example

The ptf file will have a `.ptf` extension.

- `PPT project.ptf, /usr/library/ptfdir;`

PPT_BROWSER

Automatically opens the Physical Part Filter dialog box when you open Part Information Manager to add or replace a component.

Syntax

```
PPT_BROWSER 'ON'|'OFF'
```

Example

```
PPT_BROWSER 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Show PPT Browser

PPT_OPTIONSET_PATH

Specify the path and the PPT Option Set file that you want Design Entry HDL and System Capture to use by default. This file stores the default display settings for physical properties in the schematic and in the Physical Part Filter dialog box.

Syntax

```
PPT_OPTIONSET_PATH '<PPT_Option_Set_file_path and file name>'
```

where

PPT_Option_Set_file_path and file name	is the path to the directory with the PPT Option Set file (`.dat`) and the `.dat` file name.

Example

```
PPT_OPTIONSET_PATH 'D:\project\ppt_optionset.dat'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Paths page — Input Paths — PPT Option Set

Corresponding UI Option for Allegro System Capture

None

See Also

- CATPATH
- INPUT_SCRIPT

PRESELECT_FLAG

Activates the pre-select mode for Design Entry HDL menus. If this directive is not set, you cannot set the WINDOWSMODE_FLAG directive to switch to Window mode.

Syntax

```
PRESELECT_FLAG 'ON'|'OFF'
```

Example

```
PRESELECT_FLAG 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Enable Pre-Select Mode

See Also

- WINDOWSMODE_FLAG
- PRESELECT_FLAG ALLOW_USER_CPM

PRESELECT_FLAG ALLOW_USER_CPM

Allows the `PRESELECT_FLAG` directive to be defined in the `user.cpm` file.

Syntax

```
PRESELECT_FLAG ALLOW_USER_CPM 'ON'|'OFF'
```

Example

```
PRESELECT_FLAG ALLOW_USER_CPM 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PRESELECT_FLAG

WINDOWSMODE_FLAG ALLOW_USER_CPM

PRESERVE_BYPASS

Use this directive to specify if the bypass capacitors should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_bypass 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_bypass 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_CONN

Use this directive to specify if connectivity should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_conn 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_conn 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace — Preserve Options — Preserve Connectivity

See Also

- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_DAT

System Capture generates the netlisting files inside a compressed file that can be easily shared with designers working on a same layout.

Set this directive to ON if you want to generate these files in the extracted form.

Syntax

```
PRESERVE_DAT 'ON'|'OFF'
```

The default value is ON.

Example

`PRESERVE_DAT 'ON'` PRESERVE_PIN_TEXT_ROTATION

Rotating symbols on the schematic canvas does not affect the position of their $PN properties when this directive is ON.

Syntax

```
PRESERVE_PIN_TEXT_ROTATION 'ON'|'OFF'
```

Example

```
PRESERVE_PIN_TEXT_ROTATION 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

PRESERVE_PINPAIR

Use this directive to specify if pin-pairs should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_pinpair 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_pinpair 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_PWRGRP

Use this directive to specify if power groups should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_pwrgrp 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_pwrgrp 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_REFDES

Use this directive to specify if reference designators should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_refdes 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_refdes 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_TERMINATION
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_TERMINATION

Use this directive to specify if terminations should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_termination 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_termination 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_USERPROP
- SHOW_ANALYZE_DIALOG

PRESERVE_USERPROP

Use this directive to specify if user-properties should be preserved on the new component after a Component Replace operation.

Syntax

```
preserve_userprop 'TRUE'|'FALSE'|'0'|'1'
```

Example

```
preserve_userprop 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project— Settings —Component Replace

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- SHOW_ANALYZE_DIALOG

PRESERVE_ZOOM_INFO

Preserves zoom info for a project opened in concept.

Note: In pre-15.7 releases, pages were displayed as "Fit To Page" when switching from page to page regardless of how the zoom was set on previously open pages. Release 15.7 onwards, if you zoom in on page 1 and then switch to the next page, page 2 is zoom fit to the page. If you go back to page 1, it is still zoomed in to the way it was.

Syntax

```
PRESERVE_ZOOM_INFO 'ON'|'OFF'
```

Example

```
PRESERVE_ZOOM_INFO 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

PRINT_EXCLUSION

Allows users to control the printing of specific blocks in a System Capture design. By default, all the blocks in a design are printed, for example when creating a PDF. When this directive is set in the design or `site.cpm`, an additional menu option gets enabled when you right-click a block in the Project explorer window.

Syntax

```
PRINT_EXCLUSION 'true' | 'false'
```

Example

```
PRINT_EXCLUSION 'true'
```

Corresponding UI Option for Allegro System Capture

None

PRINTLAYER

Incorporates the printing feature in the viewable PDF. This defines the layers that will be available in the printed version of the PDF that you generate for the schematic. You can take printouts of published PDF documents exactly as they appear on the viewable PDF document.

Syntax

```
PRINTLAYER '<decimal_value>'
```

Where *decimal value* determines which all layers are to be printed in the printable PDF document.

Example

```
PRINTLAYER '511'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Layers — Print Layer

See Also

EXPORT

VISIBLE

PRINTLAYERENABLE

PRINTLAYERENABLE

Works in conjunction with the Print Layer. Includes a printable PDF version in the generated PDF.

Syntax

```
PRINTLAYERENABLE '0'|'1'
```

Example

```
PRINTLAYERENABLE '0'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Insert Print Layer check box

See Also

PRINTLAYER

PROCESS_PIN_SHORT_PROP

The `PROCESS_PIN_SHORT_PROP`, directive directs Packager-XL to acknowledge the PIN_SHORT property value and create a NET_SHORT property with its value containing the physical net names connected to the logical pin names.

Syntax

```
PROCESS_PIN_SHORT_PROP 'ON'|'OFF'
```

Example

```
PROCESS_PIN_SHORT_PROP 'ON'
```

Project_Ppl

Defines the preferred parts lists (PPL) to be used in a design project. You can define multiple PPLs for a project. Only components from these defined PPLs can be added to a project. PPLs listed in this directive are displayed by default in the Relations tab in Part Information Manager.

If you defined a color for a PPL when you created the PPL in Database Administrator, then the color of the PPL to which the component belongs is also displayed in Part Information Manager. If a component belongs to multiple PPLs, all the colors will be displayed in Part Information Manager.

Part Information Manager displays a warning when you try and add a part that is not from a PPL.

Tip

If you want Part Information Manager to consider the specified PPLs in an order of priority, you could create a hierarchical PPL using Allegro EDM Database Administrator and use that instead of specifying multiple PPLs in this directive. The hierarchical PPL can include all the PPLs you want to specify.

Syntax

project_ppl '<PPL name>' '<PPL name>' '<PPL name>'

Example

project_ppl 'ppl_level1' 'cadence_approved'

Corresponding UI Option

Allegro EDM Project Wizard — Get Project Information — Preferred Parts List Name

See Also

Ppl_Only

PROP_COLOR

Use this directive to specify the color for the properties and their values in the generated document schematic in Design Entry HDL and Schgen.

Syntax

```
PROP_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
PROP_COLOR 'ORANGE'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Property

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Colors — Property Color.

See Also

- BACKGROUND_COLOR
- DOT_COLOR
- NOTE_COLOR
- HIGHLIGHT_COLOR
- ARC_COLOR
- SYMBOL_COLOR
- WIRE_COLOR

PROP_JUSTIFICATION

When adding, modifying, or replacing components, use this directive to justify the annotated property text on the canvas.

Note: This directive only works if the values of the PROP_PLACEMENT_DEFAULT, PROP_OFFSET, PROP_STACKING directives are also specified.

Syntax

```
PROP_JUSTIFICATION_<PROP_PLACEMENT_DEFAULT value> 'JUST_LEFT'|'JUST_CENTER'|'JUST_RIGHT'
```

where

JUST_LEFT	Aligns properties to the left side of the position specified in the PROP_PLACEMENT_DEFAULT directive.
JUST_CENTER	Aligns properties to the center of the position specified in the PROP_PLACEMENT_DEFAULT directive.
JUST_RIGHT	Aligns properties to the right side of the position specified in the PROP_PLACEMENT_DEFAULT directive.

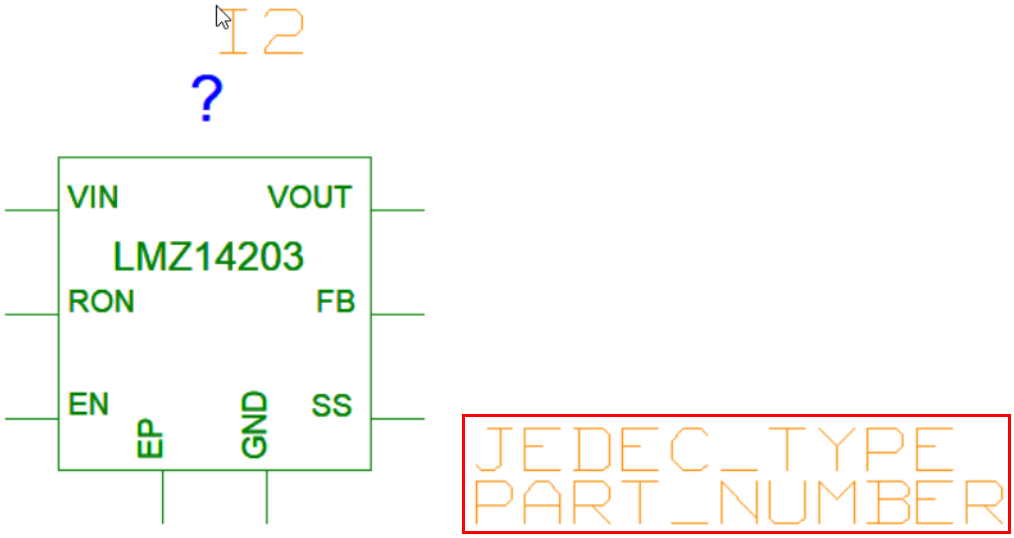
Example

```
PROP_JUSTIFICATION_LR 'JUST_LEFT'
PROP_PLACEMENT_DEFAULT 'LR'
PROP_OFFSET_LR '(50,50)'
PROP_STACKING_LR 'STACK_DOWN'
```

In this case, the position of the properties changes to *Lower Right* of the component and:

- Aligns properties to the left side.
- On the X-axis and Y-axis, positions properties at a distance of 50 from the component.

- Stacks properties upwards.



Corresponding UI Option for Allegro Design Entry HDL

None

See Also

- PROP_PLACEMENT_DEFAULT
- PROP_OFFSET
- PROP_STACKING

PROP_OFFSET

When adding, modifying, or replacing components, use this directive to position annotated properties on the canvas at a specific distance from the component.

Note: This directive will only work if the values of the PROP_PLACEMENT_DEFAULT, PROP_JUSTIFICATION, PROP_STACKING directives are also specified.

Syntax

```
PROP_OFFSET_<PROP_PLACEMENT_DEFAULT value> '(X,Y)'
```

where

X	Indicates the distance at which you want to position the property from the component on the X-axis.
Y	Indicates the distance at which you want to position the property from the component on the Y-axis.

Example

```
PROP_JUSTIFICATION_UR 'JUST_LEFT'
PROP_PLACEMENT_DEFAULT 'UR'
PROP_OFFSET_UR '(0,0)'
PROP_STACKING_UR 'STACK_UP'
```

In this case, the position of the properties changes to *Upper Right* of the component and:

- Aligns properties to the left side.
- On the X-axis and Y-axis, positions properties at a distance of `0` from the component.

Corresponding UI Option for Allegro Design Entry HDL

None

PROP_PLACEMENT_DEFAULT

When adding, modifying, or replacing components, use this directive to control the positioning of annotated properties on the canvas.

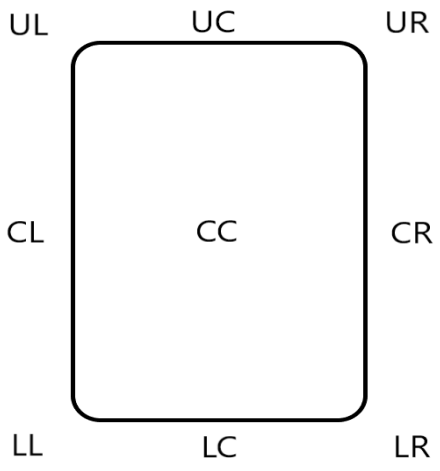
Note: This directive only works if the values of the PROP_JUSTIFICATION, PROP_OFFSET, PROP_STACKING directives are also specified.

Syntax

```
PROP_PLACEMENT_DEFAULT 'UL'|'UC'|'UR'|'CL'|'CC'|'CR'|'LL'|'LC'|'LR'
```

where the abbreviations indicate the position of properties relative to the component:

UL	*Upper Left*
UC	*Upper Center*
UR	*Upper Right*
CL	*Center Left*
CC	*Center*
CR	*Center Right*
LL	*Lower Left*
LC	*Lower Center*
LR	*Lower Right*



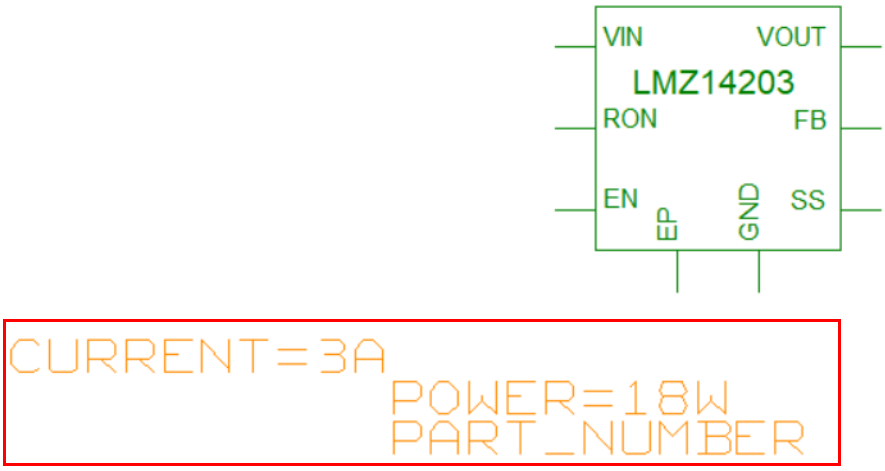
Example

```
PROP_PLACEMENT_DEFAULT 'LL'
PROP_JUSTIFICATION_LL 'JUST_RIGHT'
PROP_OFFSET_LL '(-200,-100)'
PROP_STACKING_LL 'STACK_DOWN'
```

In this case, the position of the properties changes to *Lower Left* of the component and:

- Aligns properties to the left side.

- Positions properties at a distance of `-250` on the X-axis and `-50` on the Y-axis.
- Stacks properties downwards.



Corresponding UI Option for Allegro Design Entry HDL

None

See Also

- PROP_JUSTIFICATION
- PROP_OFFSET
- PROP_STACKING

PROP_STACKING

When adding, modifying, or replacing components, use this directive to set the vertical stacking of annotated properties on the canvas.

Note: This directive only works if the values of the PROP_PLACEMENT_DEFAULT, PROP_JUSTIFICATION, PROP_OFFSET directives are also specified.

Syntax

```
PROP_STACKING_<PROP_PLACEMENT_DEFAULT value> 'STACK_UP'|'STACK_DOWN'|'UP/DOWN'
```

where

`STACK_UP`	Stacks properties in the upward direction of the position specified in the PROP_PLACEMENT_DEFAULT directive.
`STACK_DOWN`	Stacks properties in the downward direction of the position specified in the PROP_PLACEMENT_DEFAULT directive.
`UP/DOWN`	Stacks properties in the upward or downward direction of the position specified in the PROP_PLACEMENT_DEFAULT directive.

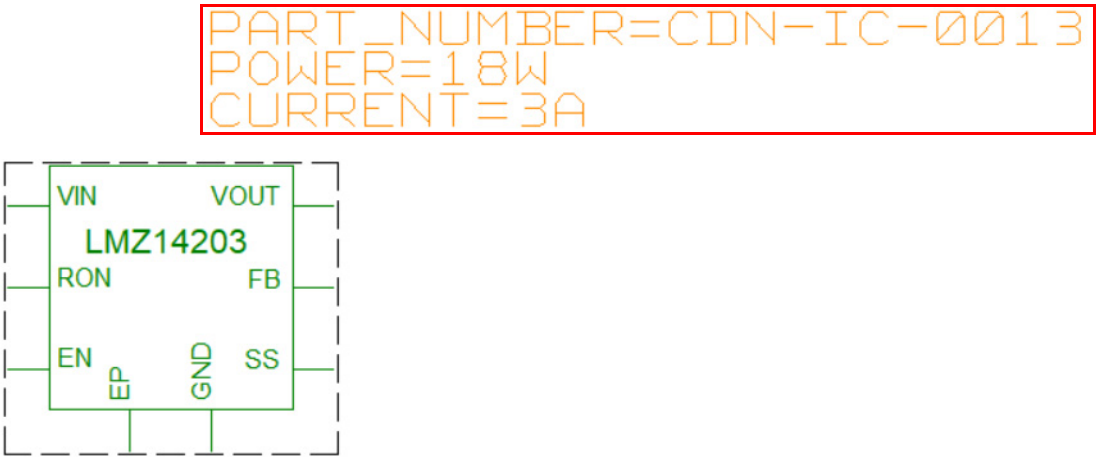
Example

```
PROP_PLACEMENT_DEFAULT 'UC'
PROP_JUSTIFICATION_UC 'JUST_LEFT'
PROP_OFFSET_UC '(20,20)'
PROP_STACKING_UC 'STACK_UP'
```

In this case, the position of the properties changes to *Upper Center* of the component and:

- Aligns properties to the left side.
- Positions properties at a distance from the component - `0` on X-axis and `250` on Y-axis.

- Stacks properties downwards.



Corresponding UI Option for Allegro Design Entry HDL

None

See Also

- PROP_PLACEMENT_DEFAULT
- PROP_JUSTIFICATION
- PROP_OFFSET

PROP_VISIBILITY

Controls the way properties are displayed.

Syntax

```
PROP_VISIBILITY 'INVISIBLE'|'NAME'|'VALUE'|'BOTH'
```

Example

```
PROP_VISIBILITY 'BOTH'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Properties — Visibility

See Also

POWERPROP_VIS

PTF_MISMATCH_EXCLUDE_INJ_PROP

This directive controls which injected properties should not be checked for mismatched values. Part Manager and Packager-XL run the part table file (PTF) mismatch check and report warnings and errors.

The PTF_MISMATCH_EXCLUDE_INJ_PROP directive supports the following values:

- ALL: all injected properties will be excluded from the mismatch check.
- NONE: injected properties will not be excluded from the mismatch check.
- Space-separated list of property names: the specified properties will be excluded from the check.

Syntax

```
PTF_MISMATCH_EXCLUDE_INJ_PROP 'ALL'|'NONE'|'<space-separated list of
property names>'
```

Example

```
PTF_MISMATCH_EXCLUDE_INJ_PROP PART_NUMBER LOCATION PART_NAME TOL
PIN_DELAY
```

See also

STOP_PST_GEN_ON_PTF_MISMATCH

PTF_VIEW

The PTF_VIEW directive specifies the Part Table View directory name.

You specify the PTF_VIEW directive in the Global section of the Project Setup form.

Syntax

```
PTF_VIEW <name>;
```

where `name` is the Part Table View name.

Example

- `PTF_View part_table;`

PULLDOWN_MAX

Defines the maximum value of resistance of a pulldown resistor.

When the `High Pulldown Resistance Value` audit rule under the *Protocol Checks* category is run, pulldown resistors with values higher than the configured minimum value are flagged.

Syntax

```
PULLDOWN_MAX '<high pulldown resistance value>'
```

Example

```
PULLDOWN_MAX '10000'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Maximum Pulldown*

Also See

- PULLUP_MIN
- PULLDOWN_MIN
- PULLUP_MAX

PULLDOWN_MIN

Defines the minimum value of resistance of a pulldown resistor.

When the `Low Pulldown Resistance Value` audit rule under the *Protocol Checks* category is run, pulldown resistors with values lower than the defined minimum value are flagged.

Syntax

```
PULLDOWN_MIN '<minimum pulldown resistance value>'
```

Example

```
PULLDOWN_MIN '1000'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Minimum Pulldown*

Also See

- PULLUP_MIN
- PULLDOWN_MAX
- PULLUP_MAX

PULLUP_MAX

Defines the maximum value of resistance of a pullup resistor.

When the `Low Pullup Resistance Value` audit rule under the *Protocol Checks* category is run, the pullup resistors with values higher than the configured minimum value are flagged.

The maximum value of the pullup resistor cannot be lower than or the same as its minimum value.

Syntax

```
PULLUP_MAX '<maximum pullup resistance value>'
```

Example

```
PULLUP_MAX '10000'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Maximum Pullup*

Also See

- PULLUP_MIN
- PULLDOWN_MIN
- PULLDOWN_MAX

PULLUP_MIN

Defines the minimum value of resistance of a pullup resistor.

When the `Low Pullup Resistance Value` audit rule under the *Protocol Checks* category is run, pullup resistors with values lower than the configured minimum value are flagged.

The maximum value of the pullup resistor cannot be lower than or the same as its minimum value.

Syntax

```
PULLUP_MIN '<minimum pullup resistance value>'
```

Example

```
PULLUP_MIN '1000'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *Minimum Pullup*

Also See

- PULLUP_MAX
- PULLDOWN_MIN
- PULLDOWN_MAX

PWR_PIN_NAME

Defines the power pin names in a design when the following audit rules are run:

- `Grounded IC Output Pins`
- `IC Output Pins Without Receiver`
- `Unconnected IC Power/Ground Pins`
- `IC Input Pins Without Driver`
- `Nets connected to SDA pins missing a pull-up`
- `Nets connected to SCL pins missing a pull-up`
- `Open collector output pin missing a pull-up`
- `All IC Input Pins Pulled Up`
- `All IC Input Pins Pulled Down`
- `Low Pullup Resistance Value`
- `High Pulldown Resistance Value`
- `High Pullup Resistance Value`
- `Low Pulldown Resistance Value`
- `Bypass capacitor voltage exceeds rated voltage`
- `Same group power pins connected to nets with different voltage values`
- `Power Nets Without Voltage`
- `Power Nets With 0V`
- `Ground Nets Without Voltage`
- `Ground Nets With Non-Zero Voltage`
- `Missing Bypass Capacitors`
- `Nets with pull-up and pull-down resistors`

Design Integrity identifies pins as power pins based on the following conditions:

- If the pin name matches the patterns defined in this directive

- If the pin type is defined as power in `chips.prt`

Syntax

```
PWR_PIN_NAME '<pin pattern>'
```

The following pin patters are supported:

```
VTT, VBAT, VBB, VDDADC, VDD3V3, DVDD3V3, VDD2V5, DVDD2V5, VDD1V2,
DVDD1V2, VDDIO
```

Example

```
PWR_PIN_NAME 'VDD' 'VCC'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Power or *Ground* tab – *Power Pins*

17

R Directives

This chapter lists the CPM directives that start with `R` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

RAS_PIN_PATTERN

Defines the patterns used for identifying pins as Row Address Strobe (RAS) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `RAS and CAS pins of ICs incorrectly connected to non-RAS or non-CAS pins`
- `RAS and CAS pins of ICs not connected to the same set of ICs or connectors`
- `RAS and CAS pins of IC incorrectly connected`
- `Unconnected RAS and CAS pins of ICs`

Syntax

```
RAS_PIN_PATTERN '<pin pattern>'
```

Example

```
RAS_PIN_PATTERN 'RAS'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *RAS Pin Patterns*

RAT_ON_REPLACE

Sets whether the connections to the pins should be changed to rats on replacing a component.

Syntax

```
RAT_ON_REPLACE 'YES'|'NO'
```

The default value is NO.

Example

```
RAT_ON_REPLACE 'NO'
```

REF_DES_LENGTH

The REF_DES_LENGTH directive is used in Allegro System Capture and Packager-XL.This directive controls the maximum length of physical reference designators generated by Packager-XL.

The REF_DES_LENGTH directive does not affect user-assigned LOCATION properties. However, if the LOCATION property value (user assigned or synthesized from the REF_DES_PATTERN directive) exceeds the maximum length of physical reference designators, an error message is generated.

The REF_DES_LENGTH directive is specified in the START_PKGRXL...END_PKGRXL and START_CANVAS...END_CANVAS sections. Packager-XL reads the directive value from the START_PKGRXL...END_PKGRXL section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the START_CANVAS...END_CANVAS section when launched from Allegro System Capture.

Syntax

```
REF_DES_LENGTH number;
```

where

number	is the maximum number of characters in the reference designator.

The default value for the REF_DES_LENGTH directive is 31 characters.

Example

```
REF_DES_LENGTH 12;
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Reference Designator Max Length

Corresponding UI Option for Design Entry HDL

None

REF_DES_PATTERN

The REF_DES_PATTERN directive is used in Allegro System Capture and Packager-XL. This directive specifies the format of reference designators (that is, location properties) assigned to the physical parts in a design and this directive also applies to all parts in your design.

If you want to specify a pattern for a particular part or instance, use the REF_DES_PATTERN property instead. The REF_DES_PATTERN directive is only applied to unpackaged parts in the design.

To change the existing reference designators, do one of the following:

- Use the REPACKAGE directive. For more information on the REPACKAGE directive, see REPACKAGE.
- Manually edit the LOCATION properties in the schematic.
- You can specify REF_DES_PATTERN as a property or a directive. For more information, refer to Allegro Platform Properties Reference Guide. Currently, in Allegro System Capture, this directive is supported only for flat designs.

The REF_DES_PATTERN directive is specified in the START_PKGRXL...END_PKGRXL and START_CANVAS...END_CANVAS sections. Packager-XL reads the directive value from the START_PKGRXL...END_PKGRXL section when launched from Design Entry HDL or System Connectivity Manager and rads the directive value from the START_CANVAS...END_CANVAS section when launched from Allegro System Capture.

Caution

REF_DES_PATTERN overrides the default naming convention used by Packager-XL.

Syntax

```
REF_DES_PATTERN pattern
```

pattern	is an alphanumeric character string

The default is:

```
REF_DES_PATTERN ($PHYS_DES_PREFIX)[0-9](1);
```

The pattern can be a combination of the following:

- Ordinary characters
- Value to be incremented

  This value is specified within square brackets [ ] and you can use this value more than once. The letters 0-9 indicate a numeric value, while the letters A-Z indicate an alphabetic value. For example:

```
U[A-Z] -->> UA, UB, UC.......UZ, UAA........
U[0-9]-X -->> U1-X, U2-X, U3-X......U9-X, U10-X.......
```

- A value in parentheses is specified to modify the value in square brackets (place parentheses after brackets).

  You can only specify this value once in a pattern. This value indicates a starting number or character other than 0 or A. The number of characters in this value controls the number of place holders in the reference designator. For example:

```
U[0-9](4422) -->> U4422, U4423, U4424, ... U9999, U10000...
U[A-Z](BBB) -->> UBBB UBBC UBBD ... UZZZ, UAAAA.....
```

- You can use a property name preceded by a dollar sign ($) in parentheses to add design properties such as page or drawing name.

  Packager-XL supplies the actual value on an instance basis.

  For example, the page and drawing properties can be specified as shown below:

  `U($PAGE)X[0-9](61)` attaches the name, U1X61 starting from the first instance of a part on page one.

  `U($DRAWING)X[0-9](61)` attaches the name, UIO_MODX61 starting from the first instance of a part on the design IO_MOD.

Note: Spaces are not allowed in a pattern.

Example

```
REF_DES_PATTERN ($PHYS_DES_PREFIX)-($DRAWING)-[0-9];
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

Note: You can use the `PHYS_DES_PREFIX` property as part of your `REF_DES_PATTERN` directive. The following pattern uses the `PHYS_DES_PREFIX` property as the first character of the reference designator and begins incrementing from number 501 to complete the pattern.

```
REF_DES_PATTERN ($PHYS_DES_PREFIX)[0-9](501);
```

If a resistor has `PHYS_DES_PREFIX=R`, `R501` is used. For a capacitor with `PHYS_DES_PREFIX=C`, `C501` is used. For more details, see the Allegro Platform Properties Reference guide.

REF_DES_PATTERN_FIX

The REF_DES_PATTERN_FIX directive is used in Allegro System Capture and Packager-XL. This directive is used to specify the format of reference designators (that is, location properties) assigned to the physical parts in a design.

When you use the `REF_DES_PATTERN_FIX` directive:

- The reference designator counter is reset on every page. Therefore, at every page change, the reference designator counters are reset to the initial values.

  Example

  For page1, the reference designators will be:

  ```
  U1XX0,
  U2XX1, ...
  ```

  For page2, the reference designators will be:

  ```
  U1XX0,
  U2XX1, ..
  ```

- The reference designator counter is maintained for every *Refdes* prefix. Therefore, different reference designator prefixes have their own reference designator counters.

  Example

  For refdes prefix `U`, the reference designators will be:

  ```
  U1XX0,
  U1XX1, ...
  ```

  For refdes prefix `C`, the reference designators will be:

  ```
  C1XX0,
  C1XX1, ...
  ```

The reference designators are generated in the new pattern only if you set the `REF_DES_PATTERN_FIX` directive to ON

If this directive is not set to ON, Packager-XL generates the reference designators using REF_DES_PATTERN.

The REF_DES_PATTERN_FIX directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and rads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture. For

information about how this directive works in Allegro System Capture, refer to REF_DES_PATTERN_FIX.

Caution

The REF_DES_PATTERN_FIX directive cannot be set from the Packager Setup dialog.

Syntax

```
REF_DES_PATTERN_FIX 'on'|'off'
```

Example

An excerpt from the *.cpm* file:

```
START_PKGRXL
REF_DES_PATTERN_FIX 'on'
ref_des_pattern '($phys_des_prefix)($PAGE)XX[0-9](0)'
END_PKGRXL
```

Corresponding UI Option for Allegro System Capture

Edit - Preference - Schematic - ECO/Packager - Reset Numbering on Each Page

Corresponding UI Option for Design Entry HDL

None

REFDES_ALPHA_NUM

Converts the refdes number generated by the packager to an alphanumeric or numeric value.

Syntax

```
REFDES_ALPHA_NUM 'FULL_ALPHA'|'UNIT_NUM'
```

`FULL_ALPHA`	converts the whole number to base 36 number. **Example:** 1, 2, 3, ..., 9; A, B, C, ..., Z, 10, 11, ...,19, 1A, 1B,..., 1Z, 20
`UNIT_NUM`	converts the number except the least significant digit, that is the number at the unit's place always remains a number. **Example:** 1, 2,3, ...,9, 10, 11,...,99, A0, A1, ..., A9, B0, B1,...B9,..., 1A0, 1A1, ..., 1A9,..., 1Z0, ...1Z9, ..., 2Z0...

Example

```
REFDES_ALPHA_NUM 'FULL_ALPHA'
```

- Numbers 0-9 remain as is
- 10 is converted A
- 35 is converted to Z
- 36 is converted to 10
- 46 is converted to 1A

```
REFDES_ALPHA_NUM 'UNIT_NUM'
```

- 100 is converted to A0

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - RefDes Numbering Characters

See Also

- REF_DES_PATTERN
- REF_DES_PATTERN_FIX
- REFDES_PAGE_PADDING

REFDES_PAGE_PADDING

Provides support for zero padding in page number. It is used if `'$Page'` is used in REF_DES_PATTERN.

Syntax

```
REFDES_PAGE_PADDING <zero padding>
```

where

`padding`	is the zero padding value for the page number.

Example

```
REFDES_PAGENUM_PADDING '2'
```

With a zero padding of 2:

- Page number 2 is displayed as 02
- Page number 10 is displayed as 10. It will be 0A in case of alphanumeric numbering, `REFDES_ALPHA_NUM 'FULL_ALPHA'`.

  Note: The numbering of pages follow the same order as defined in the REFDES_ALPHA_NUM directive.

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Fixed Page Number Length

See Also

- REF_DES_PATTERN
- REF_DES_PATTERN_FIX
- REFDES_ALPHA_NUM

REFDES_PREFIX_VISIBILITY_CHECK

Specifies the reference designators to be shown on the design canvas in Allegro System Capture.

This directive is used when the `RefDes visibility` audit rule under the *Graphical Checks* category is run.

Syntax

```
REFDES_PREFIX_VISIBILITY_CHECK '<refdes prefix>'
```

Example

```
REFDES_PREFIX_VISIBILITY_CHECK 'TP' 'J'
```

Corresponding UI Option in Allegro System Capture

Graphical Rule – Invalid Net Name – Configure – Schematic Audit Settings – Rule tab – *RefDes Visibility*

REGENERATE_PHYSICAL_NET_NAME

The `REGENERATE_PHYSICAL_NET_NAME` directive is used to delete all existing physical net names in a design and generate them afresh. The use of this directive ensures that the changes done on net names in PCB Editor are not lost during successive packaging.

Syntax

```
REGENERATE_PHYSICAL_NET_NAME 'ON'|'OFF';
```

on	Regenerates physical net names. That is, Packager-XL deletes all existing physical net names and generate them afresh.
off	Packager-XL uses and maintains the existing physical net names.

The default value of the `REGENERATE_PHYSICAL_NET_NAME` directive is `off`.

REMOVE_FROM_STATE

The `REMOVE_FROM_STATE` directive is used to remove properties from the state file.

Unwanted properties can be present in the state file because of the following reasons:

- When you delete a property from the schematic, but it remains in the state file.

  During feedback, a property from the layout system is written to the state file but is not backannotated to the schematic.You can delete these unwanted properties by using the `REMOVE_FROM_STATE` directive.

Caution

You should use the REMOVE_FROM_STATE directive with extreme caution. Properties specified in this directive are removed from every package in the state file. Therefore, to resolve most state file conflicts, use the STATE_WINS_OVER_DESIGN or REPACKAGE directives.

Syntax

```
REMOVE_FROM_STATE all|property [,property] ... ;
```

`all`	Removes all properties from the state file.
`property`	Represents a property in the schematic.

The default value for the `REMOVE_FROM_STATE` directive is *none*.

Example

```
REMOVE_FROM_STATE group
```

rename_folders_on_copyproj

This directive is used when creating a copy of a project in Allegro System Capture. If the project being copied has subfolders, such an output folder, use this directive to rename folders that get copied to the new design name.

Syntax

`rename_folders_on_copyproj './output/^design_name^/derived_data'`

Example

```
rename_folders_on_copyproj './output/^design_name^/derived_data'
'./output/^design_name^/hwconfig'
'./output/^design_name^/physical'
```

In this example, three folders, namely derived *`data`*, *`hwconfig`*, and *`physical`* under the `output` folder will get created with the new project's name.

Corresponding UI Option for Allegro System Capture

None

See Also

delete_folders_on_copyproj

REPACKAGE

The `REPACKAGE` directive specifies whether or not existing tool-assigned packaging information is used in the current run of Packager-XL.

The tool-assigned packaging is read from the state file and from the `CDS_LOCATION`, `CDS_SEC`, and `CDS_PN` properties in the schematic.

Regardless of the setting for the `REPACKAGE` directive, user-assigned properties, that is, `LOCATION`, `SEC` and `PN`, from the schematic are always preserved.

Syntax

```
REPACKAGE 'on'|'off';
```

on	Ignores tool-assigned packaging. That is, Packager-XL ignores the `CDS_LOCATION`, `CDS_SEC`, and `CDS_PN` properties from the schematic as well as from the packaging in the state file.
off	Packager-XL uses and maintains existing packaging assignments stored in the state file and the schematic.

The default for the `REPACKAGE` directive is `off`.

Example

```
REPACKAGE on;
```

REPLACE_ACT_AS_MODIFY

When this directive is set to ON, the Replace command will act as the Modify command.

For more information about the Replace and Modify commands, refer to the following sections of *Allegro Design Entry HDL Reference Guide*: *Replace*, *Modify*

Syntax

```
REPLACE_ACT_AS_MODIFY 'ON'|'OFF'
```

Example

```
REPLACE_ACT_AS_MODIFY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

REPLACE_PTF_PROPS

Causes PTF property changes such as property visibility changes to apply when you load an existing PPT Option set file, or make dynamic changes to several injected properties using Part Manager.

Syntax

```
REPLACE_PTF_PROPS 'OFF'|'ON'
```

Example

```
REPLACE_PTF_PROPS 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

REPORT_COL_PAD

Use this specify the character to use as the column pad in Text File in Tabular Form reports. The column pad is the space between two columns in the report.

Syntax

```
report_col_pad <character>
```

Example

```
report_col_pad ' '
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Text Report Separators — Column Pad

See Also

- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_COL_SEP

Use this directive to specify the characters to use as the column separator in Text File in Tabular Form reports.

Syntax

```
report_col_sep <character>
```

Example

```
report_col_sep '|'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Text Report Separators — Column Separator

See Also

- REPORT_COL_PAD
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_CURRENCY_CHAR

Use this directive to specify the characters to use as a currency symbol in reports.

Syntax

```
report_currency_char <symbol>
```

Example

```
report_currency_char '$'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Currency Symbol

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_DIR

Use this directive to specify the directory location to create the reports in System Connectivity Manager and Allegro System Capture. This directive is used for generating multiple reports.

Syntax

```
report_dir <directory path>
```

Example

```
report_dir './reports'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Report Location

Corresponding UI Option for Allegro System Capture

None

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP

- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_FILES

Use this directive to specify the path and file name of the report files.

Syntax

```
report_files <path to report files>
```

Example

```
report_files './report/processor_Bill of Materials.dsr' './reports/processor_Bill
     of Materials.html' './reports/processor_1234.html' './reports/
     processor_12345.dsr'
```

Corresponding UI Option for System Connectivity Manager

None

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_FONT_FACE

Use this directive to specify the font to use in the report.

Syntax

```
report_font_face <font face>
```

Example

```
report_font_face 'arial'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Font

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_FONT_SIZE

Use this directive to specify the font size to use in the report.

Syntax

```
report_font_size <size>
```

Example

```
report_font_size '10'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Font Size

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_FONT_STYLE

Use this directive to specify the font style to use in the report.

Syntax

```
report_font_style <font style>
```

The valid values are: Regular, Bold, Italic, Bold Italic

Example

```
report_font_style 'Bold'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Font Style

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_FORMAT

Use this directive to specify the default format in which reports are generated.

Syntax

```
report_format <format>
```

The valid values are: DSR, Text, CSV, HTML

Example

```
report_format 'DSR'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Default Format

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_HEADER_SEP

Use this directive to specify the character you want to use as the separator for column headings in Text File in Tabular Form reports.

Syntax

```
report_header_sep <character>
```

Example

```
report_header_sep '#'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Text Report Separators — Column Separator

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_HIDE_LINENO

Use this directive to specify if line numbers are to be displayed in the generated reports.

Syntax

```
report_hide_lineno 'ON'|'OFF'|'0'|'1'
```

Example

```
report_hide_lineno '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Hide Line Numbers

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_ROW_SEP
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_ROW_SEP

The character to specify the character to use as the row separator in Text File in Tabular Form reports.

Syntax

```
report_row_sep <character>
```

Example

```
report_row_sep '-'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Text Report Separators — Row Separator

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_SORT_ORDER
- REPORT_STRING_SEP

REPORT_SORT_ORDER

Use this directive to specify the order in which the data in reports is to be sorted.

Syntax

```
report_sort_order <order>
```

The valid values are: Ascending, Descending

Example

```
report_sort_order 'Ascending'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — Sort Order

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_STRING_SEP

REPORT_STRING_SEP

Use this directive to specify the character to use a string separator in reports.

Syntax

```
report_string_sep <character>
```

Example

```
report_string_sep '#'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Report Generation — General Settings — String Separator

See Also

- REPORT_COL_PAD
- REPORT_COL_SEP
- REPORT_CURRENCY_CHAR
- REPORT_DIR
- REPORT_FILES
- REPORT_FONT_FACE
- REPORT_FONT_SIZE
- REPORT_FONT_STYLE
- REPORT_FORMAT
- REPORT_HEADER_SEP
- REPORT_HIDE_LINENO
- REPORT_ROW_SEP
- REPORT_SORT_ORDER

RESTRICTIVE_WIRE_MOVE

Allows wire moves in only one direction – either horizontally along the x-axis, or vertically along the y-axis. The direction is determined by the direction of mouse movement at the beginning of the operation.

Syntax

```
RESTRICTIVE_WIRE_MOVE 'YES'|'NO'
```

YES	Wires move only in the direction of mouse movement.
NO	Wires can move in any direction.

Example

```
RESTRICTIVE_WIRE_MOVE 'YES'
```

RETAIN_EXISTING_XNETS_AND_DIFFPAIRS

Use this directive to enable or disable signal and ECSet validation when Constraint Manager is launched, or when Packager-XL is run.

Release 16.5 and onwards, by default, signal models and ECSets are validated when launching Packager-XL or Constraint Manager. If you want to disable this validation, set this directive in the `START_ECSET_MODELS` section of your `.cpm` file.

Note: If you are working with pre-16.5 designs, you can enable or disable signal and ECSet validation using the *Retain Existing XNets and Diff Pairs* option in the DE-HDL Options dialog box.

To disable validation, valid values for this directive are YES/ON/TRUE. To enable validation, valid values are NO/OFF/FALSE.

Syntax

```
RETAIN_EXISTING_XNETS_AND_DIFFPAIRS 'YES'|'ON'|'TRUE'|'NO'|'OFF'|'FALSE'
```

Example

```
RETAIN_EXISTING_XNETS_AND_DIFFPAIRS 'NO'
```

Corresponding UI Option for Allegro Design Entry HDL

None

RETAIN_FONT_SETTINGS_ON_SYMBOL_ADD

Set the value of this directive to ON if you want the symbol font settings that were defined when creating the symbol to be preserved when instantiating a component on a schematic.

For example, assume that the text color is defined as red for a symbol in Part Developer. When you instantiate this symbol in DE-HDL, the symbol color will still be red regardless of the symbol text font color specified in DE-HDL.

Syntax

```
RETAIN_FONT_SETTINGS_ON_SYMBOL_ADD 'YES'|'ON'|'TRUE'|'NO'|'OFF'|'FALSE'
```

Example

```
RETAIN_FONT_SETTINGS_ON_SYMBOL_ADD 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

RETAIN_HARD_LOCATION_ON_REPLACE

Retains the hard location properties of an instance when a component is replaced on the schematic.

Syntax

```
RETAIN_HARD_LOCATION_ON_REPLACE 'OFF'|'ON'
```

Example

```
RETAIN_HARD_LOCATION_ON_REPLACE 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

RETAIN_PATH_ON_REPLACE

RETAIN_VERSION_ON_REPLACE

RETAIN_HARDLOCATION_ON_COPY

Retains the value of the LOCATION property when a component is copied.

Syntax

```
RETAIN_HARDLOCATION_ON_COPY 'OFF'|'ON'
```

where

OFF	Ensures that the value of the LOCATION property is reset to ? when you copy an instance with a hard location.
ON	The value of the LOCATION property is retained when a component is copied. This the default value.

Example

```
RETAIN_HARDLOCATION_ON_COPY 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

RUN_FEEDBACK

RETAIN_INSTANCE_PROP_ON_VERSION

If the symbol properties and the instance-level properties on a schematic are different, DE-HDL retains the instance-level properties and not the symbol properties when this directive is ON.

Syntax

```
RETAIN_INSTANCE_PROP_ON_VERSION 'OFF'|'ON'
```

Example

```
RETAIN_INSTANCE_PROP_ON_VERSION 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

RETAIN_LOCATION_ON_COPYALL

Retains the pin numbers and location of a component when you perform the copy all operation.

Syntax

```
RETAIN_LOCATION_ON_COPYALL 'OFF'|'ON'
```

Example

```
RETAIN_LOCATION_ON_COPYALL 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

RUN_FEEDBACK

RETAIN_PATH_ON_REPLACE

This directive can be set to retain the PATH property value when replacing a component on a schematic. This ensures that the canonical path of the component instance is maintained. As a result, there will be no rip-offs on the layout.

Syntax

```
RETAIN_PATH_ON_REPLACE 'OFF'|'ON'
```

Example

```
RETAIN_PATH_ON_REPLACE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

RETAIN_HARD_LOCATION_ON_REPLACE

RETAIN_VERSION_ON_REPLACE

RETAIN_PREVIOUS_HILITE

Retains the highlighting of multiple objects on the same page.

By default, a highlighted object is dehighlighted when a new object is selected in Allegro PCB Editor or the Global Navigate window in Design Entry HDL.

In the case of groups, when you select multiple objects by dragging the mouse, DE-HDL highlights any one of the selected objects in the group on the same page. For example, if you group C1, C2, C3, and C4 on a board file in PCB Editor, DE-HDL will highlight any one of these objects in the schematic page. However, if you group objects by choosing Edit – Groups, DE-HDL will only highlight the last selected object in the group.

If you want all the selected objects on the same page to be highlighted at the same time, set the `RETAIN_PREVIOUS_HILITE` directive to `ON` in the project (`.cpm`) file. This ensures that all the selected objects are highlighted.

Syntax

```
RETAIN_PREVIOUS_HILITE 'OFF'|'ON'
```

Example

```
RETAIN_PREVIOUS_HILITE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

RETAIN_VERSION_ON_REPLACE

Retains the current version of a symbol when the symbol is replaced by another symbol on the schematic drawing.

Syntax

```
RETAIN_VERSION_ON_REPLACE 'OFF'|'ON'
```

Example

```
RETAIN_VERSION_ON_REPLACE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Retain Version on Replace

RETAIN_ZERONODE_NET

Controls whether a net is deleted from the design database after it is removed from all the pages on the schematic. This directive ensures that the constraints and properties are not lost if you remove circuitry from one page to another.

Syntax

```
RETAIN_ZERONODE_NET 'YES'|'NO'
```

YES	A net is not deleted from the database even if it is removed from the schematic pages.
NO	If a net is deleted from the of the schematic pages, it is deleted from the design database as well.

Example

```
RETAIN_ZERONODE_NET 'YES'
```

REUSE_REFDES

The `REUSE_REFDES` directive is used to control the reuse of reference designators in a project in Allegro System Capture and Packager-XL. The reference designator of a physical device can be changed or deleted in the schematic or the board. The use of the `REUSE_REFDES` directive in the Preserve mode provides Packager-XL with one of the two options:

- Reuse the existing reference designators. The values of the existing reference designators are stored in the `pxl.state` file. Packager-XL uses this file to reuse reference designators.

  Note: If the Packager-XL is run in the Repackage mode, then the `pxl.state` file is deleted and the list of existing reference designators is lost. All the components in a design will be packaged afresh.

- Lock the previously used reference designator, and assign a new reference designator for the new component. In this case, Packager-XL will continue to store all reference designators assigned by PCB Editor in the reference designator section of the `pxl.state` file.

The REUSE_REFDES directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and rads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
REUSE_REFDES 'ON'|'OFF';
```

where

on	is the default value. When the `REUSE_REFDES` directive is set to `on`, the existing reference designators for changed or deleted components are reused for new components.

off	when the `REUSE_REFDES` directive is set to `off`, new reference designators are assigned to new components. Before assigning reference designators to new components, Packager-XL reads the list of reference designators in the `pxl.state` file. The existing reference designators are not reused for new components that need new assignments. If Packager-XL can accommodate the new components with existing reference designators, then it reuses those designators. Otherwise, the new components are assigned new reference designators.

Example

Assume that there are 5 components in a design that are assigned the reference designators `U1` to `U5` by PCB Editor. When you package the design in the feedback mode, the information about reference designators is stored in the `pxl.state` file.

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - ECO/Packager - Reuse Reference Designator's Numbers

Corresponding UI Option for Design Entry HDL

None

Note: If you have not packaged your design in feedback mode, the list of reference designators will not be stored in the `pxl.state` file. As a result, Packager-XL will assume that the reference designators can be reused.

If you now delete the component with the reference designator `U2`, add a new component in the schematic, and then package the design, then whether the new component should be assigned the U2 reference designator value is decided by the `REUSE_REFDES` directive.

If the `REUSE_REFDES` directive is set to `on`, the new component will be assigned the reference designator value U2. If the `REUSE_REFDES` directive is set to `off`, the new component will **not** be assigned the reference designator value U2.

RIGHT_IN_OFFPAGE

Use this directive to specify the lib:cell:view of the symbol to be used as offpage connector for the input signals on the right of the schematic page.

Syntax

```
right_in_offpage <library.cell:view>
```

Example

```
right_in_offpage 'standard.offpage:sym_4'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — Input.

See Also

USE_OFFPAGE

RIGHT_IN_ROT

RIGHT_IO_OFFPAGE

RIGHT_IO_ROT

RIGHT_OUT_OFFPAGE

RIGHT_OUT_ROT

RIGHT_IN_ROT

Use this directive to specify the angle in degrees by which the right_in_offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by left_in_offpage as input offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
right_in_rot <angle>
```

Example

```
right_in_rot '180'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — Input — Rotate By.

See Also

USE_OFFPAGE

RIGHT_IN_OFFPAGE

RIGHT_IO_OFFPAGE

RIGHT_IO_ROT

RIGHT_OUT_OFFPAGE

RIGHT_OUT_ROT

RIGHT_IO_OFFPAGE

Use this directive to specifies the lib:cell:view of the symbol to be used as offpage connector for the input/output signal on the right of the schematic page.

Note: This option works only when the use_offpage is set to 1.

Syntax

```
right_io_offpage <library.cell:view>
```

Example

```
right_io_offpage 'standard.offpage:sym_3'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — InOut — Rotate By

See Also

USE_OFFPAGE

RIGHT_IN_OFFPAGE

RIGHT_IN_ROT

RIGHT_IO_ROT

RIGHT_OUT_OFFPAGE

RIGHT_OUT_ROT

RIGHT_IO_ROT

Use this directive to specify the angle in degrees by which the right_io_offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by left_io_offpage as input/output offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
right_io_rot <angle>
```

Example

```
right_io_rot '45'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — InOut — Rotate By

See Also

USE_OFFPAGE

RIGHT_IN_OFFPAGE

RIGHT_IN_ROT

RIGHT_IO_OFFPAGE

RIGHT_OUT_OFFPAGE

RIGHT_OUT_ROT

RIGHT_OUT_OFFPAGE

Use this directive to specifies the lib:cell:view of the symbol to be used as offpage connector for the output signal on the right of the schematic page.

Note: This option works only when the use_offpage is set to 1.

Syntax

```
right_out_offpage <library.cell:view>
```

Example

```
right_out_offpage 'standard.offpage:sym_5'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — Output.

See Also

USE_OFFPAGE

RIGHT_IN_OFFPAGE

RIGHT_IN_ROT

RIGHT_IO_OFFPAGE

RIGHT_IO_ROT

RIGHT_OUT_ROT

RIGHT_OUT_ROT

Use this directive to specify the angle in degrees by which the right_out_offpage connector symbol should be rotated before it is placed on the document schematic.

Tip

To use the symbol specified by left_out_offpage as output offpage connector for signals on the left of the schematic page, rotate the symbol by 180.

Syntax

```
right_out_rot <angle>
```

Example

```
right_out_rot '45'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols — Right Side — Out — Rotate By.

See Also

USE_OFFPAGE

RIGHT_IN_OFFPAGE

RIGHT_IN_ROT

RIGHT_IO_OFFPAGE

RIGHT_IO_ROT

RIGHT_OUT_OFFPAGE

RUN_FEEDBACK

This directive corresponds to the Package Design check box in the Import Physical dialog. If you want to transfer the physical design from the PCB Editor layout database to the Design Entry HDL schematic design, set this directive to '`YES`'.

The `run_feedback` directive is written to the `START_DESIGNSYNC` section with the value '`YES`' when the Package Design check box in the Import Physical dialog is selected.

Design Synchronization uses this value to remember the last setting with which the dialog was last run. The next time the design is packaged, Design Synchronization selects the check box by default when it launches the Import Physical dialog.

Syntax

```
run_packager 'ON'|'OFF'|'YES'|'NO'|'1'|'0'
```

Example

```
run_packager 'YES'
```

Corresponding UI Option for Design Entry HDL

File — Import Physical — Import Physical dialog —Package Design check box

RUN_GENFEEDFORMAT

When the value of this directive is set to '`YES`', the check box for *Generate Feedback Files* in the Import Physical dialog is selected when the dialog is launched. This generates feedback files during the Import Physical process. These files, which can be used to synchronize the schematic and the board, contain the connectivity and property information of the board.

Syntax

```
run_genfeedformat 'YES'|'NO'
```

Example

```
run_genfeedformat 'YES'
```

Corresponding UI Option for Design Entry HDL

File — Import Physical — Import Physical dialog — Generate Feedback Files check box

RUN_HIERWR_BEFORE_PXL

When updating a PCB Editor board with changes in the schematic (*Export — Physical*), if you want Design Entry HDL to validate and regenerate the design logical netlist before generating the physical netlist, set this directive to `ON`.

When the directive is set to ON, Packager-XL triggers a `hier_write` command, that is, it saves the entire design before validating and regenerating the design physical netlist.

Even if Design Entry HDL is not running but Packager-XL is called from another application (e.g., Design Sync), the design logical netlist is validated and regenerated.

If this directive is `ON`, and you run `pxl -proj` from the command line, then too Design Entry HDL validates and regenerates the design logical netlist before generating the physical netlist.

Note: When Packager-XL is run from the command line, it saves the hierarchy before validating and regenerating the design logical netlist before the physical netlist. However, when you run *Export — Physical,* Design Sync saves the hierarchy before Packager-XL runs. As a result, Packager-XL does not save the hierarchy again and the command line syntax or log file contains a `-nosavehier` argument.

Note: Packager-XL will abort operations if there are netlisting errors. This behavior is based on the `STOP_PACKAGE_ON_SCHEMATIC_ERROR` directive.

Syntax

```
run_hierwr_before_pxl 'YES'|'NO'
```

Example

```
run_hierwr_before_pxl 'YES'
```

Corresponding UI Option for Design Entry HDL

None

See Also

STOP_PACKAGE_ON_SCHEMATIC_ERROR

RUN_NETREV

This directive corresponds to the *Update PCB Editor Board (Netrev)* check box in the Export Physical dialog. If you want to update the PCB Editor board with changes in the schematic when exporting a design, set this directive to '`ON`'.

The `run_netrev` directive is written to the `START_DESIGNSYNC` section with the value `'ON'` when the *Update PCB Editor Board (Netrev)* check box is selected in the Export Physical dialog.

The next time the design is packaged, Design Synchronization selects the check box by default when it launches the Export Physical dialog.

Syntax

```
run_netrev 'ON'|'OFF'
```

Example

```
run_netrev 'ON'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Update PCB Editor Board (Netrev) check box

RUN_PACKAGER

This directive corresponds to the *Package Design* check box in the Export Physical dialog. If you want to update the PCB Editor board with changes in the schematic when exporting a design, set this directive to '`ON`'.

The `run_packager` directive is written to the `START_DESIGNSYNC` section with the value `'ON'` when the *Package Design* check box in the Export Physical dialog is selected. Design Synchronization uses this value to remember the last setting with which the dialog was last run.

The next time the design is packaged, Design Synchronization selects the check box by default when it launches the Export Physical dialog.

Syntax

```
run_packager 'ON'|'OFF'
```

Example

```
run_packager 'ON'
```

Corresponding UI Option for Design Entry HDL

File — Export Physical — Export Physical dialog — Package Design check box

18

S Directives

This chapter lists the CPM directives that start with `S` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

SCALAR_PARENTHESIS

This directive enables importing ‘(' and ')' characters in scalar net names from DE-HDL designs. To enable this behavior, this directive must be added to the `site.cpm`. This makes System Capture treat net names such as VCC(+) and NET(ABC) as scalar nets. The SCALAR_PARENTHESIS works with the MULTI_FORMAT directive.

Note: There is no support for '(' and ')' in vector nets in System Capture. For net names such as net(1) or net(0..7), the '(', ')' are replaced with '<', '>' internally, which is existing feature.

Syntax

```
SCALAR_PARENTHESIS '<value>'
```

Example

```
START_CANVAS

SCALAR_PARENTHESIS 'TRUE'

END_CANAVS
```

Corresponding UI Option for System Capture

None

SAVEHIER_READONLY_MSG_AS_INFO

When you run Save Hierarchy in a design, read-only pages are skipped and an error message is displayed with details of the read-only and locked pages.

However, there may be times when you work in a design with referenced blocks rather than copied blocks. In such cases, when you run Save Hierarchy, you may want an information message displayed rather than an error message.

When this directive is set to 'ON', DE-HDL displays an information message, and not an error. The information message provides details of the read-only pages that were skipped.

Syntax

```
SAVEHIER_READONLY_MSG_AS_INFO 'OFF'|'ON'
```

Example

```
SAVEHIER_READONLY_MSG_AS_INFO 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SAVE_WORKSPACE

Saves window and toolbar settings when you exit Design Entry HDL.

Syntax

```
SAVE_WORKSPACE 'OFF'|'ON'
```

Example

```
SAVE_WORKSPACE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Save Layout at Exit

SHOW_ALL_BLOCKS_FOR_IMPORT

Controls the availability of uninstantiated blocks when importing blocks in System Capture. When enabled in the `site.cpm`, the import block dialog would show the blocks outside the design hierarchy under a separate node and the blocks can be chosen and imported.

Syntax

```
SHOW_ALL_BLOCKS_FOR_IMPORT 'YES' | 'NO'
```

Example

```
SHOW_ALL_BLOCKS_FOR_IMPORT 'YES'
```

Corresponding UI Option

None

SCH_POWER_GROUP_WINS_OVER_PPT

Use this directive to ignore the `POWER_GROUP` property in the `.ptf` file.

Syntax

```
SCH_POWER_GROUP_WINS_OVER_PPT 'ON'|'OFF'
```

Example

```
SCH_POWER_GROUP_WINS_OVER_PPT 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SCL_PIN_PATTERN

Defines the patterns used for identifying pins as Serial Clock (SCL) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `SDA and SCL pins of ICs incorrectly connected to non-SDA or non-SCL pins`
- `SDA and SCL pins of ICs not connected to the same set of ICs or connectors`

Syntax

```
SCL_PIN_PATTERN '<pin pattern>'
```

Example

```
SCL_PIN_PATTERN 'SCL'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *SCL Pin Patterns*

SDA_PIN_PATTERN

Defines the patterns used for identifying pins as Serial Data (SDA) pins.

This directive is used when the following *Connectivity Checks* audit rules are run:

- `SDA and SCL pins of ICs incorrectly connected to non-SDA or non-SCL pins`
- `SDA and SCL pins of ICs not connected to the same set of ICs or connectors`

Syntax

```
SDA_PIN_PATTERN '<pin pattern>'
```

Example

```
SDA_PIN_PATTERN 'SDA' 'SDACLK'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Parameters tab – *SDA Pin Patterns*

SHOW_PNN_SIGNAME

Use to view the winning hierarchical net name in a hierarchical design. To make physical net names visible in Design Entry HDL, set this directive to `TRUE`. The next time you open the design in Design Entry HDL the property will be displayed with the winning value.

Syntax

```
SHOW_PNN_SIGNAME 'TRUE'|'FALSE'
```

Example

```
SHOW_PNN_SIGNAME 'TRUE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools - Options - Design Entry HDL Options dialog box - General tab- Show Physical Net Name

SHOW_POWER_SIGNAL_NAMES_FOR_NEW_CONNECTIONS

Controls if signal names on wires are added or displayed automatically when the wires are connected to power sources in various scenarios. This directive is set to `TRUE` by default

Syntax

```
SHOW_POWER_SIGNAL_NAMES_FOR_NEW_CONNECTIONS 'TRUE' | 'FALSE'
```

Example

```
SHOW_POWER_SIGNAL_NAMES_FOR_NEW_CONNECTIONS 'TRUE' | 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit — Preferences — Schematic — General — Show power signal names for new wire connections

See Also

- SHOW_POWER_SIGNAL_NAMES_FOR_NEW_CONNECTIONS
- IGNORE_HIDDEN_SCALAR_SIGNAL_NAMES_WHEN_PASTING
- IGNORE_HIDDEN_UNNAMED_SCALAR_SIGNAL_NAMES_WHEN_PASTING

SHOW_PROPERTIES

Temporarily makes invisible cross-references visible on the schematic.

Syntax

```
SHOW_PROPERTIES 'ON'|'OFF'
```

Example

```
SHOW_PROPERTIES 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SHOW_VARIANT_COLORS

When set to 'ON', this directive enables the variant color setup options. You can use these options to define how you want to view variant schematics.

Syntax

```
SHOW_VARIANT_COLORS 'ON'|'OFF'
```

Example

```
SHOW_VARIANT_COLORS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SHOW_XNET_STATUS

When set to 'ON', all the components that have XNets on their pins are marked with a blue arrow and all the components that do not have XNets on their pins are marked with a blue cross sign. This indication is displayed on components that are 2-pin discrete devices and are electrical parts.

Syntax

```
SHOW_XNET_STATUS 'ON'|'OFF'
```

The default value is OFF.

Example

```
SHOW_XNET_STATUS 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

- *Edit — Component — Show XNets Status* (If Windows mode is enabled)
- *Component — Show XNets Status* (If Windows mode is disabled)

See Also

- XNET_ABSENT_COLOR
- XNET_EXISTS_COLOR

SMART_PDF_LINE_WIDTH_FACTOR

Controls the thickness of the lines of design elements, such as nets, page borders, auto shapes, and so on, in the PDF generated from Allegro System Capture designs using the Smart PDF option. The width can be 1 through 10 PDF units. The default value is 10. To have lines with the same thickness as that on the canvas, set the value to 10.

Syntax

```
SMART_PDF_LINE_WIDTH_FACTOR '<value>'
```

Example

```
SMART_PDF_LINE_WIDTH_FACTOR '10'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SORT_ROWS_ON_ATTRFORM_LAUNCH

Prior to 16.6 QIR 9, properties in the Attributes dialog were displayed in a random order. Post 16.6 QIR 9, attribute form properties are sorted in alphanumeric order.

If you want to revert to the previous behavior, use this directive.

Syntax

```
SORT_ROWS_ON_ATTRFORM_LAUNCH 'ON'|'OFF'
```

Example

```
SORT_ROWS_ON_ATTRFORM_LAUNCH 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

STICKY

Deletes a default property (dangling property) from a schematic when the property has been deleted from a symbol drawing.

Syntax

```
STICKY 'OFF'|'ON'
```

Example

```
STICKY 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

SYMBOL_COLOR

Changes the default symbol/body color.

Syntax

```
SYMBOL_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
SYMBOL_COLOR 'GREEN'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Symbol

See Also

- ARC_COLOR
- BACKGROUND_COLOR
- DOT_COLOR
- HIGHLIGHT_COLOR
- WIRE_COLOR

SYMBOL_DOT_RADIUS

Adjusts the diameter of dots at wire connections in symbol drawings.

Syntax

```
SYMBOL_DOT_RADIUS '<value>'
```

The valid values range from 1 to 40. For any value greater than 40, DE-HDL retains the last valid value set by the user.

Example

```
SYMBOL_DOT_RADIUS '13'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Dots — Symbol Dot Radius

See Also

LOGIC_DOT_RADIUS

SYMBOL_GRID_MULTIPLE

Displays every nth grid line to define where objects can be placed so that pins do not fall off-grid. This ensures the correct connectivity of wires and symbols.

Syntax

```
SYMBOL_GRID_MULTIPLE '<value>'
```

where

value any positive number.

Example

```
SYMBOL_GRID_MULTIPLE '5'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Symbol Grid— Multiple

See Also

- LOGIC_GRID_MULTIPLE
- DOC_GRID_MULTIPLE
- SYMBOL_GRID_SIZE
- SYMBOL_GRID_TOGGLE
- SYMBOL_GRID_TYPE

SYMBOL_GRID_SIZE

Adjusts the symbol grid size to be smaller or larger.

Syntax

```
SYMBOL_GRID_SIZE '<value>'
```

where

value	any number greater than 0.002.

Example

```
SYMBOL_GRID_SIZE '0.050'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Symbol Grid— Size

See Also

- SYMBOL_GRID_MULTIPLE
- SYMBOL_GRID_TOGGLE
- SYMBOL_GRID_TYPE

SYMBOL_GRID_TOGGLE

Displays or hides the grid.

Syntax

```
SYMBOL_GRID_TOGGLE 'ON'|'OFF'
```

Example

```
SYMBOL_GRID_TOGGLE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Symbol Grid — TOGGLE

See Also

- SYMBOL_GRID_MULTIPLE
- SYMBOL_GRID_SIZE
- SYMBOL_GRID_TYPE
- LOGIC_GRID_TOGGLE
- DOC_GRID_TOGGLE

SYMBOL_GRID_TYPE

Displays the grid as dots or dashed Lines.

Syntax

```
SYMBOL_GRID_TYPE 'LINE'|'DOT'
```

Example

```
SYMBOL_GRID_TYPE 'LINE'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Grid page — Show Symbol Grid — Type

See Also

- SYMBOL_GRID_MULTIPLE
- SYMBOL_GRID_SIZE
- SYMBOL_GRID_TOGGLE
- LOGIC_GRID_TYPE

SYNC_ON_PAGE_EDIT

Checks for differences between the schematic and library cells every time you move from one page to another in a design.

Syntax

```
SYNC_ON_PAGE_EDIT 'ON'|'OFF'
```

Example

```
SYNC_ON_PAGE_EDIT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Metadata Options page — Schematic Metadata and Revision Check Options — Launch Component Revision Manager on Page Edit

See Also

- GENERATE_SCH_METADATA
- SYNC_ON_STARTUP

SYNC_ON_STARTUP

Checks for differences between the schematic and library cells when you open a design in Design Entry HDL. By default, the ability to check for differences between the schematic and library cells is set to off.

Syntax

```
SYNC_ON_PAGE_EDIT 'ON'|'OFF'
```

Example

```
SYNC_ON_PAGE_EDIT 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Metadata Options page — Schematic Metadata and Revision Check Options — Launch Component Revision Manager

See Also

- GENERATE_SCH_METADATA
- SYNC_ON_PAGE_EDIT

SETCONCEPTFONT

Allows you to map the default DE-HDL font, `ConceptFont`, with any font of your choice.

If you use `ConceptFont` in your schematic design, by default, DE-HDL prints PDFs using the Courier New font and embeds it in the PDF document.

Depending on the size of the text in the schematic, there can be times when the text bounding boxes, such as the signal name (sig_name), overlap. This can make it difficult to search for specific text in the PDF.

In such cases, you can do the following:

- Reduce the text font size so that the text bounding boxes are smaller.
- Use the PDFFont directive.
- Use this directive and map a font that matches well with `ConceptFont`.

Syntax

```
SETCONCEPTFONT '<Font of your choice>'
```

Example

```
SETCONCEPTFONT 'Arial'
```

Corresponding UI Option for Design Entry HDL

None

See Also

PDFFont

SCH_XR_FORMAT_IN_REPORTS

Defines whether the XR formatting specified for generation of XRs (with/without zone, with/without arrows) will be applicable in creferHDL reports.

Value

```
SCH_XR_FORMAT_IN_REPORTS 'ON'|'OFF'
```

Example

```
SCH_XR_FORMAT_IN_REPORTS 'OFF'
```

SPREADSHEET

Creates the output file in the spreadsheet format, which can be imported in an application such as MS Excel.

Value

```
SPREADSHEET 'ON'|'OFF'
```

Example

```
SPREADSHEET 'ON'
```

Corresponding UI Option for Project Manager

Tools — Packager Utilities — Bill of Materials — BOM-HDL dialog box — Report Format — Spreadsheet Format option button

SD_SUFFIX_SEPARATOR

The SD_SUFFIX_SEPARATOR directive is used to assign new reference designators. In case you reuse a design, Packager-XL assigns new reference designators for all reuse modules and updates their names. For example, if there is an instance of a reuse module named `U1`, then Packager-XL updates it to `U1_1`. By default, the character "_" is used in assigning new reference designators. The `SD_SUFFIX_SEPARATOR` directive is used to define a different character for renaming reference designators for reuse modules.

The SD_SUFFIX_SEPARATOR directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
SD_SUFFIX_SEPARATOR '<character_name>'
```

where

character_name is the new suffix for renaming reference designators.Its value should follow the same rules as used for naming the `location` property.

Note: If you use a blank space or an exclamation mark (!) instead of the default separator (_), then Packager-XL will ignore the character and retain the default separator.

Example

Assume that you have an instance of reuse module named `U1`. If you now define the following directive:

```
SD_SUFFIX_SEPARATOR '#'
```

then after packaging, the instance is assigned the reference designator `U#1` instead of `U1_1`.

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Packager-XL

None

Corresponding UI Option for System Connectivity Manager

None

STATE_WINS_OVER_DESIGN

This directive specifies that the property values in the state file override those found in the schematic. You should set this directive to “all” after you have run Packager-XL in the feedback mode.

If your design is hierarchical or has sized parts, you might not be able to backannotate all feedback data to the schematic. As a result, there can be conflicting property values in the schematic and the state file. In such cases, you should use the STATE_WINS_OVER_DESIGN directive to preserve the feedback data in the state file.

For example, if in an instance `1P`, the design has the `LOCATION` property with the value of `U1`, and the state file has the `LOCATION` property with a value of `U5`, the state file value of `U5` is used.

Syntax

```
STATE_WINS_OVER_DESIGN all| property [, property] ... ;
```

`all`	Retains all property values in the state file
`property`	Represents a property in the schematic.

The default is *none*. Packager-XL updates the state file with the value from the schematic.

Example

```
STATE_WINS_OVER_DESIGN all;
```

You need to be aware of the effect of the `STATE_WINS_OVER_DESIGN` directive when making changes to packaging assignments in the schematic.

In general, `STATE_WINS_OVER_DESIGN` should always be set to `all` and you should backannotate your schematic after each Packager-XL run. The following example illustrates this.

Instances `1P` and `2P` on the schematic have `LOCATION=U1`. The design is packaged and transferred to the layout where the `LOCATION U1` is changed to `U99`.

The layout is fed back and the packager state file now has `LOCATION=U99` for `1P` and `2P`. You now want to change the `LOCATION` for `2P` to `U98` by editing the schematic. You want the other assignments to maintain the packaging in the layout (that is, instance `1P` should have `LOCATION=U99`).

Two scenarios are possible:

- You first backannotate the schematic by running the Design Entry `Backannotate` command with the `pstback.dat` file. You then edit the schematic and change the `LOCATION` on `2P` to `U98`.
- You skip the backannotation and simply edit the schematic and change the `LOCATION` on `2P` to `U98`.

In both cases, you then run Packager-XL with `STATE_WINS_OVER_DESIGN` '`OFF`' to pick up the schematic change to `2P`. If you look at the resulting packaging, the schematic change to `LOCATION=U98` on `2P` is included correctly in both cases. However, the `LOCATION` value for `1P` is `U99` in the first case and `U1` in the second case. This is because you did not backannotate in the second case. The old packaging information in the schematic took precedence over the value in the state file (probably not what you intended).

STATE_WINS_OVER_LAYOUT

This directive specifies whether the feedback properties from the layout system override property values in the state file. However, if there is no value for the directive in the state file, the feedback value is used.

The `STATE_WINS_OVER_LAYOUT` directive applies only when feedback is allowed, and has no effect when the `NO_FEEDBACK` directive is used.

Syntax

```
STATE_WINS_OVER_LAYOUT all|property~
[,property] ... ;
```

`all`	Retains all property values in the state file.
`property`	Represents a property in the schematic.

The default value of the `STATE_WINS_OVER_LAYOUT` directive is *none*. That is, feedback properties are retained in the state file.

Example

```
STATE_WINS_OVER_LAYOUT all;
```

STOP_PACKAGE_ON_SCHEMATIC_ERROR

When this directive is set to 'ON', it stops Packager-XL (Export Physical) from proceeding if it finds graphical errors in the schematic. The default value of the `stop_package_on_schematic_error` directive is `OFF`.

Syntax

```
stop_package_on_schematic_error 'ON'|'OFF'
```

STOP_PST_GEN_ON_PTF_MISMATCH

This directive specifies whether `pst*` files will be generated in case of a PTF (part table file) mismatch. When you set the value of this directive to ‘`ON`’, the `pst*` files will not be generated if:

- The key properties of a part instance on the schematic do not match the values in the PTF.
- The PTFs are in the specified directory path, but the part is added in logical mode.

The default value of the `STOP_PST_GEN_ON_PTF_MISMATCH` directive is `ON`.

Syntax

```
STOP_PST_GEN_ON_PTF_MISMATCH 'ON'|'OFF'
```

Example

```
STOP_PST_GEN_ON_PTF_MISMATCH 'ON'
```

See also

PTF_MISMATCH_EXCLUDE_INJ_PROP

STRICT_PACKAGE_PROP

The STRICT_PACKAGE_PROP directive is used in Allegro System Capture and Packager-XL. This directive specifies that the instances with package properties cannot be packaged together with the instances without package properties.

The STRICT_PACKAGE_PROP directive works with the `PACKAGE_PROP` directive to further restrict packaging of schematic instances. Packager-XL does not package together any instances that have different values for the same package property. However, if spare slots are available, instances without package properties can be added. Packager-XL does not display any warnings or error messages if properties are overloaded or assigned to too many instances.

The STRICT_PACKAGE_PROP directive is specified in the `START_PKGRXL...END_PKGRXL` and `START_CANVAS...END_CANVAS` sections. Packager-XL reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

`STRICT_PACKAGE_PROP` *`property`* `[,`*`property`*`] ...;`

where

`property`	is a property in the schematic.

The default value for the `STRICT_PACKAGE_PROP` directive is *none*.

Example

```
STRICT_PACKAGE_PROP group subdesign_suffix;
```

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for Design Entry HDL

None

SUPPRESS

This directive is used to suppress specific warning messages.

Syntax

```
SUPPRESS number [, number ]...;
```

number	Specifies message numbers for specific warnings.

The default option for this directive is blank.

Example

```
SUPPRESS 1032;
```

SD_PREFIX_SEPARATOR

The SD_PREFIX_SEPARATOR directive is used in Allegro System Capture and System Connectivity Manager. You can set this directive if you want a different character to be used other than the default character underscore (_), as a prefix separator for reference designators of components in blocks.

For example, add a block named `MEMORY`, which has a component with the reference designator `U1` and add `A` as the prefix to the reference designator. In this case, System Connectivity Manager automatically updates the reference designator of the component in the block as `A_U1`. If you specify a hyphen (`-`) as a value of this directive, the reference designator is updated as `A-U1` instead of `A_U1`.

The SD_PREFIX_SEPARATOR directive is specified in the `START_DESIGNSTUDIO...END_DESIGNSTUDIO` and `START_CANVAS...END_CANVAS` sections. System Connectivity reads the directive value from the `START_PKGRXL...END_PKGRXL` section when launched from Design Entry HDL or System Connectivity Manager and reads the directive value from the `START_CANVAS...END_CANVAS` section when launched from Allegro System Capture.

Syntax

```
SD_PREFIX_SEPARATOR '<character>'
```

where

character is the prefix for renaming reference designators

Valid characters are: plus (+), underscore (_), hyphen (-), equals (=). If you do not want a prefix separator, you can specify an empty string `''`. An empty string as a directive value can only be specified in the `.cpm` file; it cannot be specified in the user interface.

Note: If you use a blank space or an exclamation mark (!) instead of the default separator underscore (_), System Connectivity Manager will ignore it and retain the default separator.

Example

```
SD_PREFIX_SEPARATOR ''
```

After packaging, the reference designator is updated as `AU1` instead of `A_U1`.

Corresponding UI Option for Allegro System Capture

None

Corresponding UI Option for System Connectivity Manager

Project — Settings — Packager — Block Prefix Separator

SHOW_ANALYZE_DIALOG

Use this directive to specify if the Analyze dialog box is to be displayed when you perform a Global Replace operation.

Syntax

```
show_analyze_dialog 'TRUE'|'FALSE'
```

Example

```
show_analyze_dialog 'TRUE'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Component Replace — Don't Show this dialog again, if everything is successful

See Also

- PRESERVE_BYPASS
- PRESERVE_CONN
- PRESERVE_PINPAIR
- PRESERVE_PWRGRP
- PRESERVE_REFDES
- PRESERVE_TERMINATION
- PRESERVE_USERPROP

SUPPORTLIBRARYDEFINEDDIFFPAIR

Use this directive to specify if library defined-differential pairs are supported in System Connectivity Manager.

Syntax

```
SupportLibraryDefinedDiffPair <value>
```

The valid values are: ON or OFF; 0 or 1

Example

```
SupportLibraryDefinedDiffPair '1'
```

Corresponding UI Option for System Connectivity Manager

None

Symbol_Length

The `Symbol_Length` directive specifies in grid units the minimum length of the default symbol created using the *Generate Symbol(s)* option.

Syntax

```
Symbol_Length '<value>'
```

Example

```
Symbol_Length '10'
```

Corresponding UI Option for Part Developer

Height field in the *Minimum Size (In Grid Units)* group box on *Tools –Setup – Symbol*

Symbol_OutLine

The `Symbol_OutLine` directive specifies if the outline of the symbol created using Generate Symbol(s) should be thin or thick.

Syntax

```
Symbol_OutLine '<line_style>'
```

The following line styles are supported:

- Thin
- Thick

Example

```
Symbol_OutLine 'Thin'
```

Corresponding UI Option for Part Developer

Symbol Outline list box on Tools - Setup - Symbol

Symbol_PinShape_Dot_Size

The `Symbol_PinShape_Dot_Size` directive defines the size of the circle in Line-Dot and Line-Dot-Clock pin shapes.

Syntax

```
Symbol_PinShape_Dot_Size '<value>'
```

Example

```
Symbol_PinShape_Dot_Size '26'
```

Corresponding UI Option for Part Developer

None

Symbol_Pintext_LeftRight_XOffset

The `Symbol_Pintext_LeftRight_XOffset` directive defines the default placement of pin text for left and right pins on the x axis.

Syntax

```
Symbol_Pintext_LeftRight_XOffset '<value>'
```

Example

```
Symbol_Pintext_LeftRight_XOffset '10'
```

Corresponding UI Option for Part Developer

None

Symbol_Pintext_LeftRight_YOffset

The `Symbol_Pintext_LeftRight_YOffset` directive defines the default placement of pin text for left and right pins on the y axis.

Syntax

```
Symbol_Pintext_LeftRight_YOffset '<value>'
```

Example

```
Symbol_Pintext_LeftRight_YOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_Pintext_TopBottom_XOffset

The `Symbol_Pintext_TopBottom_XOffset` directive defines the default placement of pin text for top and bottom pins on the x axis.

Syntax

```
Symbol_Pintext_TopBottom_XOffset '<value>'
```

Example

```
Symbol_Pintext_TopBottom_XOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_Pintext_TopBottom_YOffset

The `Symbol_Pintext_TopBottom_YOffset` directive defines the default placement of pin text for top and bottom pins on the y axis.

Syntax

```
Symbol_Pintext_TopBottom_YOffset '<value>'
```

Example

```
Symbol_Pintext_TopBottom_YOffset '10'
```

Corresponding UI Option for Part Developer

None

Symbol_PN_LeftRight_XOffset

The `Symbol_PN_LeftRight_XOffset` directive defines the offset of the $PN property associated with left and right pins from the connection point on the x axis The $PN property is moved on the x axis toward the symbol boundary. Therefore, for left pins, $PN moves right and for right pins, $PN moves left.

Syntax

```
Symbol_PN_LeftRight_XOffset '<value>'
```

Example

```
Symbol_PN_LeftRight_XOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_PN_LeftRight_YOffset

The `Symbol_PN_LeftRight_YOffset` directive defines the offset of the $PN property associated with left and right pins from the connection point in the positive direction on the y axis.

Syntax

```
Symbol_PN_LeftRight_YOffset '<value>'
```

Example

```
Symbol_PN_LeftRight_YOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_PN_TopBottom_XOffset

The `Symbol_PN_TopBottom_XOffset` directive defines the offset of the $PN property associated with top and bottom pins from the connection point in the negative direction on the x axis.

Syntax

```
Symbol_PN_TopBottom_XOffset '<value>'
```

Example

```
Symbol_PN_TopBottom_XOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_PN_TopBottom_YOffset

The `Symbol_PN_TopBottom_YOffset` directive defines the offset of the $PN property associated with top and bottom pins from the connection point on the y axis. The $PN property is moved on the y axis toward the symbol boundary. Therefore, for top pins, $PN moves downward and for bottom pins, $PN moves upward.

Syntax

```
Symbol_PN_TopBottom_YOffset '<value>'
```

Example

```
Symbol_PN_TopBottom_YOffset '0'
```

Corresponding UI Option for Part Developer

None

Symbol_SymSheetSize

The `Symbol_SymSheetSize directive` specifies the default sheet size of symbols.

Syntax

```
Symbol_SymSheetSize '<A>'
```

Example

```
Symbol_SymSheetSize 'A'
```

Corresponding UI Option for Part Developer

Sheet Size list box on *Tools – Setup – Symbol*

Symbol_Units

The `Symbol_Units` directive specifies the default unit to be used for symbol settings.

Syntax

```
Symbol_Units '<unit>'
```

The following units are supported:

- Inches
- Fractions
- Metrics

Example

```
Symbol_Units 'Inches'
```

Corresponding UI Option for Part Developer

System Unit list box on *Tools – Setup – Symbol*

Symbol_Width

The `Symbol_Width` directive specifies in grid units the minimum width of the default symbol created using Generate Symbol(s).

Syntax

```
Symbol_Width '<value>'
```

Example

```
Symbol_Width '10'
```

Corresponding UI Option for Part Developer

Width field in the *Minimum Size (In Grid Units)* group box on *Tools – Setup – Symbol*

Search_Attr_Name

Defines all the search attributes that should be present in the *Attributes* tab of Part Information Manager. The attributes appear in the order as they are set up, and follow the same order in the search criteria drop-down box.

Important

This directive is applicable only to the cache mode.

Syntax

```
Search_Attr_Name '<attribute 1>' '<attribute 2>' '<attribute3 >' ... ...
     '<attribute N>'
```

Example

```
Search_Attr_Name 'DESCRIPTION' 'VALUE' 'TOL' 'CURRENT' 'JEDEC_TYPE' 'PART_NUMBER'
     'POWER' 'COLOR' 'TYPE' 'TEMPERATURE' 'SPEED' 'VENDOR' 'ORIENTATION'
     'ALT_SYMBOLS' 'PRICE' 'AREA' 'HEIGHT' 'PIN_COUNT' 'STATUS' 'PACK_TYPE'
```

Corresponding UI Option

Configuration – Setup - Search Options

Search_Attr_Type

Defines the data type, string or numeric, for each of the search attributes defined under the *Attributes* tab. The attribute types are listed in the order the attributes are listed in the `Search_Attr_Name` directive.

Syntax

```
Search_Attr_Type '<type 1>' '<type 2>' '<type 3>' ... ... '<type N>'
```

Example

```
Search_Attr_Type 'String' 'Numeric' 'Numeric' 'Numeric' 'String' 'String'
     'Numeric' 'String' 'String' 'Numeric' 'Numeric' 'String' 'String' 'String'
     'Numeric' 'Numeric' 'Numeric' 'Numeric' 'String' 'String'
```

Important

This directive is applicable only to the cache mode.

Corresponding UI Option

Configuration – Setup - Search Options

Search_Result_Type

Defines if metadata or PTF should be shown in the search results. The absence of this directive in the `.cpm` file means that the PTF will be displayed.

Syntax

```
Search_Result_Type 'metadata'
```

Example

```
Search_Result_Type 'metadata'
```

Corresponding UI Option

Configuration – Setup – Show Metadata

Search_Tab_Order

Controls the order in which the `Relations`, `Properties`, and `Attributes` nodes are displayed in the right panel when any entry is selected in the left tree.

Syntax

```
Search_Tab_Order '<node 1>' '<node 2>' '<node 3>'
```

Example

```
Search_Tab_Order 'Relations' 'Properties' 'Attributes'
```

Corresponding UI Option

Configuration – Setup – Search – Search Tab Order

Search_Tolerance

Defines the search tolerance limit for search results.

Important

This directive is applicable only to the database mode.

Syntax

```
Search_Tolerance 'value'
```

Example

```
Search_Tolerance '21'
```

Corresponding UI Option

Configuration – Setup – Search – Tolerance

Search_Tree_Order

Lists the order of the tree nodes (`Browse Libraries`, `Libraries`, and `Classifications`) that appear in the left pane.

Syntax

```
Search_Tree_Order '<node 1>' 'Libraries' 'Classifications'
```

Example

```
Search_Tree_Order 'Browse Libraries' 'Libraries' 'Classifications'
```

Corresponding UI Option

Configuration – Setup – General Configuration

Show_Cell

Defines whether the name of the cell appears beside the Part Name column in the search results pane.

Syntax

```
Show_Cell 'True'| 'False'
```

Example

```
Show_Cell 'True'
```

Corresponding UI Option

Configuration – Setup – Search – Show Cell

Show_Library

Defines whether the name of the library appears beside the Part Name column in the search results pane.

Syntax

```
Show_Library 'True'| 'False'
```

Example

```
Show_Library 'False'
```

Corresponding UI Option

Configuration – Setup – Search – Show Library

Single_Click_Details

Defines whether the *Details* tab appears on single-clicking a PTF row in the Search Results pane in Part Information Manager.

Syntax

```
Single_Click_Details 'True'| 'False'
```

Example

```
Single_Click_Details 'True'
```

Corresponding UI Option

Configuration – Setup – Details – Details on Single Click

sync_properties

Specifies a preference for a key or injected property to decide which mismatched part should be auto-fixed.

For example, in the following figure, there is a mismatch in `Key2`:

	Key1	Key2	Inj1	Inj2
Local	10k	5%	def	cell
Reference	10k	7%	def	cell

Assume that the value of the `sync_properties` directive is `key1`, and in the reference PTF row, the value for `key2` has changed from `5%` to `7%`. In this case, Library Revision Manager displays the row as `Autofixable` in the *Cell/Block Details* pane, and displays a grid if you select the *Show Differences* pop-up menu option.

Syntax

```
sync_properties '<property 1> <property 2> <property 3> <property 4>'
```

Example

```
sync_properties '<KEY1> <KEY2> <INJ1> <INJ2>'
```

See Also

auto_update_minor_ptf

auto_fix_ptf

SAVE_TCL_IN_PROJECT

Set this directive to TRUE if you want the TCL Journal file to be saved in the project's temporary directory.

Syntax

```
SAVE_TCL_IN_PROJECT 'TRUE'|'FALSE'
```

The default value is TRUE.

Example

```
SAVE_TCL_IN_PROJECT 'TRUE'
```

SDA_CAPTURE_SPECIAL_LIB

Specify the name of the library that contains special parts to be used for the OrCAD Capture Library flow.

Syntax

```
SDA_CAPTURE_SPECIAL_LIB '<library_name>'
```

where,

`library_name`	Specify the name of the library.

Example

```
SDA_CAPTURE_SPECIAL_LIB 'orcadlib'
```

STUB_LENGTH

Defines the default length of the wire stubs drawn using the *Draw Stub* menu command or the *drawStubs* Tcl command.

Syntax

```
STUB_LENGTH '<value>'
```

value	Indicates the length of a wire stub in grid units. The default value is 10.

Example

```
STUB_LENGTH 'YES'
```

SHOW_UNCONNECTED_PINS

Toggles between showing and hiding unconnected pins on components.

Syntax

```
SHOW_UNCONNECTED_PINS 'ON'|'OFF'
```

Example

```
SHOW_UNCONNECTED_PINS 'ON'
```

SHOW_NET_HOTSPOTS

Set this directive to 1 to enable the display of an indicator or hotspot on unconnected nets.

Syntax

```
SHOW_NET_HOTSPOTS '0'|'1'
```

0	Disable the display of an indicator or hotspot on unconnected nets. This is the default value.
1	Enable the display of an indicator or hotspot on unconnected nets.

Example

```
SHOW_NET_HOTSPOTS '0'
```

SHOW_NET_NAME_DIALOG

Toggles between showing and hiding the net name dialog at the top-left corner of the page while a wire or a bus is being drawn on the canvas.

Syntax

```
SHOW_NET_NAME_DIALOG 'ON'|'OFF'
```

YES	Display the net name dialog. This is the default value.
NO	Hides the net name dialog.

Example

```
SHOW_NET_NAME_DIALOG 'ON'
```

SHOW_PART_MANAGER_ON_START

When this directive is set to TRUE, Part Manager automatically compares and displays the differences between design cache and reference libraries when design is loaded so that you can review the differences.

Syntax

```
SHOW_PART_MANAGER_ON_START 'TRUE'|'FALSE'
```

The default value is FALSE.

Example

```
SHOW_PART_MANAGER_ON_START 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Component - Run Part Manager on Project Load

19

T Directives

This chapter lists the CPM directives that start with `T` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

TAP_SYMBOL

Specifies the tap symbol to be used in a schematic

Syntax

```
TAP_SYMBOL '<tap_symbol>'
```

Example

```
TAP_SYMBOL 'CTAP'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Wires — Tap Symbol

TESTPAD_FOOTPRINT_NAME_PATTERN

Defines the footprint patterns to identify components as test pads. If test pads are not connected to user-defined named nets, a violation is reported.

This directive is used for the following *Mechanical Checks* audit rules:

- `Test pads not connected to a net having user-defined name`
- `Test pads not present`

Syntax

```
TESTPAD_FOOTPRINT_NAME_PATTERN '<pattern>'
```

Example

```
TESTPAD_FOOTPRINT_NAME_PATTERN 'PCB_TPAD'
```

Corresponding UI Option in Allegro System Capture

Design Integrity – Configure – Schematic Audit Settings – Rule

TEXT_EDITOR

Specifies the text editor that Design Entry HDL and Allegro System Capture open for certain functions.

For Allegro System Capture:

- Place this directive in the `Canvas` section
- Ensure that the editor supports blocking mode

  That is, when the editor is launched in the foreground, it blocks the current process, and System Capture waits for the process to complete. For example, to launch Notepad++ in blocking mode, pass the `-nosession` parameter in the command line, as shown in the example.

Syntax

```
TEXT_EDITOR '<path>\\<application> | -<blocking_parameter>| '
```

where,

application	The absolute path of the application to launch
blocking_parameter	The argument for forcing System Capture to wait

Examples

- System Capture

  ```
  START_CANVAS
  ```

  ```
  text_editor '"C:\\Program Files\\Notepad++\\notepad++.exe" -
  nosession'
  ```

  Remember to use the escape character for the backslash.

- DE-HDL

  ```
  TEXT_EDITOR 'notepad'
  ```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Text — Text Change Editor

Corresponding UI Option for Allegro System Capture

None

TEXT_JUSTIFICATION

Justifies text Left, Center, or Right.

Syntax

```
TEXT_JUSTIFICATION 'LEFT'|'RIGHT'|'CENTER'
```

Example

```
TEXT_JUSTIFICATION 'LEFT'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Text — Justification

See Also

TEXT_SIZE

TEXT_SIZE

Specifies the size of text (property name, property value, signal name, URL, or note) in the plotted schematic.

Syntax

```
TEXT_SIZE '<value>'
```

where

value	any number greater than 0.002. The default value is 0.082 inches.

Example

```
TEXT_SIZE '1.000'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Text page — Text — Size

See Also

TEXT_JUSTIFICATION

TOC_AUTO_SAVE

Automatically updates the Table of Contents (TOC) of an Allegro System Capture design when it is saved. Set this variable to `TRUE` in the `canvas` section of the cpm file or the site cpm file. This directive eliminates the need to manually refresh the TOC by right-clicking the TOC and choosing *Re-evaluate TOC*.

Syntax

```
TOC_AUTO_SAVE 'TRUE'|'FALSE'
```

Example

```
TOC_AUTO_SAVE 'TRUE'
```

TOC_DISPLAY_SHEET_RANGE

Displays a page range in the Table of Contents (TOC) of a design. Setting this variable to ON is useful in designs which have multiple rows of sheets with the same page title. Instead of having multiple entries for such rows in the TOC, you can show such entries in a single row with the sheet numbers as a range in the Sheet Number column.

Syntax

```
TOC_DISPLAY_SHEET_RANGE 'ON'|'OFF'
```

Example

```
TOC_DISPLAY_SHEET_RANGE 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

TOC_ROW_SPACING_MULTIPLIER

TOC_ROW_SPACING_MULTIPLIER

Controls the line spacing between rows of the Table of Contents of a design.

Syntax

```
TOC_ROW_SPACING_MULTIPLIER '<value>'
```

Example

```
TOC_ROW_SPACING_MULTIPLIER '1'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

TOC_DISPLAY_SHEET_RANGE

TOC_PAGE_DEFAULT_SIZE

This directive defines the page border to be used for the TOC page. This page border can be different from the page border being used for schematic sheets.

Syntax

```
TOC_PAGE_DEFAULT_SIZE 'A'|'B'|'C'|'D'|'E'
```

Following table describes the dimensions of the available pages:

A	Height: 7.2 inches
	Width: 9.7 inches
B	Height: 9.2 inches
	Width: 15.2 inches
C	Height: 15.2 inches
	Width: 20.2 inches
D	Height: 32.2 inches
	Width: 20.2 inches
E	Height: 42.2 inches
	Width: 32.7 inches

Example

```
TOC_PAGE_DEFAULT_SIZE 'C'
```

TABLE_ALTERNATE_FILL_COLOR

Sets the fill color of alternate rows of tables.

Syntax

```
TABLE_ALTERNATE_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color for alternate rows of tables. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#D2DDEF'`.

Examples

```
TABLE_ALTERNATE_FILL_COLOR '#D2DDEF'
```

TABLE_FILL_COLOR

Sets the default fill color of alternate rows of tables.

Note: This directive sets default fill color of alternate rows other than the row colors filed by the TABLE_ALTERNATE_FILL_COLOR directive.

Syntax

```
TABLE_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color of alternate rows of the tables. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#E8EFF7'`.

Examples

```
TABLE_FILL_COLOR '#E8EFF7'
```

TABLE_FILL_STYLE

Sets the style that is used to fill table rows.

Syntax

```
TABLE_FILL_STYLE '<fill_style>'
```

Where

`fill_style`	The default fill style for table rows. Valid values: `none` (default), `solid`.

Examples

```
TABLE_FILL_STYLE 'none'
TABLE_FILL_STYLE 'solid'
```

TABLE_HEADER_FILL_COLOR

Sets the default fill color of header rows of the tables.

Syntax

```
TABLE_HEADER_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color of header rows of the tables. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#5998D2'` (Blue).

Examples

```
TABLE_HEADER_FILL_COLOR '#5998D2'
```

TABLE_LINE_CAP_STYLE

A line cap style defines the ending of a line. This directive defines the line cap style for tables.

Syntax

```
TABLE_LINE_CAP_STYLE '<cap_style>'
```

Where

`cap_style`	The default line cap style for tables. Valid values: `round-cap` (default), `square-cap`, `diamond-cap`, `arrowhead-cap`.

Examples

```
TABLE_LINE_CAP_STYLE 'round-cap'
TABLE_LINE_CAP_STYLE 'diamond-cap'
```

TABLE_LINE_COLOR

Sets the line color for tables.

Syntax

```
TABLE_LINE_COLOR '<line_color>'
```

Where

`line_color`	The default line color for tables. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#003C77'` (Blue).

Examples

```
TABLE_LINE_COLOR '#003C77'
```

TABLE_LINE_JOIN_STYLE

Sets the type of corner created when two lines of a table join or meet.

Syntax

```
TABLE_LINE_JOIN_STYLE '<join_style>'
```

Where

`join_style`	The default line join style for table lines. Valid values: `miter-join`, `round-join` (default), `bevel-join`.

Examples

```
TABLE_LINE_JOIN_STYLE 'round-join'
TABLE_LINE_JOIN_STYLE 'bevel-join'
```

TABLE_LINE_STYLE

Specifies the default line style for tables.

Syntax

```
TABLE_LINE_STYLE '<line_style>'
```

Where

`line_style`	The default line style for tables. Valid values: `solid` (default), `dash`, `dot`, `dash-dot`, `dash-dot-dot`.

Examples

```
TABLE_LINE_STYLE 'dot'
TABLE_LINE_STYLE 'solid'
```

TABLE_LINE_WIDTH

Sets the default line width for tables.

Syntax

```
TABLE_LINE_WIDTH '<line_width>'
```

Where

`line_width`	The default line width for tables. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
TABLE_LINE_WIDTH '10'
```

TABLE_TEXT_FONT_BOLD

Specifies whether the text in tables appear in **bold** face.

Syntax

```
TABLE_TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
TABLE_TEXT_FONT_BOLD 'FALSE'
```

TABLE_TEXT_FONT_COLOR

Sets the font color of the text in tables.

Syntax

```
TABLE_TEXT_FONT_COLOR '<font_color>'
```

Where

`font_color`	The default font color of the text in tables. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#000000'` (Black).

Examples

```
TABLE_TEXT_FONT_COLOR '#000000'
```

TABLE_TEXT_FONT_ITALIC

Specifies whether the text in tables appear in *italics*.

Syntax

```
TABLE_TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
TABLE_TEXT_FONT_ITALIC 'FALSE'
```

TABLE_TEXT_FONT_NAME

Sets the font face used to display the text in tables.

Syntax

```
TABLE_TEXT_FONT_NAME '<font_name>'
```

Where

font_name	The default font face used to display the text in tables. The value can be any of the system-supported fonts. The default value is `'ARIAL'`.

Examples

```
TABLE_TEXT_FONT_NAME 'ARIAL'
```

TABLE_TEXT_FONT_SIZE

Sets the font size of the text in tables.

Syntax

```
TABLE_TEXT_FONT_SIZE '<font_size>'
```

Where

`font_size`	The default font size of the text in tables. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
TABLE_TEXT_FONT_SIZE '10'
```

TABLE_TEXT_FONT_UNDERLINE

Specifies whether the text in tables appear underlined.

Syntax

```
TABLE_TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
TABLE_TEXT_FONT_UNDERLINE 'FALSE'
```

TABLE_TEXT_MARGIN

Sets the margin or the space around text in tables.

Syntax

```
TABLE_TEXT_MARGIN '<value>'
```

Where

`value`	Any positive integer value. The default value is `1`.

Examples

```
TABLE_TEXT_MARGIN '1'
TABLE_TEXT_MARGIN '0.5'
```

Note:

TEXT_FONT_NAME

Sets the default font face used to display the text on canvas.

Syntax

```
TEXT_FONT_NAME '<font_name>'
```

font_name	The default font face used to display the text on canvas. The value can be any of the system-supported fonts.

Examples

```
TEXT_FONT_NAME 'Arial'
TEXT_FONT_NAME 'Helvetica'
```

TEXT_FONT_SIZE

Sets the default size of the text on canvas.

Syntax

```
TEXT_FONT_SIZE '<font_size>'
```

font_size	The default font size used to display the text.

Examples

```
TEXT_FONT_SIZE '5'
```

TEXT_FONT_ITALIC

Specifies whether the text appears in *italics*.

Syntax

```
TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

Examples

```
TEXT_FONT_ITALIC 'FALSE'
TEXT_FONT_ITALIC 'TRUE'
```

TEXT_FONT_BOLD

Specifies whether the text appears in **bold** face.

Syntax

```
TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

Examples

```
TEXT_FONT_BOLD 'TRUE'
TEXT_FONT_BOLD 'FALSE'
```

TEXT_FONT_UNDERLINE

Specifies whether the text appears underlined.

Syntax

```
TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

Examples

```
TEXT_FONT_UNDERLINE 'TRUE'
TEXT_FONT_UNDERLINE 'FALSE'
```

TEXT_FONT_COLOR

Sets the color of text on the canvas.

Syntax

```
TEXT_FONT_COLOR 'font_color'
```

font_color	The value can be expressed as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red.

Examples

```
TEXT_FONT_COLOR '#000000'
```

TEXT_MARGIN

Sets the margin or the space around the text object.

Syntax

```
TEXT_MARGIN '<value>'
```

value	Any positive integer value.

Examples

```
TEXT_MARGIN '0'
TEXT_MARGIN '0.5'
```

TEXT_WORD_WRAP

Controls whether the lines of text break within words and wrap onto the next line, to prevent text overflow when a string is too long to fit into the containing box.

Syntax

```
TEXT_WORD_WRAP 'TRUE'|'FALSE'
```

Examples

```
TEXT_WORD_WRAP 'TRUE'
TEXT_WORD_WRAP 'FALSE'
```

20

U Directives

This chapter lists the CPM directives that start with `U` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

UNNAMED_NET_GEN

This directive is automatically set if the design is to be used for analog simulations.

Syntax

```
UNNAMED_NET_GEN 'ON'|'OFF'
```

Example

```
UNNAMED_NET_GEN 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

UNNAMED_NET_USING_FULL_PINNAME

When this directive is ON, DE-HDL generates unnamed nets using the entire pin name and not simply the base pin name.

Syntax

```
UNNAMED_NET_USING_FULL_PINNAME 'ON'|'OFF'
```

Example

```
UNNAMED_NET_USING_FULL_PINNAME 'OFF'
```

Corresponding UI Option for Allegro Design Entry HDL

None

UNIQUE_FEATURE

Displays each column of a BOM report with unique values. If you do not set this directive, the unique listing is displayed only for the reference designator column provided you have selected the *Unique* option under RefDes in the BOM-HDL dialog.

Value

```
UNIQUE_FEATURE 'ON'|'OFF'
```

Example

```
UNIQUE_FEATURE 'ON'
```

Corresponding UI Option for Project Manager

Tools — Packager Utilities — Bill of Materials — BOM-HDL dialog box — Customize button — Customize Template dialog box — Report Parameters page — RefDes — Unique option button

USE_VECTOR_NOTATION

This directive specifies that individual bits for vector signals will always be saved within angular braces in the `pstxnet.dat` file. For example, if you have a bus `DATA <7..0>`, then the individual bits will be represented as `DATA <7>`, `DATA <6>`, ,and `DATA <0>`.

Syntax

```
USE_VECTOR_NOTATION 'off' | 'on'
```

By default, the directive is set to `ON`. When set to OFF, the individual vector bits are not stored within angular braces.

Note: Each time you make a change in the `USE_VECTOR_NOTATION` directive, you need to repackage the design. However, avoid making frequent changes in the representation of buses through the use of this directive.

USE_LIBRARY_PPT

This directive specifies that cell-level PPTs are to be used in addition to any PPTs you specify using the PPT directive. Cell-level PPTs are part table files that have a `.ptf` extension in the HDL environment and are located at the same level of the library hierarchy as a *chips* file.

You can control the use of individual PPTs at the cell level with either the `EXCLUDE_PPT` or `INCLUDE_PPT` directives.

You must add the `USE_LIBRARY_PPT` directive to the Part Table section of the Project Setup dialog.

Syntax

```
USE_LIBRARY_PPT 'on'|'off';
```

The default value of the `USE_LIBRARY_PPT` directive is `;`.

Example

```
USE_LIBRARY_PPT 'off';
```

USE_SUBDESIGN

The USE_SUBDESIGN directive reads the subdesign state file only once to get packaging information for the instances of the subdesigns that were not previously packaged. The information is then incorporated into the design state file.

Any further changes to the subdesign are not propagated to the packaged subdesign instances. The FORCE_SUBDESIGN directive is recommended instead of the USE_SUBDESIGN directive.

In the feedback mode, instances that have this directive applied on them take on a new value that PCB Editor might have assigned to them.

Syntax

```
USE_SUBDESIGN subdesign [,subdesign] ...;
```

subdesign	A subdesign for which a subdesign state file exists. The subdesign name is the same as the drawing name used in Design Entry.

The default value for the USE_SUBDESIGN directive is none.

Example

```
USE_SUBDESIGN COUNTER;
```

USE_OFFPAGE

Use this directive to set, if offpage connectors are used in the generated document schematic to connect signals across schematic pages.

Syntax

```
use_offpage <0 or 1>
```

Example

```
use_offpage '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Symbols — Add Offpage Symbols.

USE_POWER_SYMBOLS

Use this directive to set that in the generated document schematic, specified power symbols will be used to indicate power nets based on power_symbols directive.

Syntax

```
use_power_symbols <0 or 1>
```

Example

```
use_power_symbols '0'
```

Corresponding UI Option for System Connectivity Manager

Project — Settings — Document Schematic Generation — Power— Use Power Symbols.

UPPERCASE_SIGNAL_NAMES

When this directive is set to TRUE, all signal names appear in uppercase. If you enter lowercase characters, the tool converts them to uppercase.

Syntax

```
UPPERCASE_SIGNAL_NAMES 'TRUE'|'FALSE'
```

The default value is `TRUE`.

Example

```
UPPERCASE_SIGNAL_NAMES 'TRUE'
```

21

V Directives

This chapter lists the CPM directives that start with `V` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

VAR_OVERLAY_PROPS_VISIBLE

This directive allows you to specify the names of properties that should be made visible on the variant schematic when the property value has changed with relation to the value in the base schematic. By default, the value for this directive is set to ALL.

When you modify a component in a variant, additional properties might be added to the schematic because of changes in the selected part. Because these properties might not have placeholders on the symbol, they are not displayed on the schematic canvas. If you want these properties to be displayed on the canvas, you can use the `VAR_OVERLAY_PROPS_VISIBLE` directive in the `START_CONCEPTHDL` section of the `.cpm` file to specify which properties should be displayed on the canvas.

You can define three values for this directive:

- NONE - only variant properties, that is, `VARIANT = *`, will be visible on the schematic
- ALL - all the properties of the modified component that have changed compared to the base instance will be overlaid on the schematic in the variant view. These properties will be displayed in the color defined for changed properties. You can configure the changed property color using the Variant Specific Property option in the Design Entry HDL Options dialog.
- In the third case, you can define the properties (for example, Part Number) that you want displayed on the schematic as follows:

  `VAR_OVERLAY_PROPS_VISIBLE 'Property1' 'Property2' 'Property3'`

  In this case, the schematic will display Property1, Property2, Property3 in the changed property color in the variant view for the variant component.

  Note that the variant properties, that is, `VARIANT=*` will always be visible on the variant component in the schematic in the variant view.

Syntax

```
VAR_OVERLAY_PROPS_VISIBLE 'ALL'|'NONE'|'<modified properties>'
```

Example

```
VAR_OVERLAY_PROPS_VISIBLE 'PART_NUMBER''TOLERANCE'
```

Corresponding UI Option for Allegro Design Entry HDL

None

VAR_COMP_BOM_PROPS

When defined, this directive allows you to modify the default comparison BOM report. For example, you can include user-defined properties for each changed component in each variant of a design comparison BOM report. When this directive is ON, the report generated also automatically includes alternate rows, as well as rows for the preferred values of components.

By default, the part-number based comparison BOM report provides a part number-based comparison between the components of the base schematic and all the variants. While generating the comparison BOM report, only the preferred values of components and alternate groups are considered.

To modify the default comparison BOM report, do the following:

1. In the `START_BOMHDL...END_BOMHDL` section of the `.cpm` file, specify the following directive:

   ```
   VAR_COMP_BOM_PROPS
   ```

   Note: When this directive is `ON`, the report generated also automatically includes alternate rows, as well as rows for the preferred values of components.

2. Specify the names of the properties that you want displayed in the report. In this example, we have `'NATION' 'COLOR'`.
3. Choose *Tools — Generate Reports* in Variant Editor.
4. Specify the template file path, the output file path, and the report format in BOM-HDL dialog box. You can retain the default selection.
5. Click the *Variant BOM* button to expand the variant options.
6. Enter the path to the variant field in the Variant File field.
7. Click the *Variant Comparison BOM* radio button and select the name of the variant.
8. Click the *Generate* button.

The comparison BOM report is displayed.

```
TITLE:                  Bill of Materials
DATE:                   10/30/2017
DESIGN:                 super_design
TEMPLATE:               //v17-2-250/share/cdssetup/template.bom
CALLOUT:                bom.callouts

 Ref      Base            DEMO_VARIANT
 Des
          Part Number     Part Number  Var Status  NATION  COLOR
======    ============    ========================================
  R1      CDN0005-01      CDN0005-01   Pref*       ?       ?
  U1      ?               ?            DNI         ?       ?

Common Components List :
 Ref      Part Number
 Des
======    ============
  R2      CDN0005-01
  R3      CDN0005-05
```

Value

`VAR_COMP_BOM_PROPS`

Example

`VAR_COMP_BOM_PROPS 'COLOR'`

Corresponding UI Option

None

VAR_REPLACE_BY_PROP

Important

This feature is available only when connected to a remote Pulse server.

You can now replace components in variants in Allegro System Capture designs with placeholders. Preferred parts are no longer restricted to parts available in the project libraries. To enable this feature, add this directive in the `START_VARIANT` section of the *`project`* or `site.cpm` file. To specify the property to be used for the placeholder, use the `VAR_REPLACE_PROP` directive.

Syntax

```
VAR_REPLACE_BY_PROP 'ON|OFF'
```

Example

```
VAR_REPLACE_BY_PROP 'ON'
```

Corresponding UI Option for Allegro System Capture

None

See Also

VAR_REPLACE_PROP

VAR_REPLACE_PROP

Important

This feature is available only when connected to a remote Pulse server.

You can now replace components in variants in Allegro System Capture designs with placeholders. Preferred parts are no longer restricted to parts available in the project libraries. To enable this feature, add this directive in the `START_VARIANT` section of the *`project`* or `site.cpm` file. This directive requires the `VAR_REPLACE_BY_PROP` directive also to be set.

Syntax

```
VAR_REPLACE_PROP '<property name>'
```

Example

```
VAR_REPLACE_BY_PROP 'Part_Number'
```

Corresponding UI Option for Allegro System Capture

None

See Also

VAR_REPLACE_BY_PROP

VISIBLE

Displays or hides the selected layer.The value of this directive identifies which layers are visible in the published PDF document and which are not.

Syntax

```
VISIBLE '<decimal_value>'
```

Where *decimal value* determines which layers are visible on the published PDF document.

Example

```
VISIBLE '511'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — PDF page — General — Layers — Visible

See Also

EXPORT

PRINTLAYER

VIEW_PCB

The `VIEW_PCB` directive lets you set the path of the board (.brd) file to be used by PCB Editor, SI, and Design Sync. This directive is defined in the GLOBAL section of the project (.cpm) file.

If you have defined the `VIEW_PCB` directive, then the following occurs:

- PCB Editor and SI use the path specified by the `VIEW_PCB` directive to open the board file.
- When you run genfeedformat from Import Physical, the Import Physical browser uses the path specified by the `VIEW_PCB` directive to read the PCB Editor board file.
- When you run Netrev from Export Physical, the output board file is saved in the path set in the `VIEW_PCB` directive.

If the `VIEW_PCB` directive is not specified, the `physical` directory under the root design is used to read or store the board files.

To set the `VIEW_PCB` directive, do the following:

1. Choose the *Views* tabbed page in Project Setup.
2. The default entry in the *Physical* field is the `physical` view of the project. You can set the view name to any of the following:
   a. Any other view name present in the drop-down box.
   b. The path to the board file. This path can be a relative path.

Note: Cadence recommends that the path that you specify in the `VIEW_PCB` directive be the same as that specified in the `PHYSICAL_PATH` directive.

Viewers

Defines whether one or both viewers appear when you view part information. This can contain two values:

- `Symbol`
- `Footprint`

When you specify both, the first viewer you specify appears on the left-hand side of the *Details* pane.

Syntax

```
Viewers '<viewer 1>' ['<viewer 2>']
```

Example

```
Viewers 'Symbol' 'Footprint'
```

Corresponding UI Option

Configuration – Setup – Details – Symbol/Footprint Viewer

VARIANT_DNI_CROSS

Set this directive to ON if you want the DNI components to appear with a cross mark in the variant view.

Syntax

```
VAIRANT_DNI_CROSS 'ON'|'OFF'
```

Where

`ON`	The DNI components appear with a cross mark in the variant view.
`OFF`	The DNI components do not appear with a cross mark in the variant view.

Examples

```
VAIRANT_DNI_CROSS 'ON'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Show Cross on DNI

VARIANT_DNI_CROSS_COLOR

Sets the default color of the cross mark that appears when you set the status of a component as DNI.

Syntax

```
VARIANT_DNI_CROSS_COLOR '<cross_color>'
```

Where

`cross_color`	The default font color of the cross mark. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FF0000'` (Red).

Examples

```
VARIANT_DNI_CROSS_COLOR '#FCD054'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - DNI Cross Color

VARIANT_DNI_CROSS_LINE_WIDTH

Sets the default line width of the cross mark that appears when you set the status of a component as DNI.

Syntax

```
VARIANT_DNI_CROSS_LINE_WIDTH '<line_width>'
```

Where

`line_width`	The default line width of the cross mark. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
VARIANT_DNI_CROSS_LINE_WIDTH '10'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - DNI Line Width

VARIANT_EDITOR_DISPLAY_PROP_NAME

Sets the property name that is to be used by Variant Editor as key property.

Syntax

```
VARIANT_EDITOR_DISPLAY_PROP_NAME "<property_name>"
```

The default value is `PART_NUMBER`.

Example

```
VARIANT_EDITOR_DISPLAY_PROP_NAME "PART_NUMBER"
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Property Name corresponding to Part Number

VARIANT_INST_FILL_COLOR

Sets the default fill color for variant instances.

Syntax

```
VARIANT_INST_FILL_COLOR '<fill_color>'
```

Where

`fill_color`	The default fill color for variant instances. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FFFFFF'` (White).

Examples

```
VARIANT_INST_FILL_COLOR '#E8EFF7'
VARIANT_INST_FILL_COLOR 'WHITE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Fill Color

VARIANT_INST_FILL_STYLE

Sets the style that is used to fill variant instances.

Syntax

```
VARIANT_INST_FILL_STYLE '<fill_style>'
```

Where

`fill_style`	The default fill style for variant instances. Valid values: `none`, `solid` (default).

Examples

```
VARIANT_INST_FILL_STYLE 'none'
VARIANT_INST_FILL_STYLE 'solid'
```

VARIANT_INST_LINE_CAP_STYLE

A line cap style defines the ending of a line. This directive defines the line cap style for variant instances.

Syntax

```
VARIANT_INST_LINE_CAP_STYLE '<cap_style>'
```

Where

`cap_style`	The default line cap style for variant instances. Valid values: `round-cap` (default), `square-cap`, `diamond-cap`, `arrowhead-cap`.

Examples

```
VARIANT_INST_LINE_CAP_STYLE 'round-cap'
VARIANT_INST_LINE_CAP_STYLE 'diamond-cap'
```

VARIANT_INST_LINE_COLOR

Sets the line color for variant instances.

Syntax

```
VARIANT_INST_LINE_COLOR '<line_color>'
```

Where

`line_color`	The default line color for variant instances. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#FF0000'` (Red).

Examples

```
VARIANT_INST_LINE_COLOR '#003C77'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Line Color

VARIANT_INST_LINE_JOIN_STYLE

Sets the type of corner created when two lines of a variant instance join or meet.

Syntax

```
VARIANT_INST_LINE_JOIN_STYLE '<join_style>'
```

Where

`join_style`	The default line join style for variant instance lines. Valid values: `miter-join`, `round-join` (default), `bevel-join`.

Examples

```
VARIANT_INST_LINE_JOIN_STYLE 'round-join'
VARIANT_INST_LINE_JOIN_STYLE 'bevel-join'
```

VARIANT_INST_LINE_STYLE

Specifies the default line style for variant instances.

Syntax

```
VARIANT_INST_LINE_STYLE '<line_style>'
```

Where

`line_style`	The default line style for variant instances. Valid values: `solid` (default), `dash`, `dot`, `dash-dot`, `dash-dot-dot`.

Examples

```
VARIANT_INST_LINE_STYLE 'dot'
VARIANT_INST_LINE_STYLE 'solid'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Line Style

VARIANT_INST_LINE_WIDTH

Sets the default line width for variant instances.

Syntax

```
VARIANT_INST_LINE_WIDTH '<line_width>'
```

Where

`line_width`	The default line width for variant instances. The default value is `'2'`. Valid Range: 1 to 72

Examples

```
VARIANT_INST_LINE_WIDTH '10'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Line Width

VARIANT_INST_ITEM_OPACITY

Controls the opacity or transparency of variant instances on the canvas.

Syntax

```
VARIANT_INST_ITEM_OPACITY '<opacity_factor>'
```

Where

`opacity_factor`	Any number between 0 and 99. A lower value results in a more opaque variant instance. An opacity factor of 1 indicates that the variant instance is completely opaque, while a value of 99 indicates that the background of the variant instance is transparent. The default value is `'1'`.

Examples

```
VARIANT_INST_ITEM_OPACITY '1'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Transparency

VARIANT_PROPERTY_ITEM_OPACITY

Controls the opacity or transparency of variant properties on the canvas.

Syntax

```
VARIANT_PROPERTY_ITEM_OPACITY '<opacity_factor>'
```

Where

`opacity_factor`	Any number between 0 and 99. A lower value results in a more opaque variant property. An opacity factor of 1 indicates that the variant property is completely opaque, while a value of 99 indicates that the background of the variant property is transparent. The default value is `'1'`.

Examples

```
VARIANT_PROPERTY_ITEM_OPACITY '1'
```

VARIANT_PROPERTY_TEXT_FONT_BOLD

Specifies whether the variant property text appears in **bold** face.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_BOLD 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
VARIANT_PROPERTY_TEXT_FONT_BOLD 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Text Font

VARIANT_PROPERTY_TEXT_FONT_COLOR

Sets the font color of the variant property text.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_COLOR '<font_color>'
```

Where

`font_color`	The default font color of variant property text. The value can be specified as a hex color code or the name of the color. For example, you can specify, `'#FF0000'` or `'RED'` to represent red. The default value is `'#0000FF'` (Blue).

Examples

```
VARIANT_PROPERTY_TEXT_FONT_COLOR '#FCD054'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Font Color

VARIANT_PROPERTY_TEXT_FONT_ITALIC

Specifies whether the variant property text appears in *italics*.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_ITALIC 'TRUE'|'FALSE'
```

The default value is FALSE.

Examples

```
VARIANT_PROPERTY_TEXT_FONT_ITALIC 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Text Font

VARIANT_PROPERTY_TEXT_FONT_NAME

Sets the font face used to display variant property text.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_NAME '<font_name>'
```

Where

`font_name`	The default font face used to display variant property text. The value can be any of the system-supported fonts. The default value is `'ARIAL'`.

Examples

```
VARIANT_PROPERTY_TEXT_FONT_NAME 'ARIAL'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Font Face

VARIANT_PROPERTY_TEXT_FONT_SIZE

Sets the font size of variant property text.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_SIZE '<font_size>'
```

Where

`font_size`	The default font size of variant property text. The default value is `'5'`. Valid Range: 1 to 72

Examples

```
VARIANT_PROPERTY_TEXT_FONT_SIZE '10'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Font Size

VARIANT_PROPERTY_TEXT_FONT_UNDERLINE

Specifies whether the variant property text appears underlined.

Syntax

```
VARIANT_PROPERTY_TEXT_FONT_UNDERLINE 'TRUE'|'FALSE'
```

The default value is TRUE.

Examples

```
VARIANT_PROPERTY_TEXT_FONT_UNDERLINE 'FALSE'
```

Corresponding UI Option for Allegro System Capture

Edit - Preferences - Schematic - Variant View - Text Font

22

W Directives

This chapter lists the CPM directives that start with `W` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

WARNING_MESSAGES

Specifies where you want Warning messages to display.

Syntax

```
WARNING_MESSAGES 'DIALOG'|'COMMANDPANE'|'SUPPRESS'
```

where

DIALOG	If you set the value to Dialog, Design Entry HDL displays the messages in a dialog box.
COMMANDPANE	If you set this directive to COMMANDPANE, Design Entry HDL displays the messages in the Console Command Window.
SUPPRESS	If you set the variable to SUPPRESS Design Entry HDL does not display the type of messages.

Example

```
WARNING_MESSAGES 'DIALOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Messages — Warning

See Also

- FATAL_MESSAGES
- ERROR_MESSAGES
- WARNING_MESSAGES

WINDOWSMODE_FLAG

Activates the Windows mode. You need to set the PRESELECT_FLAG directive to 'ON' before setting this directive.

Syntax

```
WINDOWSMODE_FLAG 'ON'|'OFF'
```

Example

```
WINDOWSMODE_FLAG 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — General page — Preferences — Windows Mode option

See Also

PRESELECT_FLAG

WINDOWSMODE_FLAG ALLOW_USER_CPM

Allows the `WINDOWSMODE_FLAG` directive to be defined in the `user.cpm` file.

Syntax

```
WINDOWSMODE_FLAG ALLOW_USER_CPM 'ON'|'OFF'
```

Example

```
WINDOWSMODE_FLAG ALLOW_USER_CPM 'ON'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

PRESELECT_FLAG ALLOW_USER_CPM

WINDOWSMODE_FLAG

WIRE

Draws wires that you add and move as Orthogonal or Direct.

Syntax

```
WIRE 'ORTHOGONAL'|'DIRECT'
```

Example

```
WIRE 'ORTHOG'
```

Corresponding UI Option for Allegro Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Graphics page — Wires — Add

WIRE_COLOR

Use this directive to specify the color used to draw nets in the generated document schematic in Design Entry HDL and Schgen.

Syntax

```
WIRE_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'|'DEFAULT'
```

Example

```
WIRE_COLOR 'yellow'
```

Corresponding UI Option for Design Entry HDL

Tools — Options — Design Entry HDL Options dialog box — Color page — Graphics Color — Wire

Corresponding UI Option for System Connectivity Manager

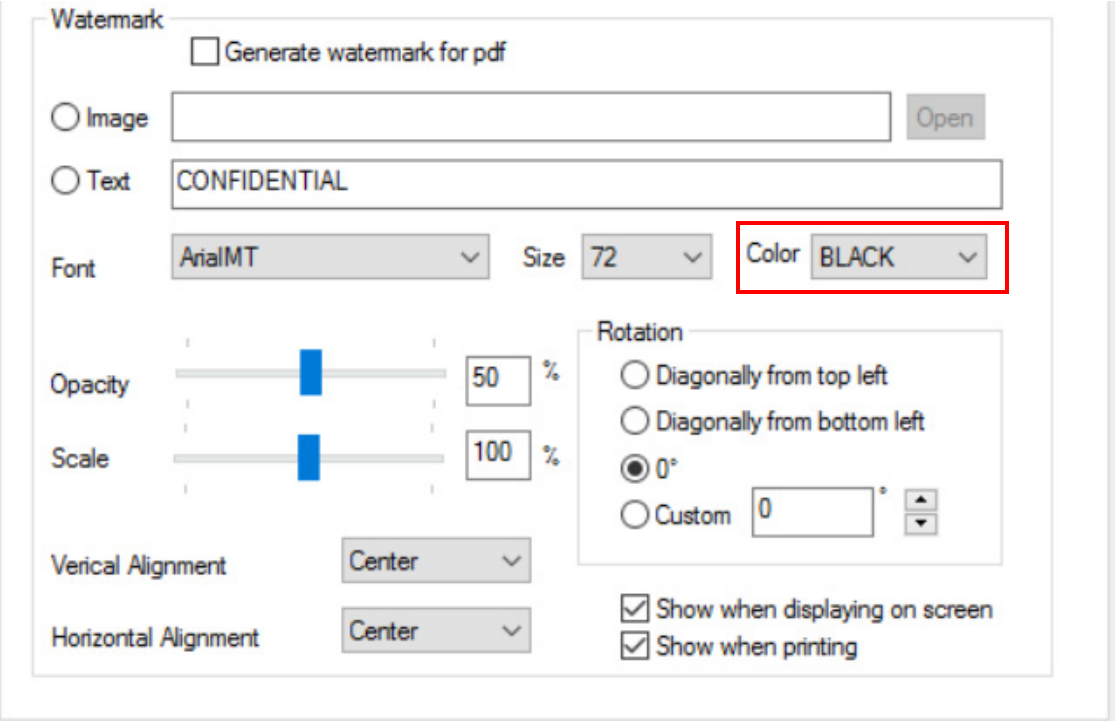
Project — Settings — Document Schematic Generation — Colors — Wire Color.

See Also

- ARC_COLOR
- BACKGROUND_COLOR
- DOT_COLOR

WM_COLOR

This directive is set when you define the font color for watermark text in a schematic PDF.



Syntax

```
WM_COLOR '<Color>'
```

Example

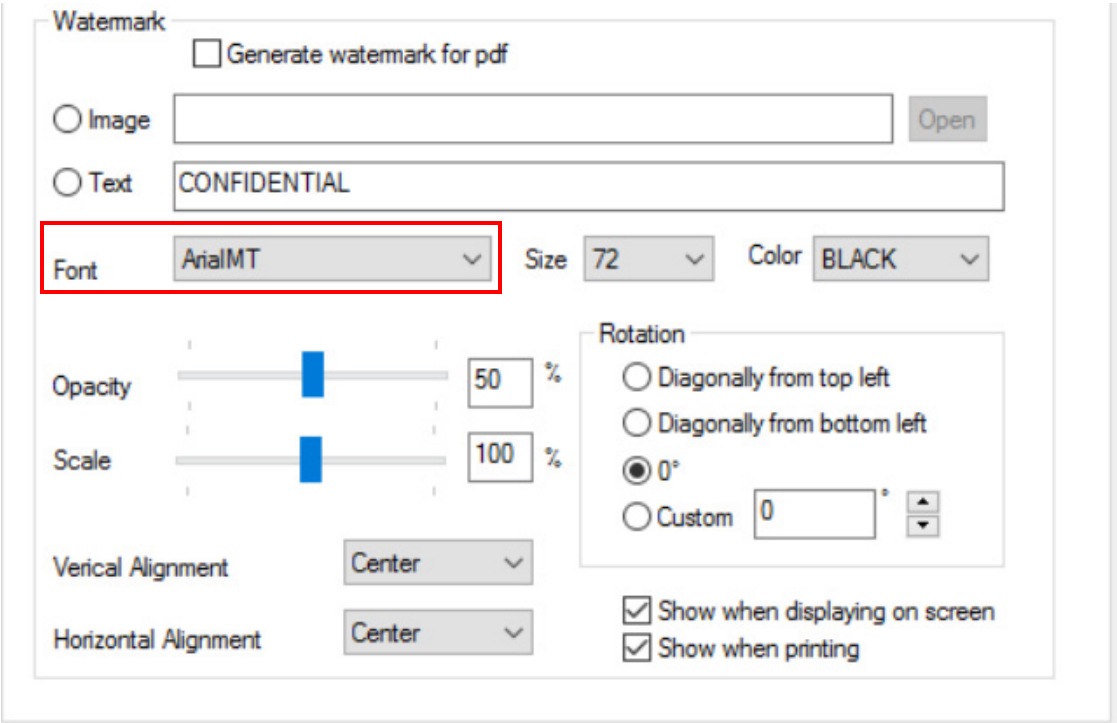
```
WM_COLOR 'Orange'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Color

WM_FONT

This directive is set when you define the font for watermark text in a schematic PDF.



Syntax

```
WM_FONT '<Font>'
```

Example

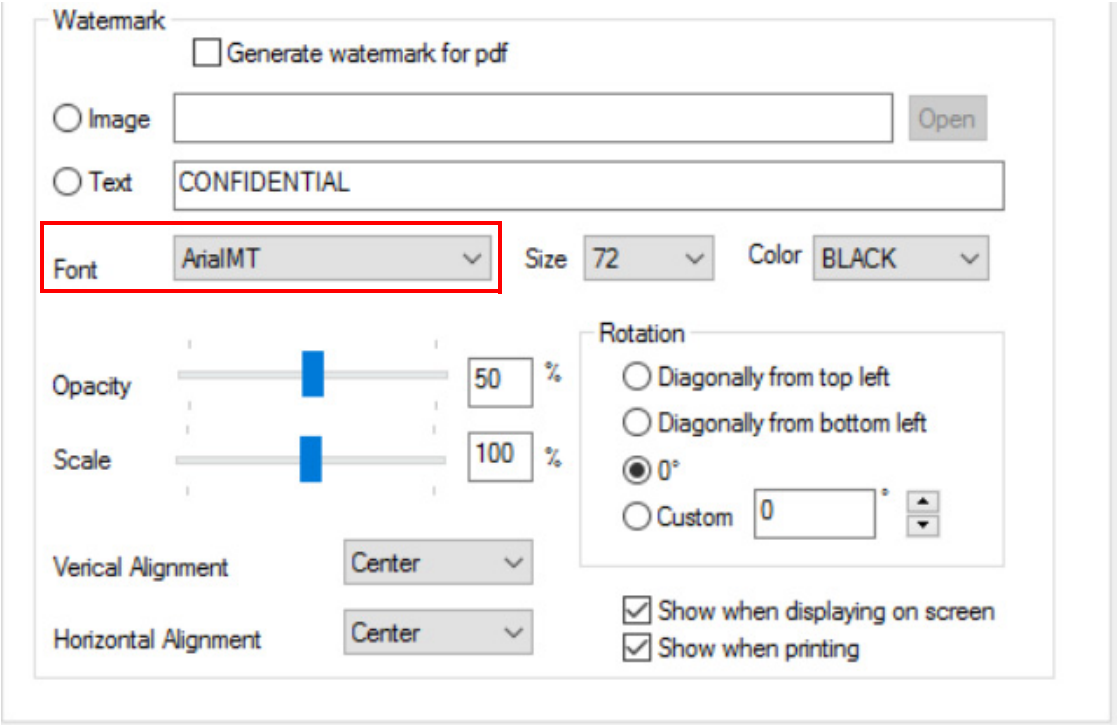
```
WM_FONT 'ArialNarrow'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Font

WM_FONTSIZE

This directive is set when you define the font size for watermark text in a schematic PDF. The default and the maximum size limit is 72.



Syntax

```
WM_SIZE '<Size>'
```

Example

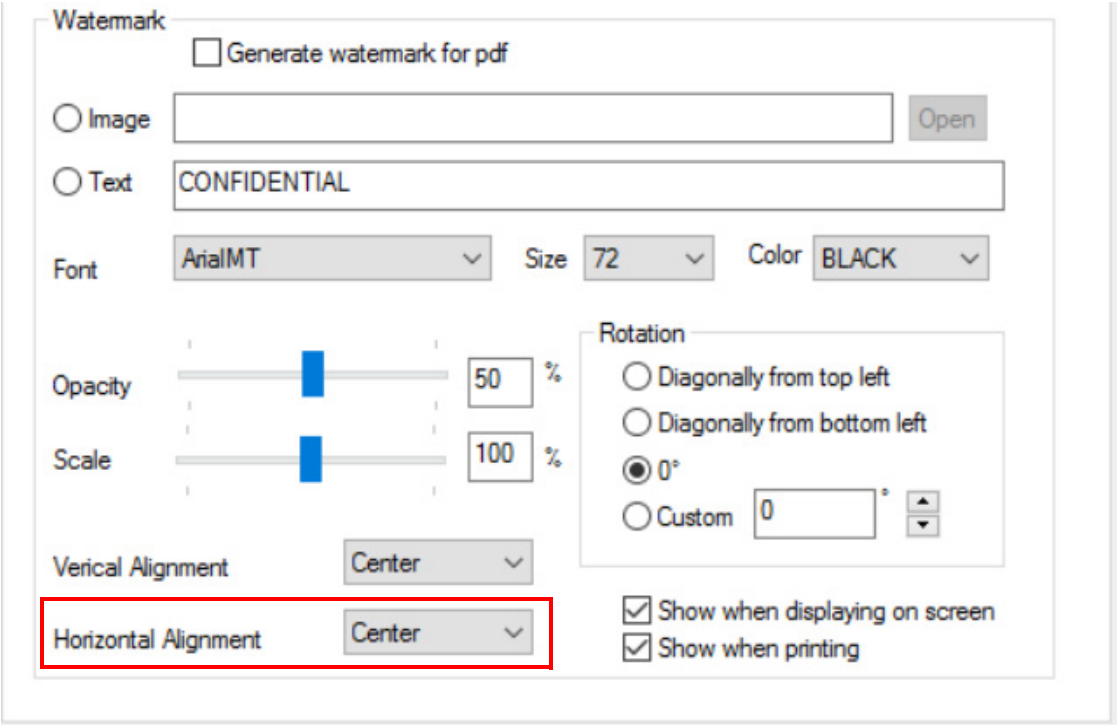
```
WM_SIZE '22'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Size

WM_HORIZONTAL_ALIGNMENT

This directive is set when you define the horizontal alignment for watermark text in a schematic PDF.



Syntax

```
WM_HORIZONTAL_ALIGNMENT 'Left|Center|Right'
```

Example

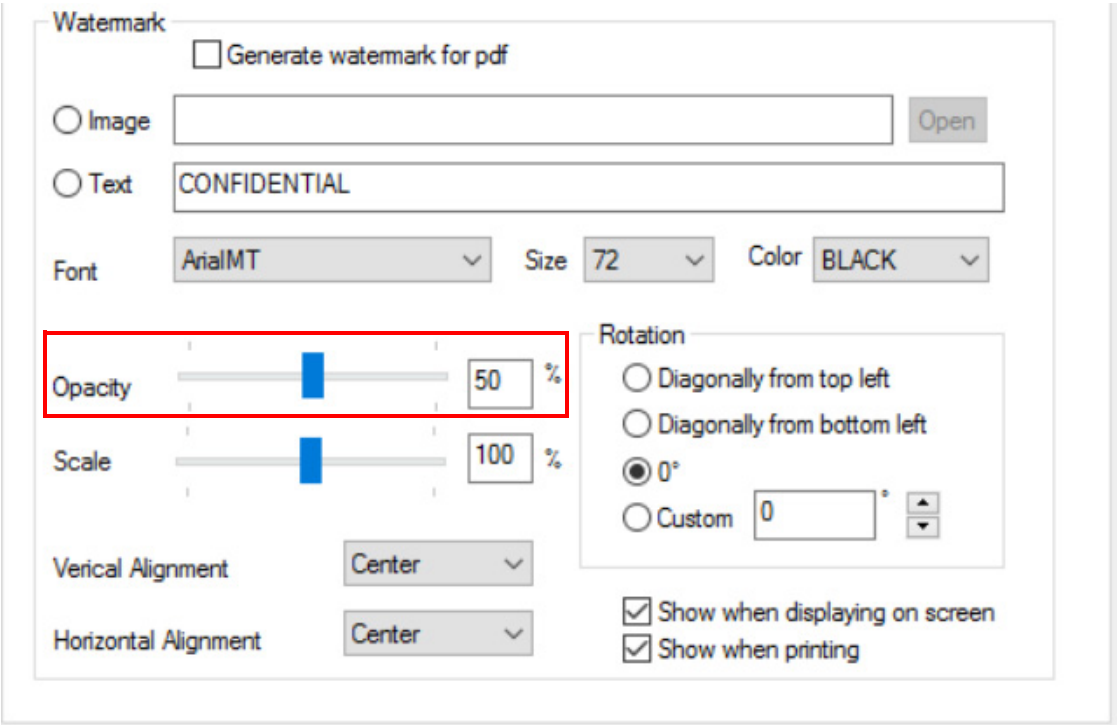
```
WM_HORIZONTAL_ALIGNMENT 'Left'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Horizontal Alignment

WM_OPACITY

This directive is set when you specify the degree of opacity for watermark text in a schematic PDF. On a scale of 0% to 100%, 0 would make the watermark fully transparent and 100 would mean that the watermark is fully opaque.



Syntax

```
WM_OPACITY '<0-100>'
```

Example

```
WM_OPACITY '40'
```

Corresponding UI Option for Design Entry HDL

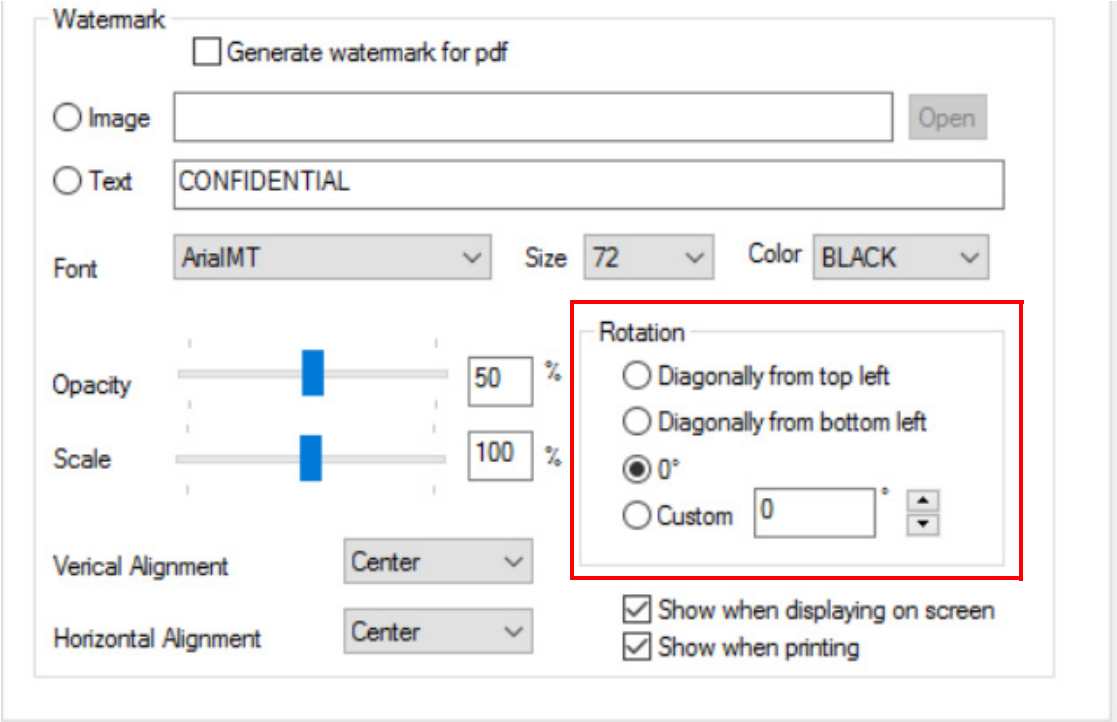
File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Opacity

WM_ROTATION

This directive is set when you select the degree of rotation for watermark text in a schematic PDF.

The options are as follows:

- *Diagonally from top left*: Select this option to display the watermark from the top left.
- *Diagonally from bottom left*: Select this option to display the watermark from the bottom left.
- *0°*: Sets the rotation of the watermark to 0° degrees.
- *Custom*: Specify a custom angle of rotation for the watermark (0 to 360 degrees).



Syntax

```
WM_ROTATION 'Diagonally from top left | Diagonally from bottom left|0 |
     Custom'
```

Example

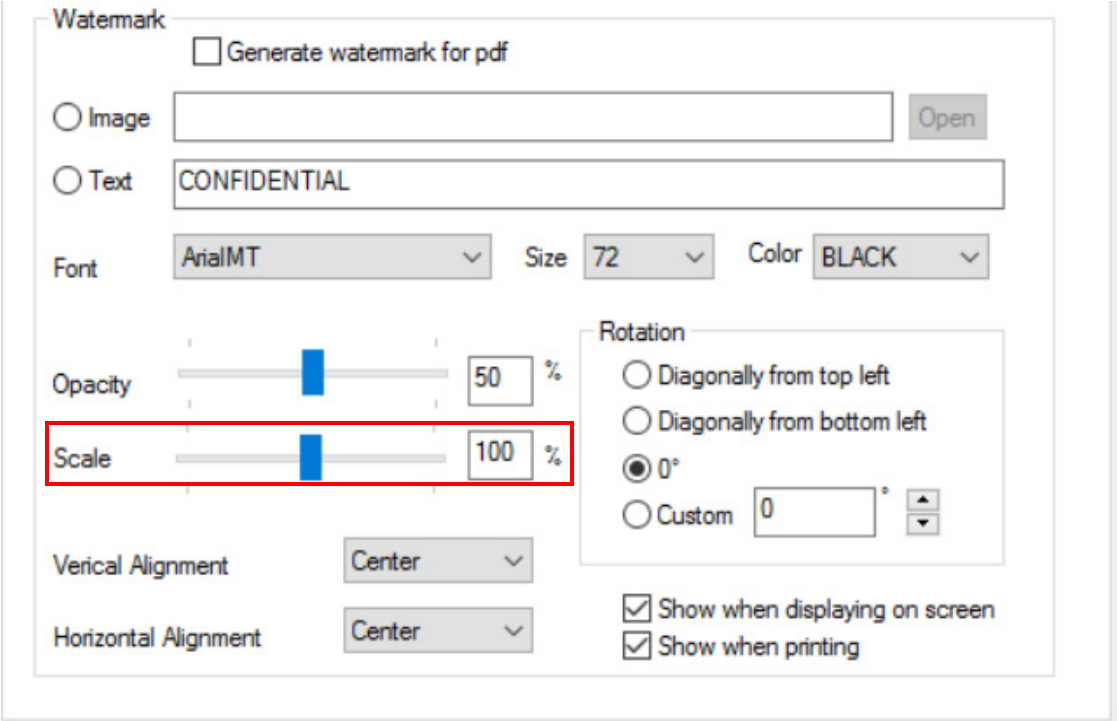
```
WM_ROTATION '40'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Rotation

WM_SCALE

This directive is set when you define the scale factor for watermark text in a schematic PDF.



Syntax

```
WM_SCALE '<1-100>'
```

Example

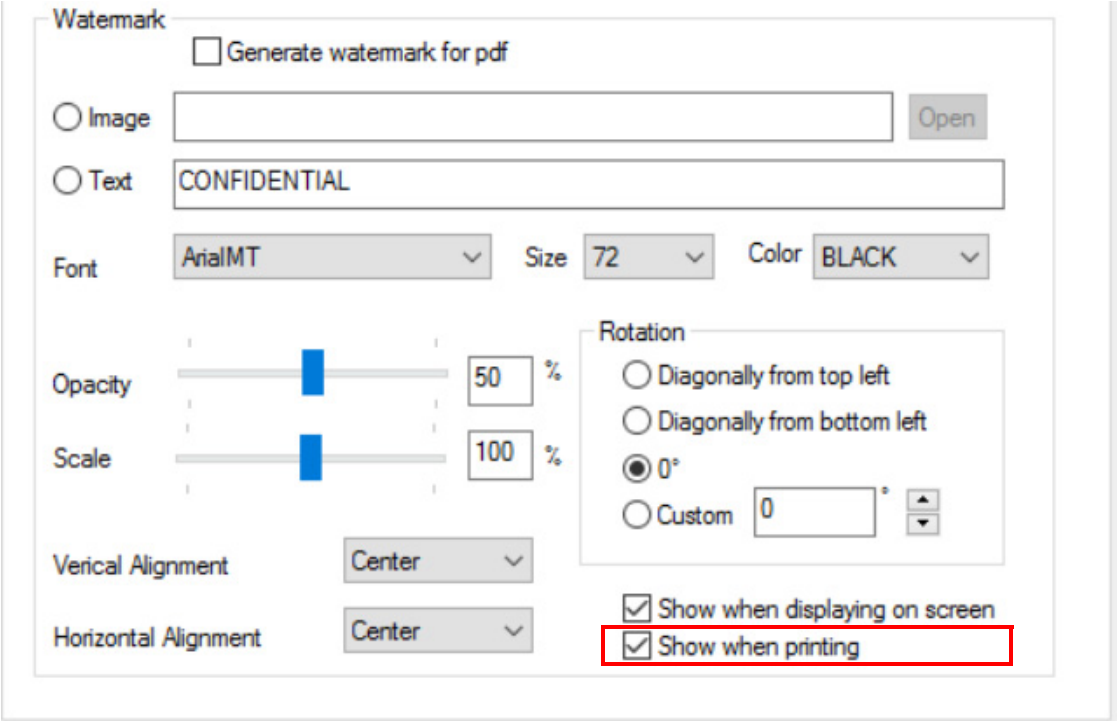
```
WM_SCALE '50'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Scale

WM_SHOW_ONPRINT

This directive specifies whether the watermark should be displayed in the printed PDF.



Syntax

```
WM_SHOW_ONPRINT 'Yes|No'
```

Example

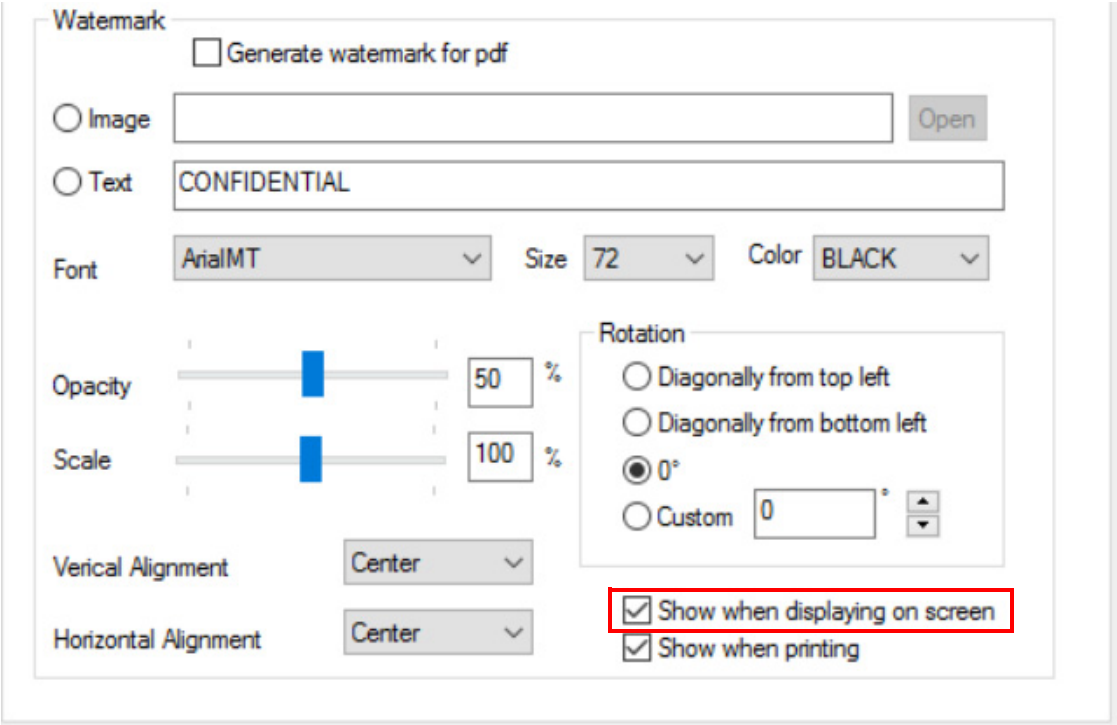
```
WM_SHOW_ONPRINT 'Yes'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Show when printing

WM_SHOW_ONSCREEN

This directive specifies whether the watermark should be displayed when viewing the PDF on screen.



Syntax

```
WM_SHOW_ONSCREEN 'Yes|No'
```

Example

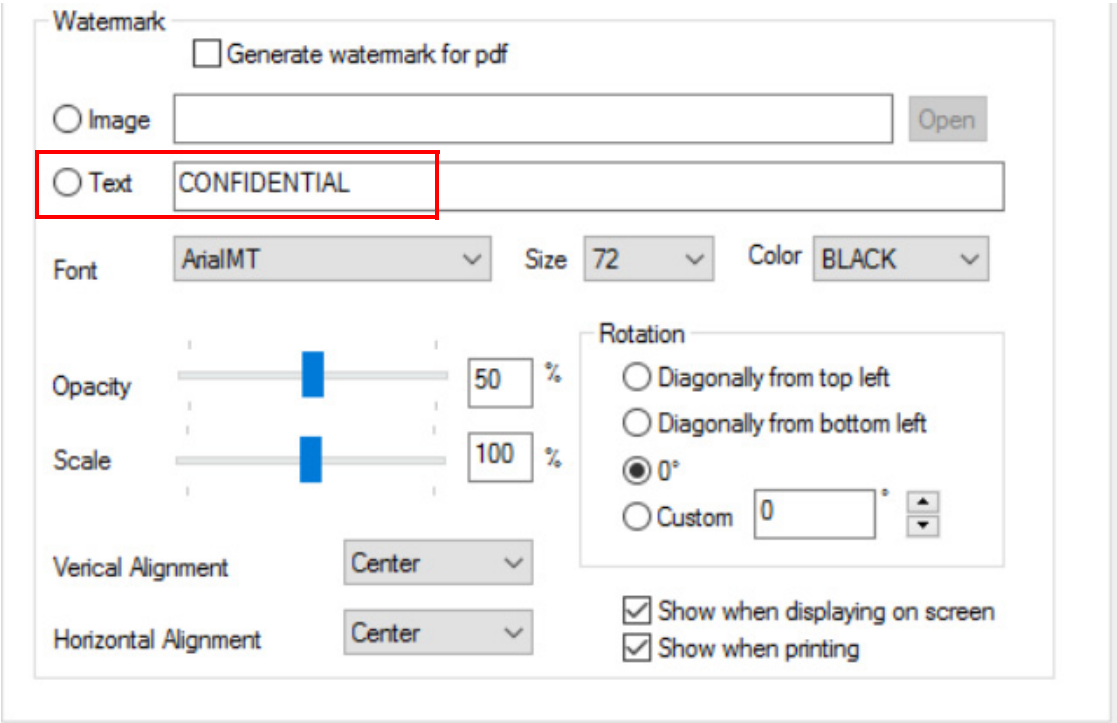
```
WM_SHOW_ONSCREEN 'Yes'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Show when displaying on screen

WM_TEXT

This directive specifies the text that you want as a watermark in your PDF file.



Syntax

```
WM_TEXT '<Text>'
```

Example

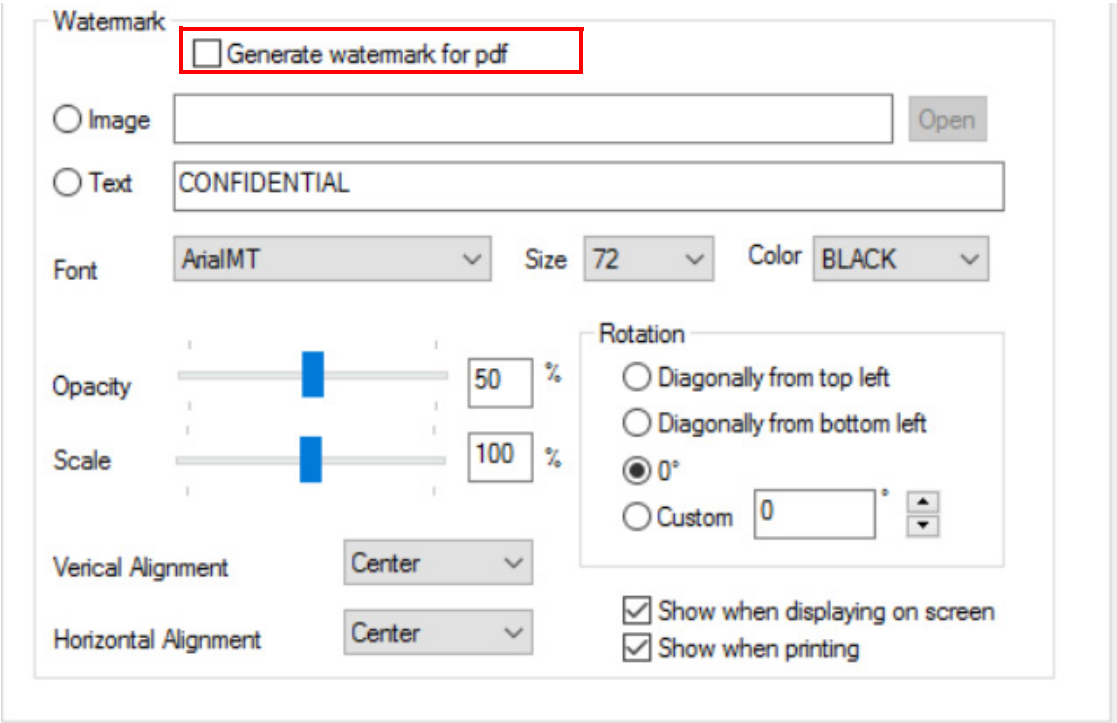
```
WM_TEXT 'FOR INTERNAL USE ONLY'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Text

WM_TYPE

Defines whether you want a watermark in the generated PDF.



Syntax

```
WM_TYPE '1|0'
```

Example

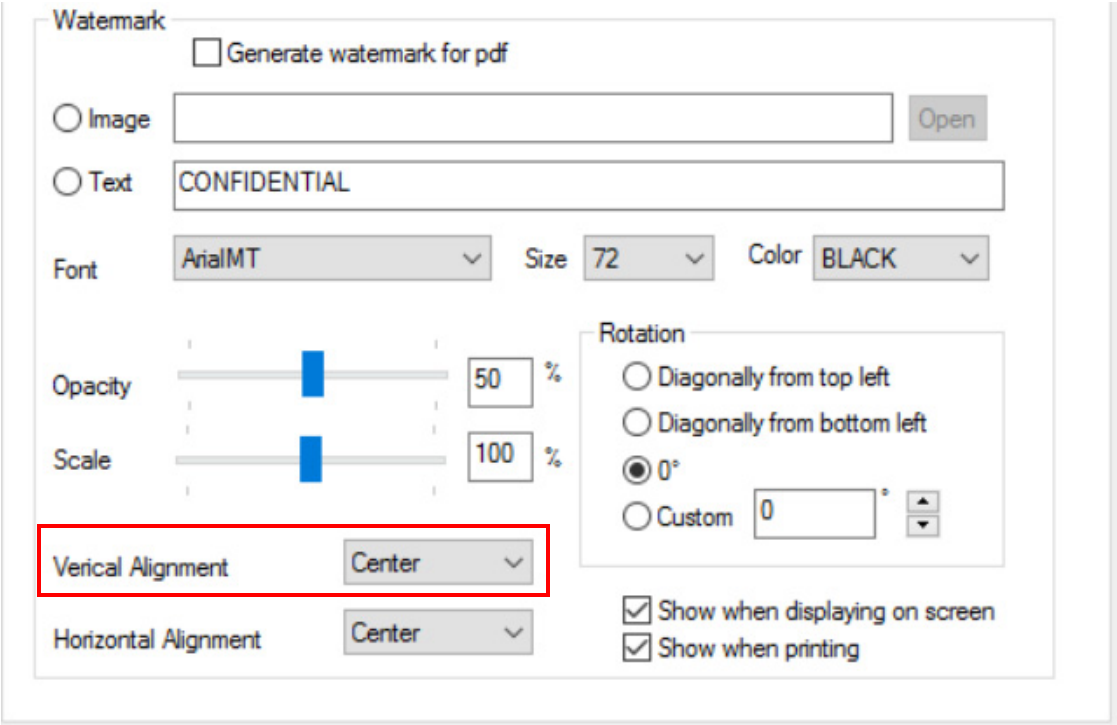
```
WM_TYPE '1'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Generate watermark for pdf

WM_VERTICAL_ALIGNMENT

This directive is set when you define the vertical placement of the watermark with respect to the document. The options are Top, Center and Bottom.



Syntax

```
WM_VERTICAL_ALIGNMENT 'Top|Center|Bottom'
```

Example

```
WM_VERTICAL_ALIGNMENT 'Top'
```

Corresponding UI Option for Design Entry HDL

File — Publish PDF — Setup — Design Entry HDL Options — Advanced — Watermark — Vertical Alignment

WARNINGS

The WARNINGS directive controls the display of all warning messages.

Syntax

```
WARNINGS on|off ;
```

on	Reports all warnings.
off	Does not report any warning.

The default value for the WARNINGS directive is on.

Example

```
WARNINGS off;
```

23

X Directives

This chapter lists the CPM directives that start with `X` and are used in the cpm files for all front-end products, such as Allegro System Capture, Design Entry HDL, System Connectivity Manager, and Packager-XL.

XNET_ABSENT_COLOR

Changes the default color of the cross sign which is displayed on a component that does not has XNets on its pins.

Syntax

```
XNET_ABSENT_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'
```

Example

```
XNET_ABSENT_COLOR 'GREEN'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

- SHOW_XNET_STATUS
- XNET_EXISTS_COLOR

XNET_EXISTS_COLOR

Changes the default color of the arrow sign which is displayed on a component that has XNets on its pins.

Syntax

```
XNET_EXISTS_COLOR
     'RED'|'BLUE'|'GREEN'|'YELLOW'|'ORANGE'|'SALMON'|'VIOLET'|'BROWN'|'SKYBLUE'|'
     WHITE'|'PEACH'|'BLUE'|'PINK'|'PURPLE'|'AQUA'|'GRAY'|'MONO'
```

Example

```
XNET_EXISTS_COLOR 'GREEN'
```

Corresponding UI Option for Allegro Design Entry HDL

None

See Also

- SHOW_XNET_STATUS
- XNET_ABSENT_COLOR