

DIN EN 61804-5

ICS 35.240.50; 35.060

Einsprüche bis 2014-04-07

Entwurf**Funktionsbausteine für die Prozessautomation und elektronische Gerätebeschreibungssprache –****Teil 5: Bibliothek vorgefertigter Unterprogramme****(IEC 65E/321/CDV:2013);****Englische Fassung FprEN 61804-5:2013**

Function blocks (FB) for process control and EDDL –

Part 5: EDDL Builtin library

(IEC 65E/321/CDV:2013);

English version FprEN 61804-5:2013

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2014-02-07 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/stellungnahme oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (Hausanschrift: Stresemannallee 15, 60596 Frankfurt am Main).

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 211 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments FprEN 61804-5:2013 „Funktionsbausteine für die Prozessautomation und elektronische Gerätebeschreibungssprache – Teil 5: Bibliothek vorgefertigter Unterprogramme“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 65E/321/CDV:2013 „Function blocks (FB) for process control and EDDL – Part 5: EDDL Built-in library“ unverändert in den Entwurf FprEN 61804-5:2013 übernommen worden.

Das internationale Dokument wurde vom SC 65E „Devices and integration in enterprise systems“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für dieses Dokument ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnichtsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für den vorliegenden Normentwurf erfüllt.

Da sich die Benutzer des vorliegenden Normentwurfs der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61804-5 veröffentlicht. Dieser Normentwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Es ist vorgesehen, auch bei der entsprechenden zukünftigen Deutschen Norm auf die deutsche Sprachfassung zu verzichten und diese in der englischsprachigen Fassung zu veröffentlichen.

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC 61804-3.

CONTENTS

1		
2		
3	FOREWORD.....	19
4	INTRODUCTION.....	21
5	1 Scope.....	22
6	2 Normative references	22
7	3 Terms, definitions, acronyms and abbreviated terms	22
8	3.1 Terms and definitions	22
9	3.2 Acronyms and abbreviated terms	23
10	4 EDDL Builtin library	23
11	4.1 General.....	23
12	4.2 Conventions for Builtin descriptions.....	23
13	4.3 Builtin Categories	24
14	4.3.1 Overview	24
15	4.3.2 User Interface Builtins	25
16	4.3.3 Communication Builtins	26
17	4.3.4 Action Builtins	27
18	4.4 Builtin _ERROR.....	28
19	4.5 Builtin _TRACE	28
20	4.6 Builtin _WARNING	29
21	4.7 Builtin abort.....	29
22	4.8 Builtin abort_on_all_comm_errors	30
23	4.9 Builtin ABORT_ON_ALL_COMM_STATUS	30
24	4.10 Builtin ABORT_ON_ALL_DEVICE_STATUS	31
25	4.11 Builtin ABORT_ON_ALL_RESPONSE_CODES	31
26	4.12 Builtin abort_on_all_response_codes	32
27	4.13 Builtin ABORT_ON_COMM_ERROR	32
28	4.14 Builtin abort_on_comm_error.....	32
29	4.15 Builtin ABORT_ON_COMM_STATUS	33
30	4.16 Builtin ABORT_ON_DEVICE_STATUS	33
31	4.17 Builtin ABORT_ON_NO_DEVICE	34
32	4.18 Builtin ABORT_ON_RESPONSE_CODE.....	34
33	4.19 Builtin abort_on_response_code.....	35
34	4.20 Builtin abortTransferPort	36
35	4.21 Builtin abs	36
36	4.22 Builtin ACKNOWLEDGE	37
37	4.23 Builtin acknowledge.....	37
38	4.24 Builtin acos	37
39	4.25 Builtin add_abort_method (version A).....	38
40	4.26 Builtin add_abort_method (version B).....	38
41	4.27 Builtin AddTime	39
42	4.28 Builtin asin	39
43	4.29 Builtin assign.....	39
44	4.30 Builtin assign_double	40
45	4.31 Builtin assign_float.....	40
46	4.32 Builtin assign_int.....	40

47	4.33 Builtin assign_var	41
48	4.34 Builtin assign2	41
49	4.35 Builtin atan	42
50	4.36 Builtin atof	42
51	4.37 Builtin atoi	42
52	4.38 Builtin browseIdentity	43
53	4.39 Builtin BUILD_MESSAGE	43
54	4.40 Builtin ByteToDouble	43
55	4.41 Builtin ByteToFloat	44
56	4.42 Builtin ByteToLong	44
57	4.43 Builtin ByteToShort	45
58	4.44 Builtin cbrt	45
59	4.45 Builtin ceil	45
60	4.46 Builtin closeTransferPort	46
61	4.47 Builtin cos	46
62	4.48 Builtin cosh	46
63	4.49 Builtin dassign	47
64	4.50 Builtin DATE_AND_TIME_VALUE_to_string	47
65	4.51 Builtin Date_to_DayOfMonth	47
66	4.52 Builtin DATE_to_days	48
67	4.53 Builtin Date_to_Month	48
68	4.54 Builtin DATE_to_string	48
69	4.55 Builtin Date_To_Time	49
70	4.56 Builtin Date_to_Year	49
71	4.57 Builtin days_to_DATE	49
72	4.58 Builtin DELAY	50
73	4.59 Builtin delay	50
74	4.60 Builtin DELAY_TIME	51
75	4.61 Builtin delayfor	51
76	4.62 Builtin delayfor2	52
77	4.63 Builtin DICT_ID	52
78	4.64 Builtin dictionary_string	53
79	4.65 Builtin DiffTime	53
80	4.66 Builtin discard_on_exit	53
81	4.67 Builtin DISPLAY	54
82	4.68 Builtin display	54
83	4.69 Builtin display_bitenum	55
84	4.70 Builtin display_builtin_error	55
85	4.71 Builtin display_comm_error	56
86	4.72 Builtin display_comm_status	56
87	4.73 Builtin display_device_status	57
88	4.74 Builtin display_dynamics	57
89	4.75 Builtin display_dynamics2	58
90	4.76 Builtin display_message	58
91	4.77 Builtin display_message2	59
92	4.78 Builtin display_response_code	59
93	4.79 Builtin display_response_status	60
94	4.80 Builtin display_xmtr_status	61
95	4.81 Builtin DoubleToByte	

96	4.82 Builtin drand	62
97	4.83 Builtin dseed	62
98	4.84 Builtin edit_device_value	62
99	4.85 Builtin edit_device_value2	63
100	4.86 Builtin edit_local_value	64
101	4.87 Builtin edit_local_value2	65
102	4.88 Builtin exp	65
103	4.89 Builtin ext_send_command	66
104	4.90 Builtin ext_send_command_trans	66
105	4.91 Builtin fail_on_all_comm_errors	67
106	4.92 Builtin fail_on_all_response_codes	67
107	4.93 Builtin fail_on_comm_error	68
108	4.94 Builtin fail_on_response_code	68
109	4.95 Builtin fassign	69
110	4.96 Builtin fGetByte	69
111	4.97 Builtin fgetval	70
112	4.98 Builtin float_value	70
113	4.99 Builtin FloatToByte	70
114	4.100 Builtin floor	71
115	4.101 Builtin fmod	71
116	4.102 Builtin fpclassify	72
117	4.103 Builtin From_DATE_AND_TIME_VALUE	72
118	4.104 Builtin From_TIME_VALUE	72
119	4.105 Builtin fsetval	73
120	4.106 Builtin ftoa	73
121	4.107 Builtin fvar_value	73
122	4.108 Builtin get_acknowledgement	74
123	4.109 Builtin get_acknowledgement2	74
124	4.110 Builtin get_block_instance_by_object_index	75
125	4.111 Builtin get_block_instance_by_tag	75
126	4.112 Builtin get_block_instance_count	76
127	4.113 Builtin get_comm_error	76
128	4.114 Builtin get_comm_error_string	77
129	4.115 Builtin get_date	77
130	4.116 Builtin get_date_lelem	78
131	4.117 Builtin get_date_lelem2	78
132	4.118 Builtin get_date_value	79
133	4.119 Builtin get_date_value2	79
134	4.120 Builtin GET_DD_REVISION	80
135	4.121 Builtin get_dds_error	80
136	4.122 Builtin GET_DEV_VAR_VALUE	81
137	4.123 Builtin get_dev_var_value	81
138	4.124 Builtin GET_DEVICE_REVISION	82
139	4.125 Builtin GET_DEVICE_TYPE	82
140	4.126 Builtin get_dictionary_string	82
141	4.127 Builtin get_double	83
142	4.128 Builtin get_double_lelem	83
143	4.129 Builtin get_double_lelem2	84
144	4.130 Builtin get_double_value	

145	4.131 Builtin get_double_value2	85
146	4.132 Builtin get_enum_string	85
147	4.133 Builtin get_float	86
148	4.134 Builtin get_float_lelem	86
149	4.135 Builtin get_float_lelem2	87
150	4.136 Builtin get_float_value	87
151	4.137 Builtin get_float_value2	88
152	4.138 Builtin GET_LOCAL_VAR_VALUE	88
153	4.139 Builtin get_local_var_value	89
154	4.140 Builtin GET_MANUFACTURER	89
155	4.141 Builtin get_more_status	89
156	4.142 Builtin get_resolve_status	90
157	4.143 Builtin get_response_code	90
158	4.144 Builtin get_response_code_string	91
159	4.145 Builtin get_rspcode_string	92
160	4.146 Builtin get_rspcode_string_by_id	92
161	4.147 Builtin get_signed	92
162	4.148 Builtin get_signed_lelem	93
163	4.149 Builtin get_signed_lelem2	93
164	4.150 Builtin get_signed_value	94
165	4.151 Builtin get_signed_value2	94
166	4.152 Builtin get_status_code_string	95
167	4.153 Builtin get_status_string	95
168	4.154 Builtin get_stddict_string	96
169	4.155 Builtin get_string	96
170	4.156 Builtin get_string_lelem	97
171	4.157 Builtin get_string_lelem2	97
172	4.158 Builtin get_string_value	98
173	4.159 Builtin get_string_value2	98
174	4.160 Builtin GET_TICK_COUNT	99
175	4.161 Builtin get_transfer_status	99
176	4.162 Builtin get_unsigned	100
177	4.163 Builtin get_unsigned_lelem	100
178	4.164 Builtin get_unsigned_lelem2	101
179	4.165 Builtin get_unsigned_value	101
180	4.166 Builtin get_unsigned_value2	102
181	4.167 Builtin get_variable_string	103
182	4.168 Builtin GetCurrentDate	103
183	4.169 Builtin GetCurrentDateAndTime	103
184	4.170 Builtin GetCurrentTime	104
185	4.171 Builtin iassign	104
186	4.172 Builtin igetval	104
187	4.173 Builtin IGNORE_ALL_COMM_STATUS	105
188	4.174 Builtin IGNORE_ALL_DEVICE_STATUS	105
189	4.175 Builtin IGNORE_ALL_RESPONSE_CODES	106
190	4.176 Builtin IGNORE_COMM_ERROR	106
191	4.177 Builtin IGNORE_COMM_STATUS	107
192	4.178 Builtin IGNORE_DEVICE_STATUS	107
193	4.179 Builtin IGNORE_NO_DEVICE	107

194	4.180 Builtin IGNORE_RESPONSE_CODE	108
195	4.181 Builtin int_value	109
196	4.182 Builtin is_NaN	109
197	4.183 Builtin isetval	109
198	4.184 Builtin isOffline	110
199	4.185 Builtin ITEM_ID	110
200	4.186 Builtin itoa (version A)	110
201	4.187 Builtin itoa (version B)	111
202	4.188 Builtin ivar_value	111
203	4.189 Builtin lassign	111
204	4.190 Builtin lgetval	112
205	4.191 Builtin ListDeleteElementAt	112
206	4.192 Builtin ListDeleteElementAt2	113
207	4.193 Builtin ListInsert	113
208	4.194 Builtin ListInsert2	114
209	4.195 Builtin log	114
210	4.196 Builtin LOG_MESSAGE	114
211	4.197 Builtin log10	115
212	4.198 Builtin log2	115
213	4.199 Builtin long_value	115
214	4.200 Builtin LongToByte	116
215	4.201 Builtin lsetval	116
216	4.202 Builtin lvar_value	117
217	4.203 Builtin Make_Time	117
218	4.204 Builtin MEMBER_ID	117
219	4.205 Builtin MenuDisplay	118
220	4.206 Builtin method_abort	118
221	4.207 Builtin nan	119
222	4.208 Builtin NaN_value	119
223	4.209 Builtin nanf	120
224	4.210 Builtin ObjectReference	120
225	4.211 Builtin openTransferPort	121
226	4.212 Builtin pop_abort_method	121
227	4.213 Builtin pow	121
228	4.214 Builtin process_abort	122
229	4.215 Builtin push_abort_method	122
230	4.216 Builtin put_date	122
231	4.217 Builtin put_date_value	123
232	4.218 Builtin put_date_value2	123
233	4.219 Builtin put_double	124
234	4.220 Builtin put_double_value	125
235	4.221 Builtin put_double_value2	125
236	4.222 Builtin put_float	126
237	4.223 Builtin put_float_value	126
238	4.224 Builtin put_float_value2	127
239	4.225 Builtin PUT_MESSAGE	127
240	4.226 Builtin put_message	128
241	4.227 Builtin put_signed	129
242	4.228 Builtin put_signed_value	

243	4.229 Builtin put_signed_value2	130
244	4.230 Builtin put_string	130
245	4.231 Builtin put_string_value	131
246	4.232 Builtin put_string_value2	132
247	4.233 Builtin put_unsigned	132
248	4.234 Builtin put_unsigned_value	133
249	4.235 Builtin put_unsigned_value2	134
250	4.238 Builtin read_value	135
251	4.239 Builtin read_value2	136
252	4.240 Builtin ReadCommand	136
253	4.241 Builtin readItemFromDevice	136
254	4.242 Builtin remove_abort_method (version A)	137
255	4.243 Builtin remove_abort_method (version B)	137
256	4.244 Builtin remove_all_abort_methods	138
257	4.245 Builtin resolve_array_ref	138
258	4.246 Builtin resolve_block_ref	139
259	4.247 Builtin resolve_block_ref2	139
260	4.248 Builtin resolve_list_ref	140
261	4.249 Builtin resolve_local_ref	140
262	4.250 Builtin resolve_local_ref2	141
263	4.251 Builtin resolve_param_list_ref	141
264	4.252 Builtin resolve_param_ref	142
265	4.253 Builtin resolve_param_ref2	142
266	4.254 Builtin resolve_record_ref	143
267	4.255 Builtin resolve_record_ref2	143
268	4.256 Builtin ret_double_value	144
269	4.257 Builtin ret_double_value2	144
270	4.258 Builtin ret_float_value	145
271	4.259 Builtin ret_float_value2	145
272	4.260 Builtin ret_signed_value	145
273	4.261 Builtin ret_signed_value2	146
274	4.262 Builtin ret_unsigned_value	146
275	4.263 Builtin ret_unsigned_value2	147
276	4.264 Builtin retry_on_all_comm_errors	147
277	4.265 Builtin RETRY_ON_ALL_COMM_STATUS	148
278	4.266 Builtin RETRY_ON_ALL_DEVICE_STATUS	148
279	4.267 Builtin RETRY_ON_ALL_RESPONSE_CODES	149
280	4.268 Builtin retry_on_all_response_codes	149
281	4.269 Builtin RETRY_ON_COMM_ERROR	150
282	4.270 Builtin retry_on_comm_error	150
283	4.271 Builtin RETRY_ON_COMM_STATUS	151
284	4.272 Builtin RETRY_ON_DEVICE_STATUS	151
285	4.273 Builtin RETRY_ON_NO_DEVICE	152
286	4.274 Builtin RETRY_ON_RESPONSE_CODE	152
287	4.275 Builtin retry_on_response_code	153
288	4.276 Builtin round	153
289	4.277 Builtin save_on_exit	153
290	4.278 Builtin save_values	154
291	4.279 Builtin seconds_to_TIME_VALUE	154

292	4.280 Builtin SELECT_FROM_LIST	155
293	4.281 Builtin select_from_list	155
294	4.282 Builtin select_from_menu	156
295	4.283 Builtin select_from_menu2	157
296	4.284 Builtin send	157
297	4.285 Builtin send_all_values	158
298	4.286 Builtin send_command	158
299	4.287 Builtin send_command_trans	159
300	4.288 Builtin send_on_exit	159
301	4.289 Builtin send_trans	160
302	4.290 Builtin send_value	160
303	4.291 Builtin send_value2	161
304	4.292 Builtin SET_NUMBER_OF_RETRIES	162
305	4.293 Builtin sgetval	162
306	4.294 Builtin ShortToByte	162
307	4.295 Builtin sin	163
308	4.296 Builtin sinh	163
309	4.297 Builtin sqrt	163
310	4.298 Builtin ssetval	164
311	4.299 Builtin strcmp	164
312	4.300 Builtin strleft	164
313	4.301 Builtin strlen	165
314	4.302 Builtin strlwr	165
315	4.303 Builtin strmid	165
316	4.304 Builtin strright	166
317	4.305 Builtin strstr	166
318	4.306 Builtin strtrim	166
319	4.307 Builtin strupr	167
320	4.308 Builtin tan	167
321	4.309 Builtin tanh	167
322	4.310 Builtin Time_To_Date	168
323	4.311 Builtin TIME_VALUE_to_Hour	168
324	4.312 Builtin TIME_VALUE_to_Minute	168
325	4.313 Builtin TIME_VALUE_to_Second	169
326	4.314 Builtin TIME_VALUE_to_seconds	169
327	4.315 Builtin TIME_VALUE_to_string	169
328	4.316 Builtin timet_to_string	170
329	4.317 Builtin timet_to_TIME_VALUE	170
330	4.318 Builtin To_Date	170
331	4.319 Builtin To_Date_and_Time	171
332	4.320 Builtin To_Time	171
333	4.321 Builtin To_TIME_VALUE	172
334	4.322 Builtin trunc	172
335	4.323 Builtin VARID	172
336	4.324 Builtin vassign	173
337	4.325 Builtin WriteCommand	173
338	4.326 Builtin writeItemToDevice	173
339	4.327 Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	174
340	4.328 Builtin XMTR_ABORT_ON_ALL_DATA	

341	4.329 Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS.....	175
342	4.330 Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES.....	175
343	4.331 Builtin XMTR_ABORT_ON_COMM_ERROR.....	176
344	4.332 Builtin XMTR_ABORT_ON_COMM_STATUS.....	176
345	4.333 Builtin XMTR_ABORT_ON_DATA.....	177
346	4.334 Builtin XMTR_ABORT_ON_DEVICE_STATUS.....	177
347	4.335 Builtin XMTR_ABORT_ON_NO_DEVICE.....	178
348	4.336 Builtin XMTR_ABORT_ON_RESPONSE_CODE.....	178
349	4.337 Builtin XMTR_IGNORE_ALL_COMM_STATUS.....	179
350	4.338 Builtin XMTR_IGNORE_ALL_DATA.....	179
351	4.339 Builtin XMTR_IGNORE_ALL_DEVICE_STATUS.....	180
352	4.340 Builtin XMTR_IGNORE_ALL_RESPONSE_CODES.....	180
353	4.341 Builtin XMTR_IGNORE_COMM_ERROR.....	181
354	4.342 Builtin XMTR_IGNORE_COMM_STATUS.....	181
355	4.343 Builtin XMTR_IGNORE_DEVICE_STATUS.....	182
356	4.344 Builtin XMTR_IGNORE_NO_DEVICE.....	182
357	4.345 Builtin XMTR_IGNORE_RESPONSE_CODE.....	183
358	4.346 Builtin XMTR_RETRY_ON_ALL_COMM_STATUS.....	183
359	4.347 Builtin XMTR_RETRY_ON_ALL_DATA.....	184
360	4.348 Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS.....	184
361	4.349 Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE (removed).....	185
362	4.350 Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES.....	185
363	4.351 Builtin XMTR_RETRY_ON_COMM_ERROR.....	185
364	4.352 Builtin XMTR_RETRY_ON_COMM_STATUS.....	186
365	4.353 Builtin XMTR_RETRY_ON_DATA.....	186
366	4.354 Builtin XMTR_RETRY_ON_DEVICE_STATUS.....	187
367	4.355 Builtin XMTR_RETRY_ON_NO_DEVICE.....	187
368	4.356 Builtin XMTR_RETRY_ON_RESPONSE_CODE.....	188
369	4.357 Builtins return codes.....	188
370	Annex A (normative) Profiles of Builtins.....	190
371	Table 1 – Format for the Builtins lexical element tables.....	23
372	Table 2 – Contents of the lexical element table.....	24
373	Table 3 – Usage of Builtins.....	24
374	Table 4 – User Interface Builtins.....	25
375	Table 5 – Communication Builtins.....	26
376	Table 6 – Action Builtins.....	28
377	Table 7 – Builtin _ERROR.....	28
378	Table 8 – Builtin _TRACE.....	29
379	Table 9 – Builtin _WARNING.....	29
380	Table 10 – Builtin abort.....	29
381	Table 11 – Builtin abort_on_all_comm_errors.....	30
382	Table 12 – Builtin ABORT_ON_ALL_COMM_STATUS.....	30
383	Table 13 – Builtin ABORT_ON_ALL_DEVICE_STATUS.....	31
384	Table 14 – Builtin ABORT_ON_ALL_RESPONSE_CODES.....	31
385	Table 15 – Builtin abort_on_all_response_codes.....	32
386	Table 16 – Builtin ABORT_ON_COMM_ERROR.....	32
387	Table 17 – Builtin abort_on_comm_error.....	33

388	Table 18 – Builtin ABORT_ON_COMM_STATUS	33
389	Table 19 – Builtin ABORT_ON_DEVICE_STATUS	34
390	Table 20 – Builtin ABORT_ON_NO_DEVICE	34
391	Table 21 – Builtin ABORT_ON_RESPONSE_CODE	35
392	Table 22 – Builtin abort_on_response_code	36
393	Table 23 – Builtin abortTransferPort	36
394	Table 24 – Builtin abs	36
395	Table 25 – Builtin ACKNOWLEDGE	37
396	Table 26 – Builtin acknowledge	37
397	Table 27 – Builtin acos	37
398	Table 28 – Builtin add_abort_method	38
399	Table 29 – Builtin add_abort_method	38
400	Table 30 – Builtin AddTime	39
401	Table 31 – Builtin asin	39
402	Table 32 – Builtin assign	40
403	Table 33 – Builtin assign_double	40
404	Table 34 – Builtin assign_float	40
405	Table 35 – Builtin assign_int	41
406	Table 36 – Builtin assign_var	41
407	Table 37 – Builtin assign2	41
408	Table 38 – Builtin atan	42
409	Table 39 – Builtin atof	42
410	Table 40 – Builtin atoi	43
411	Table 41 – Builtin browselidentity	43
412	Table 42 – Builtin BUILD_MESSAGE	43
413	Table 43 – Builtin ByteToDouble	44
414	Table 44 – Builtin ByteToFloat	44
415	Table 45 – Builtin ByteToLong	45
416	Table 46 – Builtin ByteToShort	45
417	Table 47 – Builtin cbrt	45
418	Table 48 – Builtin ceil	46
419	Table 49 – Builtin closeTransferPort	46
420	Table 50 – Builtin cos	46
421	Table 51 – Builtin cosh	47
422	Table 52 – Builtin dassign	47
423	Table 53 – Builtin DATE_AND_TIME_VALUE_to_string	47
424	Table 54 – Builtin Date_to_DayOfMonth	48
425	Table 55 – Builtin DATE_to_days	48
426	Table 56 – Builtin Date_to_Month	48
427	Table 57 – Builtin DATE_to_string	49
428	Table 58 – Builtin Date_To_Time	49
429	Table 59 – Builtin Date_to_Year	49
430	Table 60 – Builtin days_to_DATE	50

431	Table 61 – Builtin DELAY	50
432	Table 62 – Builtin delay	50
433	Table 63 – Builtin DELAY_TIME	51
434	Table 64 – Builtin delayfor	51
435	Table 65 – Builtin delayfor2	52
436	Table 66 – Builtin DICT_ID	52
437	Table 67 – Builtin dictionary_string	53
438	Table 68 – Builtin DiffTime	53
439	Table 69 – Builtin discard_on_exit	54
440	Table 70 – Builtin DISPLAY	54
441	Table 71 – Builtin display	55
442	Table 72 – Builtin display_bitenum	55
443	Table 73 – Builtin display_builtin_error	55
444	Table 74 – Builtin display_comm_error	56
445	Table 75 – Builtin display_comm_status	56
446	Table 76 – Builtin display_device_status	57
447	Table 77 – Builtin display_dynamics	57
448	Table 78 – Builtin display_dynamics2	58
449	Table 79 – Builtin display_message	59
450	Table 80 – Builtin display_message2	59
451	Table 81 – Builtin display_response_code	60
452	Table 82 – Builtin display_response_status	60
453	Table 83 – Builtin display_xmtr_status	61
454	Table 84 – Builtin DoubleToByte	61
455	Table 85 – Builtin drand	62
456	Table 86 – Builtin dseed	62
457	Table 87 – Builtin edit_device_value	63
458	Table 88 – Builtin edit_device_value2	64
459	Table 89 – Builtin edit_local_value	64
460	Table 90 – Builtin edit_local_value2	65
461	Table 91 – Builtin exp	66
462	Table 92 – Builtin ext_send_command	66
463	Table 93 – Builtin ext_send_command_trans	67
464	Table 94 – Builtin fail_on_all_comm_errors	67
465	Table 95 – Builtin fail_on_all_response_codes	68
466	Table 96 – Builtin fail_on_comm_error	68
467	Table 97 – Builtin fail_on_response_code	69
468	Table 98 – Builtin fassign	69
469	Table 99 – Builtin fGetByte	70
470	Table 100 – Builtin fgetval	70
471	Table 101 – Builtin float_value	70
472	Table 102 – Builtin FloatToByte	71
473	Table 103 – Builtin floor	71

474	Table 104 – Builtin fmod	71
475	Table 105 – Builtin fpclassifyd	72
476	Table 106 – Builtin From_DATE_AND_TIME_VALUE	72
477	Table 107 – Builtin From_TIME_VALUE	73
478	Table 108 – Builtin fsetval	73
479	Table 109 – Builtin ftoa	73
480	Table 110 – Builtin fvar_value	74
481	Table 111 – Builtin get_acknowledgement	74
482	Table 112 – Builtin get_acknowledgement2	74
483	Table 113 – Builtin get_block_instance_by_object_index	75
484	Table 114 – Builtin get_block_instance_by_tag	75
485	Table 115 – Builtin get_block_instance_count	76
486	Table 116 – Builtin get_comm_error	77
487	Table 117 – Builtin get_comm_error_string	77
488	Table 118 – Builtin get_date	78
489	Table 119 – Builtin get_date_lelem	78
490	Table 120 – Builtin get_date_lelem2	78
491	Table 121 – Builtin get_date_value	79
492	Table 122 – Builtin get_date_value2	80
493	Table 123 – Builtin GET_DD_REVISION	80
494	Table 124 – Builtin get_dds_error	81
495	Table 125 – Builtin GET_DEV_VAR_VALUE	81
496	Table 126 – Builtin get_dev_var_value	82
497	Table 127 – Builtin GET_DEVICE_REVISION	82
498	Table 128 – Builtin GET_DEVICE_TYPE	82
499	Table 129 – Builtin get_dictionary_string	83
500	Table 130 – Builtin get_double	83
501	Table 131 – Builtin get_double_lelem	83
502	Table 132 – Builtin get_double_lelem2	84
503	Table 133 – Builtin get_double_value	85
504	Table 134 – Builtin get_double_value2	85
505	Table 135 – Builtin get_enum_string	86
506	Table 136 – Builtin get_float	86
507	Table 137 – Builtin get_float_lelem	86
508	Table 138 – Builtin get_float_lelem2	87
509	Table 139 – Builtin get_float_value	87
510	Table 140 – Builtin get_float_value2	88
511	Table 141 – Builtin GET_LOCAL_VAR_VALUE	88
512	Table 142 – Builtin get_local_var_value	89
513	Table 143 – Builtin GET_MANUFACTURER	89
514	Table 144 – Builtin get_more_status	90
515	Table 145 – Builtin get_resolve_status	90
516	Table 146 – Builtin get_response_code	91

517	Table 147 – Builtin get_response_code_string	91
518	Table 148 – Builtin get_rspcode_string	92
519	Table 149 – Builtin get_rspcode_string_by_id	92
520	Table 150 – Builtin get_signed	93
521	Table 151 – Builtin get_signed_lelem	93
522	Table 152 – Builtin get_signed_lelem2	93
523	Table 153 – Builtin get_signed_value	94
524	Table 154 – Builtin get_signed_value2	95
525	Table 155 – Builtin get_status_code_string	95
526	Table 156 – Builtin get_status_string	96
527	Table 157 – Builtin get_stddict_string	96
528	Table 158 – Builtin get_string	97
529	Table 159 – Builtin get_string_lelem	97
530	Table 160 – Builtin get_string_lelem2	98
531	Table 161 – Builtin get_string_value	98
532	Table 162 – Builtin get_string_value2	99
533	Table 163 – Builtin GET_TICK_COUNT	99
534	Table 164 – Builtin get_transfer_status	100
535	Table 165 – Builtin get_unsigned	100
536	Table 166 – Builtin get_unsigned_lelem	101
537	Table 167 – Builtin get_unsigned_lelem2	101
538	Table 168 – Builtin get_unsigned_value	102
539	Table 169 – Builtin get_unsigned_value2	102
540	Table 170 – Builtin get_variable_string	103
541	Table 171 – Builtin GetCurrentDate	103
542	Table 172 – Builtin GetCurrentDateAndTime	104
543	Table 173 – Builtin GetCurrentTime	104
544	Table 174 – Builtin iassign	104
545	Table 175 – Builtin igetval	105
546	Table 176 – Builtin IGNORE_ALL_COMM_STATUS	105
547	Table 177 – Builtin IGNORE_ALL_DEVICE_STATUS	106
548	Table 178 – Builtin IGNORE_ALL_RESPONSE_CODES	106
549	Table 179 – Builtin IGNORE_COMM_ERROR	107
550	Table 180 – Builtin IGNORE_COMM_STATUS	107
551	Table 181 – Builtin IGNORE_DEVICE_STATUS	108
552	Table 182 – Builtin IGNORE_NO_DEVICE	108
553	Table 183 – Builtin IGNORE_RESPONSE_CODE	109
554	Table 184 – Builtin int_value	109
555	Table 185 – Builtin is_NaN	109
556	Table 186 – Builtin isetval	110
557	Table 187 – Builtin isOffline	110
558	Table 188 – Builtin ITEM_ID	110
559	Table 189 – Builtin itoa (version A)	111

560	Table 190 – Builtin itoa (version B)	111
561	Table 191 – Builtin ivar_value	111
562	Table 192 – Builtin lassign	112
563	Table 193 – Builtin lgetval	112
564	Table 194 – Builtin ListDeleteElementAt	112
565	Table 195 – Builtin ListDeleteElementAt2	113
566	Table 196 – Builtin ListInsert	113
567	Table 197 – Builtin ListInsert2	114
568	Table 198 – Builtin log	114
569	Table 199 – Builtin LOG_MESSAGE	115
570	Table 200 – Builtin log10	115
571	Table 201 – Builtin log2	115
572	Table 202 – Builtin long_value	116
573	Table 203 – Builtin LongToByte	116
574	Table 204 – Builtin lsetval	116
575	Table 205 – Builtin lvar_value	117
576	Table 206 – Builtin Make_Time	117
577	Table 207 – Builtin MEMBER_ID	118
578	Table 208 – Builtin MenuDisplay	118
579	Table 209 – Builtin method_abort	119
580	Table 210 – Builtin nan	119
581	Table 211 – Builtin NaN_value	119
582	Table 212 – Builtin nanf	120
583	Table 213 – Builtin ObjectReference	120
584	Table 214 – Builtin openTransferPort	121
585	Table 215 – Builtin pop_abort_method	121
586	Table 216 – Builtin pow	121
587	Table 217 – Builtin process_abort	122
588	Table 218 – Builtin push_abort_method	122
589	Table 219 – Builtin put_date	122
590	Table 220 – Builtin put_date_value	123
591	Table 221 – Builtin put_date_value2	124
592	Table 222 – Builtin put_double	124
593	Table 223 – Builtin put_double_value	125
594	Table 224 – Builtin put_double_value2	125
595	Table 225 – Builtin put_float	126
596	Table 226 – Builtin put_float_value	126
597	Table 227 – Builtin put_float_value2	127
598	Table 228 – Format Options	128
599	Table 229 – Builtin PUT_MESSAGE	128
600	Table 230 – Builtin put_message	129
601	Table 231 – Builtin put_signed	129
602	Table 232 – Builtin put_signed_value	129

603	Table 233 – Builtin put_signed_value2.....	130
604	Table 234 – Builtin put_string	131
605	Table 235 – Builtin put_string_value	131
606	Table 236 – Builtin put_string_value2	132
607	Table 237 – Builtin put_unsigned	133
608	Table 238 – Builtin put_unsigned_value	133
609	Table 239 – Builtin put_unsigned_value2	134
610	Table 240 – Builtin re_read_file	135
611	Table 241 – Builtin re_write_file	135
612	Table 242 – Builtin read_value.....	135
613	Table 243 – Builtin read_value2.....	136
614	Table 244 – Builtin ReadCommand	136
615	Table 245 – Builtin readItemFromDevice.....	137
616	Table 246 – Builtin remove_abort_method	137
617	Table 247 – Builtin remove_abort_method	137
618	Table 248 – Builtin remove_all_abort_methods	138
619	Table 249 – Builtin resolve_array_ref.....	138
620	Table 250 – Builtin resolve_block_ref	139
621	Table 251 – Builtin resolve_block_ref2.....	139
622	Table 252 – Builtin resolve_list_ref	140
623	Table 253 – Builtin resolve_local_ref	140
624	Table 254 – Builtin resolve_local_ref2	141
625	Table 255 – Builtin resolve_param_list_ref.....	141
626	Table 256 – Builtin resolve_param_ref	142
627	Table 257 – Builtin resolve_param_ref2	142
628	Table 258 – Builtin resolve_record_ref.....	143
629	Table 259 – Builtin resolve_record_ref2.....	144
630	Table 260 – Builtin ret_double_value	144
631	Table 261 – Builtin ret_double_value2	144
632	Table 262 – Builtin ret_float_value	145
633	Table 263 – Builtin ret_float_value2	145
634	Table 264 – Builtin ret_signed_value	146
635	Table 265 – Builtin ret_signed_value2	146
636	Table 266 – Builtin ret_unsigned_value	147
637	Table 267 – Builtin ret_unsigned_value2.....	147
638	Table 268 – Builtin retry_on_all_comm_errors	148
639	Table 269 – Builtin RETRY_ON_ALL_COMM_STATUS	148
640	Table 270 – Builtin RETRY_ON_ALL_DEVICE_STATUS	148
641	Table 271 – Builtin RETRY_ON_ALL_RESPONSE_CODES.....	149
642	Table 272 – Builtin retry_on_all_response_codes	149
643	Table 273 – Builtin RETRY_ON_COMM_ERROR.....	150
644	Table 274 – Builtin retry_on_comm_error.....	150
645	Table 275 – Builtin RETRY_ON_COMM_STATUS	151

646	Table 276 – Builtin RETRY_ON_DEVICE_STATUS	151
647	Table 277 – Builtin RETRY_ON_NO_DEVICE.....	152
648	Table 278 – Builtin RETRY_ON_RESPONSE_CODE	152
649	Table 279 – Builtin retry_on_response_code.....	153
650	Table 280 – Builtin round	153
651	Table 281 – Builtin save_on_exit	154
652	Table 282 – Builtin save_values.....	154
653	Table 283 – Builtin seconds_to_TIME_VALUE	154
654	Table 284 – Builtin SELECT_FROM_LIST	155
655	Table 285 – Builtin select_from_list	156
656	Table 286 – Builtin select_from_menu	156
657	Table 287 – Builtin select_from_menu2	157
658	Table 288 – Builtin send	158
659	Table 289 – Builtin send_all_values	158
660	Table 290 – Builtin send_command	159
661	Table 291 – Builtin send_command_trans.....	159
662	Table 292 – Builtin send_on_exit	160
663	Table 293 – Builtin send_trans.....	160
664	Table 294 – Builtin send_value	161
665	Table 295 – Builtin send_value2	161
666	Table 296 – Builtin SET_NUMBER_OF_RETRIES	162
667	Table 297 – Builtin sgetval.....	162
668	Table 298 – Builtin ShortToByte.....	162
669	Table 299 – Builtin sin	163
670	Table 300 – Builtin sinh	163
671	Table 301 – Builtin sqrt	163
672	Table 302 – Builtin ssetval	164
673	Table 303 – Builtin strcmp	164
674	Table 304 – Builtin strleft	164
675	Table 305 – Builtin strlen	165
676	Table 306 – Builtin strlwr	165
677	Table 307 – Builtin strmid	165
678	Table 308 – Builtin strright	166
679	Table 309 – Builtin strstr	166
680	Table 310 – Builtin strtrim	166
681	Table 311 – Builtin strupr.....	167
682	Table 312 – Builtin tan	167
683	Table 313 – Builtin tanh	167
684	Table 314 – Builtin Time_To_Date	168
685	Table 315 – Builtin TIME_VALUE_to_Hour	168
686	Table 316 – Builtin TIME_VALUE_to_Minute	168
687	Table 317 – Builtin TIME_VALUE_to_Second	169
688	Table 318 – Builtin TIME_VALUE_to_seconds.....	169

689	Table 319 – Builtin TIME_VALUE_to_string	169
690	Table 320 – Builtin timet_to_string	170
691	Table 321 – Builtin timet_to_TIME_VALUE	170
692	Table 322 – Builtin To_Date	171
693	Table 323 – Builtin To_Date_and_Time	171
694	Table 324 – Builtin To_Time	171
695	Table 325 – Builtin To_TIME_VALUE	172
696	Table 326 – Builtin trunc	172
697	Table 327 – Builtin VARID	173
698	Table 328 – Builtin vassign	173
699	Table 329 – Builtin WriteCommand	173
700	Table 330 – Builtin writeItemToDevice	174
701	Table 331 – Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	174
702	Table 332 – Builtin XMTR_ABORT_ON_ALL_DATA	175
703	Table 333 – Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS	175
704	Table 334 – Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES	176
705	Table 335 – Builtin XMTR_ABORT_ON_COMM_ERROR	176
706	Table 336 – Builtin XMTR_ABORT_ON_COMM_STATUS	177
707	Table 337 – Builtin XMTR_ABORT_ON_DATA	177
708	Table 338 – Builtin XMTR_ABORT_ON_DEVICE_STATUS	178
709	Table 339 – Builtin XMTR_ABORT_ON_NO_DEVICE	178
710	Table 340 – Builtin XMTR_ABORT_ON_RESPONSE_CODE	179
711	Table 341 – Builtin XMTR_IGNORE_ALL_COMM_STATUS	179
712	Table 342 – Builtin XMTR_IGNORE_ALL_DATA	180
713	Table 343 – Builtin XMTR_IGNORE_ALL_DEVICE_STATUS	180
714	Table 344 – Builtin XMTR_IGNORE_ALL_RESPONSE_CODES	181
715	Table 345 – Builtin XMTR_IGNORE_COMM_ERROR	181
716	Table 346 – Builtin XMTR_IGNORE_COMM_STATUS	182
717	Table 347 – Builtin XMTR_IGNORE_DEVICE_STATUS	182
718	Table 348 – Builtin XMTR_IGNORE_NO_DEVICE	183
719	Table 349 – Builtin XMTR_IGNORE_RESPONSE_CODE	183
720	Table 350 – Builtin XMTR_RETRY_ON_ALL_COMM_STATUS	184
721	Table 351 – Builtin XMTR_RETRY_ON_ALL_DATA	184
722	Table 352 – Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS	185
723	Table 353 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES	185
724	Table 354 – Builtin XMTR_RETRY_ON_COMM_ERROR	186
725	Table 355 – Builtin XMTR_RETRY_ON_COMM_STATUS	186
726	Table 356 – Builtin XMTR_RETRY_ON_DATA	187
727	Table 357 – Builtin XMTR_RETRY_ON_DEVICE_STATUS	187
728	Table 358 – Builtin XMTR_RETRY_ON_NO_DEVICE	188
729	Table 359 – Builtin XMTR_RETRY_ON_RESPONSE_CODE	188
730	Table 360 – Contents of the return codes description table	188
731	Table 361 – Return code descriptions	188

732	Table 362 – Return code descriptions	189
733	Table A.1 – Builtin profile for PI	190
734	Table A.2 – Builtin profile for Fieldbus Foundation	195
735	Table A.3 – Builtin profile for HCF	200
736	Table A.4 – Builtin profile for Communication Servers	205
737		
738		

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**Devices and integration in enterprise systems –
Function blocks (FB) for process control and EDDL –****Part 5: EDDL Builtin library**

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61804-5 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this standard is based on the following documents:

FDIS	Report on voting
65E/XXX/FDIS	65E/XXX/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

790 The committee has decided that the contents of this publication will remain unchanged until
791 the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data
792 related to the specific publication. At this date, the publication will be

- 793 • reconfirmed,
- 794 • withdrawn,
- 795 • replaced by a revised edition, or
- 796 • amended.

797 A bilingual version of this publication may be issued at a later date.

798 The National Committees are requested to note that for this publication the stability date
799 is 2016.

800 THIS TEXT IS INCLUDED FOR THE INFORMATION OF THE NATIONAL COMMITTEES AND WILL BE
801 DELETED AT THE PUBLICATION STAGE.

802

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

803

804

805

806

INTRODUCTION

807 This part of the IEC 61804 is an extension to IEC 61804-3. In earlier editions of IEC 61804-3
808 the EDDL Builtin library was part of the Electronic Device Description Language (EDDL)
809 document IEC 61804-3. Based on editorial problems, the EDDL Builtin library is now moved to
810 this separate part of IEC 61804.

811 .

812

**Devices and integration in enterprise systems –
Function blocks (FB) for process control and EDDL –**

Part 5: EDDL Builtin library

1 Scope

This part of IEC 61804 specifies the EDDL Builtin library and provides the profiles of the various fieldbuses.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61804-3:2013, *Devices and integration in enterprise systems – Function blocks (FB) for process control and EDDL – Electronic Device Description Language*

3 Terms, definitions, acronyms and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61804-3 and the following apply.

3.2 Acronyms and abbreviated terms

CP	Communication Profile
CPF	Communication Profile Family
DD	Device Description
EDD	Electronic Device Description
EDDL	Electronic Device Description Language
HCF	HART® ¹ Communication Foundation
I/O	Input/Output
ID	Identifier
PI	PROFIBUS & PROFINET ² International
PNO	PROFIBUS Nutzerorganisation

832

4 EDDL Builtin library**4.1 General**

This Clause 4 describes Builtin subroutines, which can be used in an EDD. The Builtins support user or device interactions, see Annex A. Typical user interactions are, for example, user inputs of VARIABLE values or acknowledgments. Typical device interactions are, for example, send commands to the device or interpreting response codes and status bytes. This Clause 4 contains the library of Builtins.

In addition to the Builtins specified in this document, user-specific Builtins can be used. These Builtins shall be integrated into the EDD application.

NOTE The Builtins are listed in alphabetical order.

4.2 Conventions for Builtin descriptions

The following conventions apply for the Builtin definitions.

- Table 1 defines the format for the Builtins lexical element tables.
- Table 2 defines the contents of the columns of Table 1.

Table 1 – Format for the Builtins lexical element tables

Parameter Name	Type	Direction	Usage	Description

848

¹ HART® is a registered trademark of the consortium HART Communication Foundation (HCF) (non-profit organization). This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name. Use of the trade names requires permission from the trade name holder.

² PROFIBUS & PROFINET are trade names of PROFIBUS Nutzerorganisation e.V. (PNO) as part of PROFIBUS & PROFINET International. PNO is a non-profit trade organization to support the fieldbus technologies PROFIBUS & PROFINET. This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the tradename. Use of the trade names requires permission from the trade name holder.

849

Table 2 – Contents of the lexical element table

Column	Text	Meaning
Parameter name	<name>	The name of the formal parameter.
	"<return>"	The return value of the Builtin.
Type	data type	Abstract data type (basic or derived) of the formal parameter.
Direction	I	An input parameter.
	O	An output parameter.
	I/O	An input and output parameter.
	—	There is no returned value (VOID).
Usage	m	Formal parameter is mandatory.
	o	Formal parameter is optional.
	c	Formal parameter is conditional.
	—	There is no returned value (VOID).
Description	<text>	Descriptive text for the parameter..
	—	There is no returned value (VOID).

850

851 **4.3 Builtin Categories**852 **4.3.1 Overview**

853 There are the following special categories of builtins:

854 • User Interface Builtins

855 • Communication Builtins

856 • Action Builtins

857 Restrictions apply to the usage of builtins, depending on their category.

858

Table 3 – Usage of Builtins

Builtin Usage	User Interface Builtins	Communication Builtins	Action Builtins
methods referenced in user interface menus	allowed	allowed	not allowed
PRE_EDIT_ACTIONS	allowed	not recommended	allowed ^a
POST_EDIT_ACTIONS	allowed	not recommended	allowed ^a
REFRESH_ACTIONS	not recommended	GRAPH and CHART: allowed others: not recommended	allowed ^a
PRE_READ_ACTIONS	not recommended	allowed	allowed ^a
POST_READ_ACTIONS	not recommended	allowed	allowed ^a
PRE_WRITE_ACTIONS	not recommended	allowed	allowed ^a
POST_WRITE_ACTIONS	not recommended	allowed	allowed ^a
INIT_ACTIONS	not recommended	allowed	not allowed
EXIT_ACTIONS	not recommended	allowed	not allowed
^a Shall only be used for actions called from a VARIABLE			

859

860 **4.3.2 User Interface Builtins**

861 User interface builtins display information and in addition some of them require interaction
 862 from the user.

863 **Table 4 – User Interface Builtins**

Builtin name
ACKNOWLEDGE
acknowledge
DELAY
delay
DISPLAY
display
display_bitenum
display_builtin_error
display_comm_error
display_comm_status
display_device_status
display_dynamics
display_dynamics2
display_message
display_message2
display_response_code
display_response_status
display_xmtr_status
edit_device_value
edit_device_value2
edit_local_value
edit_local_value2
get_acknowledgement2
get_acknowledgement
GET_DEV_VAR_VALUE
get_dev_var_value
GET_LOCAL_VAR_VALUE
get_local_var_value
MenuDisplay
PUT_MESSAGE
put_message
SELECT_FROM_LIST
select_from_list
select_from_menu
select_from_menu2

864

865

866 **4.3.3 Communication Builtins**

867 The communication builtins allow methods to communicate with the device or control the
 868 communication with the device.

869 **Table 5 – Communication Builtins**

Builtin name
abort_on_all_comm_errors
ABORT_ON_ALL_COMM_STATUS
ABORT_ON_ALL_DEVICE_STATUS
ABORT_ON_ALL_RESPONSE_CODES
abort_on_all_response_codes
ABORT_ON_COMM_ERROR
abort_on_comm_error
ABORT_ON_COMM_STATUS
ABORT_ON_DEVICE_STATUS
ABORT_ON_NO_DEVICE
ABORT_ON_RESPONSE_CODE
abort_on_response_code
ext_send_command
ext_send_command_trans
IGNORE_ALL_COMM_STATUS
IGNORE_ALL_DEVICE_STATUS
IGNORE_ALL_RESPONSE_CODES
IGNORE_COMM_ERROR
IGNORE_COMM_STATUS
IGNORE_DEVICE_STATUS
IGNORE_NO_DEVICE
IGNORE_RESPONSE_CODE
read_value
read_value2
ReadCommand
retry_on_all_comm_errors
RETRY_ON_ALL_COMM_STATUS
RETRY_ON_ALL_DEVICE_STATUS
RETRY_ON_ALL_RESPONSE_CODES
retry_on_all_response_codes
RETRY_ON_COMM_ERROR
retry_on_comm_error
RETRY_ON_COMM_STATUS
RETRY_ON_DEVICE_STATUS
RETRY_ON_NO_DEVICE
RETRY_ON_RESPONSE_CODE
retry_on_response_code
send
send_all_values

Builtin name
send_command
send_command_trans
send_on_exit
send_trans
send_value
send_value2
WriteCommand
XMTR_ABORT_ON_ALL_COMM_STATUS
XMTR_ABORT_ON_ALL_DATA
XMTR_ABORT_ON_ALL_DEVICE_STATUS
XMTR_ABORT_ON_ALL_RESPONSE_CODES
XMTR_ABORT_ON_COMM_ERROR
XMTR_ABORT_ON_COMM_STATUS
XMTR_ABORT_ON_DATA
XMTR_ABORT_ON_DEVICE_STATUS
XMTR_ABORT_ON_NO_DEVICE
XMTR_ABORT_ON_RESPONSE_CODE
XMTR_IGNORE_ALL_COMM_STATUS
XMTR_IGNORE_ALL_DATA
XMTR_IGNORE_ALL_DEVICE_STATUS
XMTR_IGNORE_ALL_RESPONSE_CODES
XMTR_IGNORE_COMM_ERROR
XMTR_IGNORE_COMM_STATUS
XMTR_IGNORE_DEVICE_STATUS
XMTR_IGNORE_NO_DEVICE
XMTR_IGNORE_RESPONSE_CODE
XMTR_RETRY_ON_ALL_COMM_STATUS
XMTR_RETRY_ON_ALL_DATA
XMTR_RETRY_ON_ALL_DEVICE_STATUS
XMTR_RETRY_ON_ALL_RESPONSE_CODE
XMTR_RETRY_ON_ALL_RESPONSE_CODES
XMTR_RETRY_ON_COMM_ERROR
XMTR_RETRY_ON_COMM_STATUS
XMTR_RETRY_ON_DATA
XMTR_RETRY_ON_DEVICE_STATUS
XMTR_RETRY_ON_NO_DEVICE
XMTR_RETRY_ON_RESPONSE_CODE

870

871 **4.3.4 Action Builtins**

872 The action builtins allow for accessing the value of the variable from which the action was
873 called. These builtins may not be used in other methods or in actions called from a MENU.

874

Table 6 – Action Builtins

Builtin name
fgetval
fsetval
get_date
get_double
get_float
get_signed
get_string
get_unsigned
igetval
isetval
put_date
put_double
put_float
put_signed
put_string
put_unsigned
sgetval
ssetval

875

876

877 4.4 Builtin _ERROR**878 Purpose**

879 The Builtin _ERROR puts a textual information in a log. This information is put into the
 880 category error. The Builtin _TRACE and the Builtin _WARNING write information to this same
 881 log.

882 Any variable reference contained in the format string will be expanded to the current value.
 883 Method local variables and VARIABLES can be referenced in the message parameter. For
 884 more information, refer to the Builtin PUT_MESSAGE.

885 Lexical structure

886 _ERROR(text)

887 The lexical elements of the Builtin _ERROR are shown in Table 7.

888

Table 7 – Builtin _ERROR

Parameter name	Type	Direction	Usage	Description
text	DD_STRING	I	m	The textual information to be put into the log.
<return>	VOID	—	—	—

889

890 4.5 Builtin _TRACE**891 Purpose**

892 The Builtin _TRACE puts a textual information in a log. This information is put into the
 893 category trace. The Builtin _ERROR and the Builtin _WARNING write information to this same
 894 log.

Any variable reference contained in the format string will be expanded to the current value. Method local variables and VARIABLES can be referenced in the message parameter. For more information, refer to the Builtin PUT_MESSAGE.

Lexical structure

`_TRACE(text)`

The lexical elements of the Builtin `_TRACE` are shown in Table 8.

Table 8 – Builtin `_TRACE`

Parameter name	Type	Direction	Usage	Description
text	DD_STRING	I	m	The textual information to be put into the log.
<return>	VOID	—	—	—

4.6 Builtin `_WARNING`

Purpose

The Builtin `_WARNING` puts a textual information in a log. This information is put into the category warning. The Builtin `_ERROR` and the Builtin `_TRACE` write information to this same log.

Any variable reference contained in the format string will be expanded to the current value. Method local variables and VARIABLES can be referenced in the message parameter. For more information, refer to the Builtin PUT_MESSAGE.

Lexical structure

`_WARNING(text)`

The lexical elements of the Builtin `_WARNING` are shown in Table 9.

Table 9 – Builtin `_WARNING`

Parameter name	Type	Direction	Usage	Description
text	DD_STRING	I	m	The textual information to be put into the log.
<return>	VOID	—	—	—

4.7 Builtin abort

Purpose

The Builtin abort will display a message indicating that the method has been aborted and wait for acknowledgement from the user. Once acknowledgement has been made, the system will execute any abort methods in the abort method list and will exit the method.

Lexical structure

`abort(VOID)`

The lexical elements of the Builtin abort are shown in Table 10.

Table 10 – Builtin abort

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

4.8 Builtin abort_on_all_comm_errors**Purpose**

The Builtin abort_on_all_comm_errors sets the method to abort upon receiving any communication error.

When a method is started, it does not abort on a communication error until abort_on_all_comm_errors is called. This frees a method from having to handle any communication error codes by aborting the method if it receives any communications error.

After this call, any Builtin call which makes a communication request that returns an error will cause the method to abort. A communication error of zero is not considered a communication error.

The communication error codes are defined by the consortia. Their related strings may be defined in a text dictionary.

Lexical structure

abort_on_all_comm_errors(VOID)

The lexical element of the Builtin abort_on_all_comm_errors is shown in Table 11.

Table 11 – Builtin abort_on_all_comm_errors

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

4.9 Builtin ABORT_ON_ALL_COMM_STATUS**Purpose**

The Builtin ABORT_ON_ALL_COMM_STATUS will set all of the bits in the comm status abort mask. This will cause the system to abort the current method if the device returns any comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks, and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_ALL_COMM_STATUS(VOID)

The lexical element of the Builtin ABORT_ON_ALL_COMM_STATUS is shown in Table 12.

Table 12 – Builtin ABORT_ON_ALL_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

960 4.10 Builtin ABORT_ON_ALL_DEVICE_STATUS

961 Purpose

962 The Builtin ABORT_ON_ALL_DEVICE_STATUS will set all of the bits in the device status
963 abort mask. This will cause the system to abort the current method if the device returns any
964 device status value. Device status is defined to be the second data octet returned in a
965 transaction.

966 The retry and abort masks are reset to their default values at the start of each method, so the
967 new mask value will only be valid during the current method. See the send_command function
968 for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the
969 available masks and their default values.

970 The mask affected by this function is used by the Builtins send, send_trans, send_command,
971 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
972 display.

973 Lexical structure

974 ABORT_ON_ALL_DEVICE_STATUS (VOID)

975 The lexical element of the Builtin ABORT_ON_ALL_DEVICE_STATUS is shown in Table 13.

976 Table 13 – Builtin ABORT_ON_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

977

978 4.11 Builtin ABORT_ON_ALL_RESPONSE_CODES

979 Purpose

980 The Builtin ABORT_ON_ALL_RESPONSE_CODES will set all of the bits in the response code
981 abort mask. This will cause the system to abort the current method if the device returns any
982 response code value.

983 The response code is defined to be the first data octet returned in a transaction, when bit 7 of
984 this octet is 0.

985 The retry and abort masks are reset to their default values at the start of each method, so the
986 new mask value will only be valid during the current method. See the send command function
987 for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the
988 available masks and their default values.

989 The mask affected by this function is used by the Builtins send, send_trans, send_command,
990 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
991 display.

992 Lexical structure

993 ABORT_ON_ALL_RESPONSE_CODES (VOID)

994 The lexical element of the Builtin ABORT_ON_ALL_RESPONSE_CODES is shown in
995 Table 14.

996 Table 14 – Builtin ABORT_ON_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

997

998 **4.12 Builtin abort_on_all_response_codes**999 **Purpose**

1000 The Builtin abort_on_all_response_codes causes the method to abort if it receives any
1001 non-zero response code, after calling a Builtin, which results in a request to a device. This
1002 Builtin frees a method from having to handle any error response codes. When a method is
1003 started, it does not abort on any response code until this Builtin is called. A response code is
1004 a value representing an application-specific error condition. A response code value of zero is
1005 not considered as an error response code. Each response code and response code type is
1006 defined in the EDDL.

1007 **Lexical structure**

1008 abort_on_all_response_codes (VOID)

1009 The lexical element of the Builtin abort_on_all_response_codes is shown in Table 15.

1010 **Table 15 – Builtin abort_on_all_response_codes**

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

1011

1012 **4.13 Builtin ABORT_ON_COMM_ERROR**1013 **Purpose**

1014 The Builtin ABORT_ON_COMM_ERROR will set the no comm error mask in such a way that
1015 the method will be aborted if a comm error is found while sending a command.

1016 The retry and abort masks are reset to their default values at the start of each method, so the
1017 new mask value will only be valid during the current method. See the send_command function
1018 for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the
1019 available masks and their default values.

1020 The mask affected by this function is used by the Builtins send, send_trans, send_command,
1021 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
1022 display.

1023 **Lexical structure**

1024 ABORT_ON_COMM_ERROR (VOID)

1025 The lexical element of the Builtin ABORT_ON_COMM_ERROR is shown in Table 16.

1026 **Table 16 – Builtin ABORT_ON_COMM_ERROR**

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

1027

1028 **4.14 Builtin abort_on_comm_error**1029 **Purpose**

1030 The Builtin abort_on_comm_error causes the method to abort if it receives a specific
1031 communications error.

1032 When a method is started, it does not abort on a specific communications error until Builtin
1033 abort_on_comm_error is called.

1034 NOTE A communications error of zero is not considered a communications error.

1035 This Builtin permits the method writer to select which communications errors should be
1036 handled by the method and it frees a method from having to handle a specific communications
1037 error code.

1038 The communications error codes are defined by the consortia. Their related strings may be
1039 defined in a text dictionary.

1040 Lexical structure

1041 `abort_on_comm_error(error)`

1042 The lexical elements of the Builtin `abort_on_comm_error` are shown in Table 17.

1043 **Table 17 – Builtin `abort_on_comm_error`**

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	A communications error code.
<return>	long	O	m	One of the return codes specified in Table 361.

1044

1045 4.15 Builtin `ABORT_ON_COMM_STATUS`

1046 Purpose

1047 The Builtin `ABORT_ON_COMM_STATUS` will set the correct bit(s) in the comm status abort
1048 mask in such a way that the specified `comm_status_value` will cause the method to abort.
1049 Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this
1050 octet is 1.

1051 The retry and abort masks are reset to their default values at the start of each method, so the
1052 new mask value will only be valid during the current method. See the `send` command function
1053 for implementation of the masks. See `ABORT_ON_RESPONSE_CODE` for a list of the
1054 available masks and their default values.

1055 The mask affected by this function is used by the Builtins `send`, `send_trans`, `send_command`,
1056 `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and
1057 `display`.

1058 Lexical structure

1059 `ABORT_ON_COMM_STATUS(comm_status)`

1060 The lexical elements of the Builtin `ABORT_ON_COMM_STATUS` are shown in Table 18.

1061 **Table 18 – Builtin `ABORT_ON_COMM_STATUS`**

Parameter name	Type	Direction	Usage	Description
comm_status	unsigned char	I	m	A bit mask for the communication status.
<return>	VOID	—	—	—

1062

1063 4.16 Builtin `ABORT_ON_DEVICE_STATUS`

1064 Purpose

1065 The Builtin `ABORT_ON_DEVICE_STATUS` will set the correct bit(s) in the device status abort
1066 mask such that the specified device status value will cause the method to abort. Device status
1067 is defined to be the second data octet returned in a transaction.

1068 The retry and abort masks are reset to their default values at the start of each method, so the
 1069 new mask value will only be valid during the current method. See the send command function
 1070 for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the
 1071 available masks and their default values.

1072 The mask affected by this function is used by the Builtins send, send_trans, send_command,
 1073 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
 1074 display.

1075 Lexical structure

1076 ABORT_ON_DEVICE_STATUS(device_status)

1077 The lexical elements of the Builtin ABORT_ON_DEVICE_STATUS are shown in Table 19.

1078 **Table 19 – Builtin ABORT_ON_DEVICE_STATUS**

Parameter name	Type	Direction	Usage	Description
device_status	unsigned char	I	m	A bit mask for the device status.
<return>	VOID	—	—	—

1079

1080 4.17 Builtin ABORT_ON_NO_DEVICE

1081 Purpose

1082 The Builtin ABORT_ON_NO_DEVICE will set the no devices mask in such a way that the
 1083 method will be aborted if no device is found while sending a command.

1084 The retry and abort masks are reset to their default values at the start of each method, so the
 1085 new mask value will only be valid during the current method. See the send command function
 1086 for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the
 1087 available masks and their default values.

1088 The mask affected by this function is used by the Builtins send, send_trans, send_command,
 1089 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
 1090 display.

1091 Lexical structure

1092 ABORT_ON_NO_DEVICE(VOID)

1093

1094 The lexical element of the Builtin ABORT_ON_NO_DEVICE is shown in Table 20.

1095 **Table 20 – Builtin ABORT_ON_NO_DEVICE**

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

1096

1097 4.18 Builtin ABORT_ON_RESPONSE_CODE

1098 Purpose

1099 The Builtin ABORT_ON_RESPONSE_CODE will set the correct bit(s) in the response code
 1100 abort mask such that the specified response code value will cause the method to abort. The
 1101 response code is defined to be the first data octet returned in a transaction, when bit 7 of this
 1102 octet is 0.

1103 The retry and abort masks are reset to their default values at the start of each method, so the
1104 new mask value will only be valid during the current method.

1105 The following is a list of all the available masks. The default values of the masks are as
1106 follows:

- 1107 – response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- 1108 – response_code_retry_mask = NO BITS SET
- 1109 – cmd_48_response_code_abort_mask = ACCESS_RESTRICTED |
- 1110 CMD_NOT_IMPLEMENTED
- 1111 – cmd_48_response_code_retry_mask = NO BITS SET
- 1112 – device_status_abort_mask = DEVICE_MALFUNCTION
- 1113 – device_status_retry_mask = NO BITS SET
- 1114 – cmd_48_device_status_abort_mask = DEVICE_MALFUNCTION
- 1115 – cmd_48_device_status_retry_mask = NO BITS SET
- 1116 – comm_status_abort_mask = NO BITS SET
- 1117 – comm_status_retry_mask = ALL BITS SET
- 1118 – cmd_48_comm_status_abort_mask = NO BITS SET
- 1119 – cmd_48_comm_status_retry_mask = ALL BITS SET
- 1120 – no_device_abort_mask = ALL BITS SET
- 1121 – comm_error_abort_mask = ALL BITS SET
- 1122 – comm_error_retry_mask = NO BITS SET
- 1123 – cmd_48_no_device_abort_mask = ALL BITS SET
- 1124 – cmd_48_comm_error_abort_mask = ALL BITS SET
- 1125 – cmd_48_comm_error_retry_mask = NO BITS SET
- 1126 – cmd_48_data_abort_mask = NO BITS SET
- 1127 – cmd_48_data_retry_mask = NO BITS SET

1128 The mask affected by this function is used by the Builtins send, send_trans, send_command,
1129 send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and
1130 display.

1131 Lexical structure

1132 ABORT_ON_RESPONSE_CODE(response_code)

1133 The lexical elements of the Builtin ABORT_ON_RESPONSE_CODE are shown in Table 21.

1134 **Table 21 – Builtin ABORT_ON_RESPONSE_CODE**

Parameter name	Type	Direction	Usage	Description
response_code	int	l	m	A bit mask for the response code.
<return>	VOID	—	—	—

1135

1136 4.19 Builtin abort_on_response_code

1137 Purpose

1138 The Builtin abort_on_response_code causes the method to abort upon receiving the specified
1139 response code.

1140 The Builtin causes the method to abort if it receives a specific response code. This Builtin
1141 frees a method from having to handle a specific response code.

1142 When a method is started, it will not abort on a specific response code until this Builtin is
1143 called. A response code is a value representing an application-specific error condition.

1144 NOTE A response code value of zero is not considered an error response code. Each response code type is
1145 defined in the EDD.

1146 Lexical structure

1147 `abort_on_response_code (code)`

1148 The lexical elements of the Builtin `abort_on_response_code` are shown in Table 22.

1149 **Table 22 – Builtin `abort_on_response_code`**

Parameter name	Type	Direction	Usage	Description
code	unsigned long	I	m	A bit mask for the response code.
<return>	long	O	m	One of the return codes specified in Table 361.

1150

1151 4.20 Builtin `abortTransferPort`

1152 Purpose

1153 The `abortTransferPort` stops the Block Data Transfer indicated by the handle.

1154 Lexical structure

1155 `abortTransferPort (handle)`

1156 The lexical elements of the Builtin `abortTransferPort` are shown in Table 23.

1157 **Table 23 – Builtin `abortTransferPort`**

Parameter name	Type	Direction	Usage	Description
handle	unsigned long	I	m	The handle for the block transfer
<return>	int	O	m	One of the return codes specified in Table 362.

1158

1159 4.21 Builtin `abs`

1160 Purpose

1161 The Builtin `abs` returns the absolute value of a floating point value.

1162 Lexical structure

1163 `abs (x)`

1164 The lexical elements of the Builtin `abs` are shown in Table 24.

1165 **Table 24 – Builtin `abs`**

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating-point value.
<return>	double	O	m	The absolute value.

1166

1167 4.22 Builtin ACKNOWLEDGE**1168 Purpose**

1169 The Builtin ACKNOWLEDGE will display the prompt and wait for the enter key to be pressed.
1170 Unlike Builtin acknowledge, the prompt may NOT contain embedded device VARIABLES.

1171 Lexical structure

1172 `ACKNOWLEDGE(prompt)`

1173 The lexical elements of the Builtin ACKNOWLEDGE are shown in Table 25.

1174 **Table 25 – Builtin ACKNOWLEDGE**

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	Message to be displayed.
<return>	int	O	m	One of the return codes specified in Table 362.

1175

1176 4.23 Builtin acknowledge**1177 Purpose**

1178 The Builtin acknowledge will display the prompt and wait for acknowledgment from the user.
1179 The prompt may contain local and/or device variable values.

1180 Lexical structure

1181 `acknowledge(prompt, global_var_ids)`

1182 The lexical elements of the Builtin acknowledge are shown in Table 26.

1183 **Table 26 – Builtin acknowledge**

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	Message to be displayed.
global_var_ids	array of int	I	m	An array that contains unique identifiers of VARIABLE instances.
<return>	int	O	m	One of the return codes specified in Table 362.

1184

1185 4.24 Builtin acos**1186 Purpose**

1187 The Builtin acos returns the arc cosine of a floating-point value.

1188 Lexical structure

1189 `acos(x)`

1190 The lexical elements of the Builtin acos are shown in Table 27.

1191 **Table 27 – Builtin acos**

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating-point value.
<return>	double	O	m	The arc cosine.

1192

4.25 Builtin add_abort_method (version A)**Purpose**

The Builtin add_abort_method will add a method to the abort method list, which is the list of methods to be executed if the current method is aborted. The abort method list can hold up to 20 methods at any one time. The methods are run in the order they are added to the list, and the same method may be added to the list more than once. The list is cleared after each non_abort method is executed.

add_abort_method is only executed when the method is aborted, and not when the method exits under normal operating conditions. Methods can be aborted due to an abort mask condition when sending a command, or when the Builtin abort is called or when the user aborts the method.

Lexical structure

add_abort_method(abort_method_id)

The lexical elements of the Builtin add_abort_method are shown in Table 28.

Table 28 – Builtin add_abort_method

Parameter name	Type	Direction	Usage	Description
abort_method	reference	I	m	A reference to a METHOD.
<return>	int	O	m	One of the return codes specified in Table 362.

4.26 Builtin add_abort_method (version B)**Purpose**

The Builtin add_abort_method adds an abort method to the calling method's abort method list. The abort method list is the list of methods, which will be executed if the method aborts. The abort method list applies only to the current execution of the method.

Methods abort when response codes or communication errors occur which have been set to trigger an abort via other Builtin calls, for example, abort_on_all_comm_errors and abort_on_all_response_codes. Methods can also be explicitly aborted by a direct call to Builtin method_abort, or when an error occurs. The normal completion of a method is not considered an abort.

The abort method list is initially empty when a method begins execution. The abort methods are added to the back of the list and are removed and run from the front of the list in the order they appear, i.e. first-in first-out. An abort method may appear more than once in the list.

Abort methods may be shared between methods, meaning two methods may have the same abort method in their respective abort method lists.

The method_id parameter is specified by the method writer using the Builtin ITEM_ID and the name of the method. Builtin abort methods cannot call add_abort_method.

Lexical structure

add_abort_method(method_id)

The lexical elements of the Builtin add_abort_method are shown in Table 29.

Table 29 – Builtin add_abort_method

Parameter name	Type	Direction	Usage	Description
----------------	------	-----------	-------	-------------

Parameter name	Type	Direction	Usage	Description
method_id	unsigned long	I	m	The ID of the method.
<return>	long	O	m	One of the return codes specified in Table 361.

1230

1231 **4.27 Builtin AddTime**1232 **Purpose**

1233 The Builtin AddTime adds a number of seconds to a time_t value.

1234 **Lexical Structure**

1235 AddTime(t0, seconds)

1236 The lexical elements of the Builtin AddTime are shown in Table 30.

1237

Table 30 – Builtin AddTime

Parameter name	Type	Direction	Usage	Description
t0	time_t	I	m	The initial time_t value.
seconds	double	I	m	The number of seconds to add.
<return>	time_t	O	m	The specified time_t value.

1238

1239 **4.28 Builtin asin**1240 **Purpose**

1241 The Builtin asin returns the arc sine of a floating point value.

1242 **Lexical structure**

1243 asin(x)

1244 The lexical elements of the Builtin asin are shown in Table 31.

1245

Table 31 – Builtin asin

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating point value.
<return>	double	O	m	The arc sine.

1246

1247 **4.29 Builtin assign**1248 **Purpose**1249 The Builtin assign sets the value of a VARIABLE to another VARIABLE's value. It sets the
1250 value of the destination VARIABLE equal to the value of the source VARIABLE.

1251 NOTE The Builtin is a shorthand way of doing an ABN_GET_TYPE_VALUE followed by a put_type_value.

1252 The parameters dst_id, src_id, dst_member_id, and src_member_id are specified using the
1253 Builtins ITEM_ID and MEMBER ID and the reference of the VARIABLE. Both the source
1254 VARIABLEs and the destination VARIABLE shall have the same type.1255 **Lexical structure**

1256 assign(dst_id, dst_member_id, src_id, src_member_id)

1257 The lexical elements of the Builtin assign are shown in Table 32.

1258

Table 32 – Builtin assign

Parameter name	Type	Direction	Usage	Description
dst_id	unsigned long	I	m	The item ID of the destination variable.
dst_member_id	unsigned long	I	m	The member ID of the destination variable.
src_id	unsigned long	I	m	The item ID of the source variable.
src_member_id	unsigned long	I	m	The member ID of the source variable.
<return>	long	O	m	One of the return codes specified in Table 361.

1259

1260 4.30 Builtin assign_double**1261 Purpose**

1262 The Builtin assign_double assigns a value to a VARIABLE.

1263 Lexical structure

1264 assign_double(device_var, new_value)

1265 The lexical elements of the Builtin assign_double are shown in Table 33.

1266

Table 33 – Builtin assign_double

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	The name of the variable.
new_value	double	I	m	The new value of the referenced variable.
<return>	int	O	m	One of the return codes specified in Table 362.

1267

1268 4.31 Builtin assign_float**1269 Purpose**

1270 The Builtin assign_float assigns a value to VARIABLE.

1271 Lexical structure

1272 assign_float(device_var, new_value)

1273 The lexical elements of the Builtin assign_float are shown in Table 34.

1274

Table 34 – Builtin assign_float

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	The name of the variable.
new_value	double	I	m	The new value of the referenced variable.
<return>	int	O	m	One of the return codes specified in Table 362.

1275

1276 4.32 Builtin assign_int**1277 Purpose**

1278 The Builtin assign_int assigns a value to VARIABLE.

1279 **Lexical structure**1280 `assign_int(device_var, new_value)`1281 The lexical elements of the Builtin `assign_int` are shown in Table 35.1282 **Table 35 – Builtin `assign_int`**

Parameter name	Type	Direction	Usage	Description
<code>device_var</code>	reference	I	m	The name of the variable.
<code>new_value</code>	int	I	m	The new value of the referenced variable.
<return>	int	O	m	One of the return codes specified in Table 362.

1283

1284 **4.33 Builtin `assign_var`**1285 **Purpose**1286 The Builtin `assign_var` assigns a value of the source VARIABLE to the destination VARIABLE.

1287 Both variables shall have the same type.

1288 **Lexical structure**1289 `assign_var(destination_var, source_var)`1290 The lexical elements of the Builtin `assign_var` are shown in Table 36.1291 **Table 36 – Builtin `assign_var`**

Parameter name	Type	Direction	Usage	Description
<code>destination_var</code>	reference	I	m	The name of the destination variable.
<code>source_var</code>	reference	I	m	The name of the source variable.
<return>	int	O	m	One of the return codes specified in Table 362.

1292

1293 **4.34 Builtin `assign2`**1294 **Purpose**1295 The Builtin `assign2` sets the value of a VARIABLE to another VARIABLE's value. It sets the
1296 value of the destination VARIABLE equal to the value of the source VARIABLE.1297 The parameters `dst_id`, `src_id`, `dst_member_id`, and `src_member_id` are specified using the
1298 Builtins `ITEM_ID` and `MEMBER ID` and the reference of the VARIABLE. Both the source
1299 VARIABLEs and the destination VARIABLE shall have the same type.1300 **Lexical structure**1301 `assign2(dst_blk_id, dst_blk_num, dst_id, dst_member_id, src_blk_id,`
1302 `src_blk_num, src_id, src_member_id)`1303 The lexical elements of the Builtin `assign2` are shown in Table 37.1304 **Table 37 – Builtin `assign2`**

Parameter name	Type	Direction	Usage	Description
<code>dst_blk_id</code>	unsigned long	I	m	The item ID of the destination block.
<code>dst_blk_num</code>	unsigned long	I	m	The occurrence number of the destination block.
<code>dst_id</code>	unsigned	I	m	The item ID of the destination variable.

Parameter name	Type	Direction	Usage	Description
	long			
dst_member_id	unsigned long	I	m	The member ID of the destination variable.
src_blk_id	unsigned long	I	m	The item ID of the source block.
src_blk_num	unsigned long	I	m	The occurrence number of the source block.
src_id	unsigned long	I	m	The item ID of the source variable.
src_member_id	unsigned long	I	m	The member ID of the source variable.
<return>	long	O	m	One of the return codes specified in Table 361.

1305

1306 **4.35 Builtin atan**1307 **Purpose**

1308 The Builtin atan returns the arc tangent of a floating point value.

1309 **Lexical structure**1310 `atan(x)`

1311 The lexical elements of the Builtin atan are shown in Table 38.

1312

Table 38 – Builtin atan

Parameter Name	Type	Direction	Usage	Description
x	double	I	m	A floating point value.
<return>	double	O	m	The arc tangent.

1313

1314 **4.36 Builtin atof**1315 **Purpose**

1316 The Builtin converts a text into a floating-point value.

1317 **Lexical structure**1318 `atof(str)`

1319 The lexical elements of the Builtin atof are shown in Table 39.

1320

Table 39 – Builtin atof

Parameter name	Type	Direction	Usage	Description
str	char[]	I	m	The text to convert.
<return>	float	O	m	The floating point value.

1321

1322 **4.37 Builtin atoi**1323 **Purpose**

1324 The Builtin converts a text into an integer value.

1325 **Lexical structure**1326 `atoi(str)`

1327 The lexical elements of the Builtin atoi are shown in Table 40.

1328 **Table 40 – Builtin atoi**

Parameter name	Type	Direction	Usage	Description
str	char[]	I	m	The text to convert.
<return>	int	O	m	The integer value.

1329

1330 **4.38 Builtin browselidentity**

1331 **Purpose**

1332 The BLOB declaration may not have a declared IDENTITY. The browselidentity function
1333 allows the user to select the BLOB using a Host Application specific mechanism.

1334 **Lexical structure**

1335 `browseIdentity(blob_item)`

1336 The lexical elements of the Builtin browselidentity are shown in Table 41.

1337 **Table 41 – Builtin browselidentity**

Parameter name	Type	Direction	Usage	Description
blob_item	DD_ITEM &	I	m	The BLOB for which the identity is browsed
<return>	int	O	m	One of the return codes specified in Table 362.

1338

1339 **4.39 Builtin BUILD_MESSAGE**

1340 **Purpose**

1341 The Builtin BUILD_MESSAGE returns a string based on a format specification. Any variable
1342 reference contained in the format string will be expanded to the current value. Method local
1343 variables and VARIABLES can be referenced in the message parameter. For more
1344 information, refer to the Builtin PUT_MESSAGE.

1345 **Lexical structure**

1346 `BUILD_MESSAGE(string)`

1347 The lexical elements of the Builtin BUILD_MESSAGE are shown in Table 42.

1348 **Table 42 – Builtin BUILD_MESSAGE**

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	The format specification.
<return>	DD_STRING	O	m	The formatted string.

1349

1350 **4.40 Builtin ByteToDouble**

1351 **Purpose**

1352 The Builtin ByteToDouble will return a double value from a vector of eight octets.

1353 **Lexical structure**

1354 `ByteToDouble(in1, in2, in3, in4, in5, in6, in7, in8)`

1355 The lexical element of the Builtin ByteToDouble is shown in Table 43.

1356

Table 43 – Builtin ByteToDouble

Parameter name	Type	Direction	Usage	Description
in1	unsigned char	I	m	First element of the double value (most significant byte).
in2	unsigned char	I	m	Second element of the double value.
in3	unsigned char	I	m	Third element of the double value.
in4	unsigned char	I	m	Fourth element of the double value.
in5	unsigned char	I	m	Fifth element of the double value.
in6	unsigned char	I	m	Sixth element of the double value.
in7	unsigned char	I	m	Seventh element of the double value.
in8	unsigned char	I	m	Eighth element of the double value (least significant byte).
<return>	double	O	m	A double value from the given vector of eight octets.

1357

1358 4.41 Builtin ByteToFloat**1359 Purpose**

1360 The Builtin ByteToFloat will return a floating value from a vector of four octets.

1361 Lexical structure

1362 `ByteToFloat(in1, in2, in3, in4)`

1363 The lexical element of the Builtin ByteToFloat is shown in Table 44.

1364

Table 44 – Builtin ByteToFloat

Parameter name	Type	Direction	Usage	Description
in1	unsigned char	I	m	First element of the float value (most significant byte).
in2	unsigned char	I	m	Second element of the float value.
in3	unsigned char	I	m	Third element of the float value.
in4	unsigned char	I	m	Fourth element of the float value (least significant byte).
<return>	float	O	m	A floating point value from the given vector of four octets.

1365

1366 4.42 Builtin ByteToLong**1367 Purpose**

1368 The Builtin ByteToLong will return a long value from a vector of four octets.

1369 Lexical structure

1370 `ByteToLong(in1, in2, in3, in4)`

1371 The lexical element of the Builtin ByteToLong is shown in Table 45.

1372

Table 45 – Builtin ByteToLong

Parameter name	Type	Direction	Usage	Description
in1	unsigned char	I	m	First element of the long value (most significant byte).
in2	unsigned char	I	m	Second element of the long value.
in3	unsigned char	I	m	Third element of the long value.
in4	unsigned char	I	m	Fourth element of the long value (least significant byte).
<return>	long	O	m	A long value from the given vector of four octets.

1373

1374 4.43 Builtin ByteToShort**1375 Purpose**

1376 The Builtin ByteToShort will return a short value from a vector of two octets.

1377 Lexical structure

1378 `ByteToShort(in1, in2)`

1379 The lexical element of the Builtin ByteToShort is shown in Table 46.

1380

Table 46 – Builtin ByteToShort

Parameter name	Type	Direction	Usage	Description
in1	unsigned char	I	m	First element of the short value (most significant byte).
in2	unsigned char	I	m	Second element of the short value (least significant byte).
<return>	short	O	m	A short value from the given vector of two octets.

1381

1382 4.44 Builtin cbrt**1383 Purpose**

1384 The Builtin cbrt returns the cubic root of a floating-point value.

1385 Lexical structure

1386 `cbrt(x)`

1387 The lexical elements of the Builtin cbrt are shown in Table 47.

1388

Table 47 – Builtin cbrt

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating-point value.
<return>	double	O	m	The cubic root.

1389

1390 4.45 Builtin ceil**1391 Purpose**

1392 The Builtin ceil returns the smallest integral value not less than a specified floating-point value.

1393

1394 Lexical structure1395 `ceil(x)`1396 The lexical elements of the Builtin `ceil` are shown in Table 48.1397 **Table 48 – Builtin `ceil`**

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating-point value.
<return>	double	O	m	The ceiling value.

1398

1399 4.46 Builtin `closeTransferPort`**1400 Purpose**1401 The `closeTransferPort` function closes the block transfer indicated by the handle.**1402 Lexical structure**1403 `closeTransferPort(handle)`1404 The lexical elements of the Builtin `closeTransferPort` are shown in Table 49.1405 **Table 49 – Builtin `closeTransferPort`**

Parameter name	Type	Direction	Usage	Description
handle	unsigned long	I	m	The handle for the block transfer
<return>	int	O	m	One of the return codes specified in Table 362.

1406

1407 4.47 Builtin `cos`**1408 Purpose**1409 The Builtin `cos` returns the cosine of a floating point value.**1410 Lexical structure**1411 `cos(x)`1412 The lexical elements of the Builtin `cos` are shown in Table 50.1413 **Table 50 – Builtin `cos`**

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating point value.
<return>	double	O	m	The cosine.

1414

1415 4.48 Builtin `cosh`**1416 Purpose**1417 The Builtin `cosh` returns the hyperbolic cosine of a floating point value.**1418 Lexical structure**1419 `cosh(x)`1420 The lexical elements of the Builtin `cosh` are shown in Table 51.

1421

Table 51 – Builtin cosh

Parameter name	Type	Direction	Usage	Description
x	double	I	m	A floating-point value.
<return>	double	O	m	The hyperbolic cosine.

1422

4.49 Builtin dassign**Purpose**

The Builtin dassign assigns a value to a VARIABLE.

Lexical structure

```
dassign(device_var, new_value)
```

The lexical elements of the Builtin dassign are shown in Table 52.

1429

Table 52 – Builtin dassign

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	The name of the variable.
new_value	double	I	m	The new value of the referenced variable.
<return>	int	O	m	One of the return codes specified in Table 362.

1430

4.50 Builtin DATE_AND_TIME_VALUE_to_string**Purpose**

The Builtin DATE_AND_TIME_VALUE_to_string creates a string representation of a DATE and a TIME_VALUE. The ANSI C strftime function defines the structure of the format string.

Lexical structure

```
DATE_AND_TIME_VALUE_to_string(string, format, date, time_value)
```

The lexical element of the Builtin DATE_AND_TIME_VALUE_to_string is shown in Table 53.

1438

Table 53 – Builtin DATE_AND_TIME_VALUE_to_string

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I/O	m	The string into which the formatted DATE and TIME_VALUE is copied.
format	DD_STRING	I	m	How the DATE and TIME_VALUE are to be formatted.
date	long	I	m	A long value containing the DATE to be formatted.
time_value	unsigned long	I	M	The TIME_VALUE to be formatted.
<return>	int	O	m	The number of character actually copied or -1 if an error occurs.

1439

4.51 Builtin Date_to_DayOfMonth**Purpose**

The Builtin Date_to_DayOfMonth calculates the day of the month from a given number of days since 1 January 1900.

1443

1444 Lexical structure**1445** `Date_to_DayOfMonth(date)`**1446** The lexical elements of the Builtin `Date_to_DayOfMonth` are shown in Table 54.**1447 Table 54 – Builtin `Date_to_DayOfMonth`**

Parameter name	Type	Direction	Usage	Description
date	long	I	m	A long value containing the DATE
<return>	int	O	m	The day of month.

1448**1449 4.52 Builtin `DATE_to_days`****1450 Purpose****1451** The Builtin `DATE_to_days` calculates the number of days between two dates.**1452 Lexical structure****1453** `DATE_to_days(date1, date0)`**1454** The lexical element of the Builtin `DATE_to_days` is shown in Table 55.**1455 Table 55 – Builtin `DATE_to_days`**

Parameter name	Type	Direction	Usage	Description
date1	long	I	m	The more recent date.
date0	long	I	m	The less recent date.
<return>	long	O	m	The number of days between date1 and date0.

1456**1457 4.53 Builtin `Date_to_Month`****1458 Purpose****1459** The Builtin `Date_to_Month` calculates the month from a given number of days since 1 January
1460 1900.**1461 Lexical structure****1462** `Date_to_Month(date)`**1463** The lexical elements of the Builtin `Date_to_Month` are shown in Table 56.**1464 Table 56 – Builtin `Date_to_Month`**

Parameter name	Type	Direction	Usage	Description
date	long	I	m	The number of days. The EDD application shall support an implicit typecast between long and DATE, so this Builtin will also accept DATE types.
<return>	int	O	m	The month.

1465**1466 4.54 Builtin `DATE_to_string`****1467 Purpose****1468** The Builtin `DATE_to_string` creates a string representation of a DATE. The ANSI C `strftime`
1469 function defines the structure of the format string.

1470 **Lexical structure**1471 `DATE_to_string(string, format, date)`1472 The lexical element of the Builtin `DATE_to_string` is shown in Table 57.1473 **Table 57 – Builtin `DATE_to_string`**

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	The string into which formatted DATE is copied.
format	DD_STRING	I	m	How the DATE is to be formatted.
date	long	I	m	A long value containing the DATE
<return>	int	O	m	The number of character actually copied or -1 is an error occurs.

1474

1475 **4.55 Builtin `Date_To_Time`**1476 **Purpose**1477 The Builtin `Date_To_Time` creates a `time_t` from a date value. The time of day is set to
1478 00:00:00.1479 **Lexical Structure**1480 `Date_To_Time(date)`1481 The lexical elements of the Builtin `Date_To_Time` are shown in Table 58.1482 **Table 58 – Builtin `Date_To_Time`**

Parameter name	Type	Direction	Usage	Description
date	long	I	m	A long value containing the DATE
<return>	time_t	O	m	A <code>time_t</code> value.

1483

1484 **4.56 Builtin `Date_to_Year`**1485 **Purpose**1486 The Builtin `Date_to_Year` calculates the year from a given number of days since 1 January
1487 1900.1488 **Lexical structure**1489 `Date_to_Year(date)`1490 The lexical elements of the Builtin `Date_to_Year` are shown in Table 59.1491 **Table 59 – Builtin `Date_to_Year`**

Parameter name	Type	Direction	Usage	Description
date	long	I	m	The number of days. The EDD application shall support an implicit typecast between long and DATE, so this Builtin will also accept DATE types.
<return>	int	O	m	The year (4 digit).

1492

1493 **4.57 Builtin `days_to_DATE`**1494 **Purpose**1495 The Builtin `days_to_DATE` calculates a future date by adding a number of days to a DATE.