

DIN EN 61804-5

ICS 35.060; 35.240.50

**Funktionsbausteine für die Prozessautomation und elektronische
Gerätebeschreibungssprache –
Teil 5: Bibliothek vorgefertigter Unterprogramme
(IEC 61804-5:2015);
Englische Fassung EN 61804-5:2015**

Function blocks (FB) for process control and electronic device description
language (EDDL) –
Part 5: EDDL Builtin library
(IEC 61804-5:2015);
English version EN 61804-5:2015

Blocs fonctionnels (FB) pour les procédés industriels et le langage de description
électronique de produit (EDDL) –
Partie 5: Bibliothèque de Builtin EDDL
(IEC 61804-5:2015);
Version anglaise EN 61804-5:2015

Gesamtumfang 243 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2015-07-14 angenommene Europäische Norm als DIN-Norm ist 2016-04-01.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN EN 61804-5:2014-03.

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

Die enthaltene IEC-Publikation wurde vom SC 65E „Devices and integration in enterprise systems“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (stability date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnitsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

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

Der Zusammenhang der zitierten Dokumente mit den entsprechenden Deutschen Dokumenten 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 die vorliegende Norm erfüllt.

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61804-5 veröffentlicht. Für die meisten der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Einleitung

Dieser Teil der DIN EN 61804 Reihe ist eine Erweiterung der DIN EN 61804-3. In den vorherigen Ausgaben der DIN EN 61804-3 war die Bibliothek der vorgefertigten Unterprogramme Teil der elektronischen Gerätebeschreibungssprache (EDDL) nach DIN EN 61804-3. Aufgrund redaktioneller Problemen wird die EDDL Bibliothek der vorgefertigten Unterprogramme auf diesen separaten Teil der DIN EN 61804 verschoben.

– Leerseite –

English Version

Function blocks (FB) for process control and electronic device
description language (EDDL) - Part 5: EDDL Built-in library
(IEC 61804-5:2015)

Blocs fonctionnels (FB) pour les procédés industriels et le
langage de description électronique de produit (EDDL) -
Partie 5: Bibliothèque de Built-in EDDL
(IEC 61804-5:2015)

Funktionsbausteine für die Prozessautomation und
elektronische Gerätebeschreibungssprache - Teil 5:
Bibliothek vorgefertigter Unterprogramme
(IEC 61804-5:2015)

This European Standard was approved by CENELEC on 2015-07-14. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

European foreword

The text of document 65E/450/FDIS, future edition 1 of IEC 61804-5, prepared by SC 65E "Devices and integration in enterprise systems" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61804-5:2015.

The following dates are fixed:

- latest date by which the document has to be (dop) 2016-04-14
implemented at national level by
publication of an identical national
standard or by endorsement
- latest date by which the national (dow) 2018-07-14
standards conflicting with the
document have to be withdrawn

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 61804-5:2015 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION.....	20
1 Scope.....	21
2 Normative references	21
3 Terms, definitions, acronyms and abbreviated terms	21
3.1 Terms and definitions	21
3.2 Acronyms and abbreviated terms.....	21
4 EDDL Builtin library	22
4.1 General.....	22
4.2 Conventions for Builtin descriptions	22
4.3 Builtin categories	23
4.3.1 Overview	23
4.3.2 User interface Builtins	24
4.3.3 Communication Builtins	26
4.3.4 Action Builtins	28
4.4 Builtin _ERROR	28
4.5 Builtin _TRACE	29
4.6 Builtin _WARNING	29
4.7 Builtin abort	29
4.8 Builtin abort_on_all_comm_errors.....	30
4.9 Builtin ABORT_ON_ALL_COMM_STATUS.....	30
4.10 Builtin ABORT_ON_ALL_DEVICE_STATUS.....	31
4.11 Builtin ABORT_ON_ALL_RESPONSE_CODES	31
4.12 Builtin abort_on_all_response_codes.....	32
4.13 Builtin ABORT_ON_COMM_ERROR.....	32
4.14 Builtin abort_on_comm_error.....	33
4.15 Builtin ABORT_ON_COMM_STATUS	33
4.16 Builtin ABORT_ON_DEVICE_STATUS	34
4.17 Builtin ABORT_ON_NO_DEVICE.....	34
4.18 Builtin ABORT_ON_RESPONSE_CODE	35
4.19 Builtin abort_on_response_code.....	36
4.20 Builtin abortTransferPort	36
4.21 Builtin abs.....	37
4.22 Builtin ACKNOWLEDGE	37
4.23 Builtin acknowledge	37
4.24 Builtin acos	38
4.25 Builtin add_abort_method (version A)	38
4.26 Builtin add_abort_method (version B)	39
4.27 Builtin AddTime.....	39
4.28 Builtin asin	40
4.29 Builtin assign	40
4.30 Builtin assign_double	40
4.31 Builtin assign_float.....	41
4.32 Builtin assign_int.....	41
4.33 Builtin assign_var.....	42
4.34 Builtin assign2	42
4.35 Builtin atan.....	43

4.36	Builtin atof	43
4.37	Builtin atoi	44
4.38	Builtin browseIdentity	44
4.39	Builtin BUILD_MESSAGE	44
4.40	Builtin ByteToDouble	45
4.41	Builtin ByteToFloat	45
4.42	Builtin ByteToLong	46
4.43	Builtin ByteToShort	46
4.44	Builtin cbrt	47
4.45	Builtin ceil	47
4.46	Builtin closeTransferPort	47
4.47	Builtin cos	48
4.48	Builtin cosh	48
4.49	Builtin dassign	48
4.50	Builtin DATE_AND_TIME_VALUE_to_string	49
4.51	Builtin Date_to_DayOfMonth	49
4.52	Builtin DATE_to_days	50
4.53	Builtin Date_to_Month	50
4.54	Builtin DATE_to_string	50
4.55	Builtin Date_To_Time	51
4.56	Builtin Date_to_Year	51
4.57	Builtin days_to_DATE	51
4.58	Builtin DELAY	52
4.59	Builtin delay	52
4.60	Builtin DELAY_TIME	53
4.61	Builtin delayfor	53
4.62	Builtin delayfor2	54
4.63	Builtin DICT_ID	55
4.64	Builtin dictionary_string	55
4.65	Builtin DiffTime	55
4.66	Builtin discard_on_exit	56
4.67	Builtin DISPLAY	56
4.68	Builtin display	57
4.69	Builtin display_bitenum	57
4.70	Builtin display_builtin_error	58
4.71	Builtin display_comm_error	58
4.72	Builtin display_comm_status	59
4.73	Builtin display_device_status	59
4.74	Builtin display_dynamics	60
4.75	Builtin display_dynamics2	60
4.76	Builtin display_message	61
4.77	Builtin display_message2	62
4.78	Builtin display_response_code	62
4.79	Builtin display_response_status	63
4.80	Builtin display_xmtr_status	63
4.81	Builtin DoubleToByte	64
4.82	Builtin drand	64
4.83	Builtin dseed	65

4.84	Builtin edit_device_value	65
4.85	Builtin edit_device_value2	66
4.86	Builtin edit_local_value	67
4.87	Builtin edit_local_value2	68
4.88	Builtin exp	69
4.89	Builtin ext_send_command	69
4.90	Builtin ext_send_command_trans	70
4.91	Builtin fail_on_all_comm_errors	71
4.92	Builtin fail_on_all_response_codes	71
4.93	Builtin fail_on_comm_error	72
4.94	Builtin fail_on_response_code	72
4.95	Builtin fassign	73
4.96	Builtin fGetByte	73
4.97	Builtin fgetval	74
4.98	Builtin float_value	74
4.99	Builtin FloatToByte	74
4.100	Builtin floor	75
4.101	Builtin fmod	75
4.102	Builtin fpclassify	76
4.103	Builtin From_DATE_AND_TIME_VALUE	76
4.104	Builtin From_TIME_VALUE	76
4.105	Builtin fsetval	77
4.106	Builtin ftoa	77
4.107	Builtin fvar_value	78
4.108	Builtin get_acknowledgement	78
4.109	Builtin get_acknowledgement2	78
4.110	Builtin get_block_instance_by_object_index	79
4.111	Builtin get_block_instance_by_tag	79
4.112	Builtin get_block_instance_count	80
4.113	Builtin get_comm_error	80
4.114	Builtin get_comm_error_string	81
4.115	Builtin get_date	81
4.116	Builtin get_date_lelem	82
4.117	Builtin get_date_lelem2	82
4.118	Builtin get_date_value	83
4.119	Builtin get_date_value2	83
4.120	Builtin GET_DD_REVISION	84
4.121	Builtin get_dd_error	84
4.122	Builtin GET_DEV_VAR_VALUE	85
4.123	Builtin get_dev_var_value	85
4.124	Builtin GET_DEVICE_REVISION	86
4.125	Builtin GET_DEVICE_TYPE	86
4.126	Builtin get_dictionary_string	87
4.127	Builtin get_double	87
4.128	Builtin get_double_lelem	88
4.129	Builtin get_double_lelem2	88
4.130	Builtin get_double_value	89
4.131	Builtin get_double_value2	89

4.132	Builtin get_enum_string.....	90
4.133	Builtin get_float.....	90
4.134	Builtin get_float_lelem.....	91
4.135	Builtin get_float_lelem2.....	91
4.136	Builtin get_float_value.....	92
4.137	Builtin get_float_value2.....	92
4.138	Builtin GET_LOCAL_VAR_VALUE.....	93
4.139	Builtin get_local_var_value.....	93
4.140	Builtin GET_MANUFACTURER.....	94
4.141	Builtin get_more_status.....	94
4.142	Builtin get_resolve_status.....	95
4.143	Builtin get_response_code.....	95
4.144	Builtin get_response_code_string.....	96
4.145	Builtin get_rspcode_string.....	96
4.146	Builtin get_rspcode_string_by_id.....	97
4.147	Builtin get_signed.....	97
4.148	Builtin get_signed_lelem.....	98
4.149	Builtin get_signed_lelem2.....	98
4.150	Builtin get_signed_value.....	99
4.151	Builtin get_signed_value2.....	99
4.152	Builtin get_status_code_string.....	100
4.153	Builtin get_status_string.....	100
4.154	Builtin get_stddict_string.....	101
4.155	Builtin get_string.....	102
4.156	Builtin get_string_lelem.....	102
4.157	Builtin get_string_lelem2.....	103
4.158	Builtin get_string_value.....	103
4.159	Builtin get_string_value2.....	104
4.160	Builtin GET_TICK_COUNT.....	105
4.161	Builtin get_transfer_status.....	105
4.162	Builtin get_unsigned.....	105
4.163	Builtin get_unsigned_lelem.....	106
4.164	Builtin get_unsigned_lelem2.....	106
4.165	Builtin get_unsigned_value.....	107
4.166	Builtin get_unsigned_value2.....	108
4.167	Builtin get_variable_string.....	108
4.168	Builtin GetCurrentDate.....	109
4.169	Builtin GetCurrentDateAndTime.....	109
4.170	Builtin GetCurrentTime.....	109
4.171	Builtin iassign.....	110
4.172	Builtin igetval.....	110
4.173	Builtin IGNORE_ALL_COMM_STATUS.....	110
4.174	Builtin IGNORE_ALL_DEVICE_STATUS.....	111
4.175	Builtin IGNORE_ALL_RESPONSE_CODES.....	111
4.176	Builtin IGNORE_COMM_ERROR.....	112
4.177	Builtin IGNORE_COMM_STATUS.....	112
4.178	Builtin IGNORE_DEVICE_STATUS.....	113
4.179	Builtin IGNORE_NO_DEVICE.....	113

4.180	Builtin IGNORE_RESPONSE_CODE	114
4.181	Builtin int_value	114
4.182	Builtin is_NaN	115
4.183	Builtin isetval	115
4.184	Builtin isOffline.....	116
4.185	Builtin ITEM_ID	116
4.186	Builtin itoa (version A)	116
4.187	Builtin itoa (version B)	117
4.188	Builtin ivar_value.....	117
4.189	Builtin lassign	117
4.190	Builtin lgetval	118
4.191	Builtin ListDeleteElementAt	118
4.192	Builtin ListDeleteElementAt2	119
4.193	Builtin ListInsert	119
4.194	Builtin ListInsert2	120
4.195	Builtin log.....	120
4.196	Builtin LOG_MESSAGE.....	121
4.197	Builtin log10.....	121
4.198	Builtin log2.....	121
4.199	Builtin long_value.....	122
4.200	Builtin LongToByte	122
4.201	Builtin lsetval	123
4.202	Builtin lvar_value.....	123
4.203	Builtin Make_Time.....	123
4.204	Builtin MEMBER_ID	124
4.205	Builtin MenuDisplay.....	124
4.206	Builtin method_abort	125
4.207	Builtin nan.....	125
4.208	Builtin NaN_value	126
4.209	Builtin nanf.....	126
4.210	Builtin ObjectReference.....	127
4.211	Builtin openTransferPort.....	127
4.212	Builtin pop_abort_method.....	128
4.213	Builtin pow	128
4.214	Builtin process_abort.....	128
4.215	Builtin push_abort_method	129
4.216	Builtin put_date	129
4.217	Builtin put_date_value	130
4.218	Builtin put_date_value2	130
4.219	Builtin put_double	131
4.220	Builtin put_double_value	132
4.221	Builtin put_double_value2	132
4.222	Builtin put_float	133
4.223	Builtin put_float_value	133
4.224	Builtin put_float_value2	134
4.225	Builtin PUT_MESSAGE	134
4.226	Builtin put_message	135
4.227	Builtin put_signed.....	136

4.228	Builtin put_signed_value	136
4.229	Builtin put_signed_value2	137
4.230	Builtin put_string	137
4.231	Builtin put_string_value	138
4.232	Builtin put_string_value2	139
4.233	Builtin put_unsigned	140
4.234	Builtin put_unsigned_value	140
4.235	Builtin put_unsigned_value2	141
4.236	Builtin re_read_file	142
4.237	Builtin re_write_file	142
4.238	Builtin read_value	142
4.239	Builtin read_value2	143
4.240	Builtin ReadCommand	143
4.241	Builtin readItemFromDevice	144
4.242	Builtin remove_abort_method (version A)	144
4.243	Builtin remove_abort_method (version B)	145
4.244	Builtin remove_all_abort_methods	145
4.245	Builtin resolve_array_ref	145
4.246	Builtin resolve_array_ref2	146
4.247	Builtin resolve_block_ref	147
4.248	Builtin resolve_block_ref2	147
4.249	Builtin resolve_list_ref	148
4.250	Builtin resolve_local_ref	148
4.251	Builtin resolve_local_ref2	149
4.252	Builtin resolve_param_list_ref	149
4.253	Builtin resolve_param_ref	150
4.254	Builtin resolve_param_ref2	150
4.255	Builtin resolve_record_ref	151
4.256	Builtin resolve_record_ref2	152
4.257	Builtin ret_double_value	152
4.258	Builtin ret_double_value2	153
4.259	Builtin ret_float_value	153
4.260	Builtin ret_float_value2	154
4.261	Builtin ret_signed_value	154
4.262	Builtin ret_signed_value2	154
4.263	Builtin ret_unsigned_value	155
4.264	Builtin ret_unsigned_value2	155
4.265	Builtin retry_on_all_comm_errors	156
4.266	Builtin RETRY_ON_ALL_COMM_STATUS	156
4.267	Builtin RETRY_ON_ALL_DEVICE_STATUS	157
4.268	Builtin RETRY_ON_ALL_RESPONSE_CODES	157
4.269	Builtin retry_on_all_response_codes	158
4.270	Builtin RETRY_ON_COMM_ERROR	158
4.271	Builtin retry_on_comm_error	159
4.272	Builtin RETRY_ON_COMM_STATUS	159
4.273	Builtin RETRY_ON_DEVICE_STATUS	160
4.274	Builtin RETRY_ON_NO_DEVICE	160
4.275	Builtin RETRY_ON_RESPONSE_CODE	161

4.276	Builtin retry_on_response_code.....	161
4.277	Builtin round	162
4.278	Builtin save_on_exit	162
4.279	Builtin save_values	163
4.280	Builtin seconds_to_TIME_VALUE	163
4.281	Builtin seconds_to_TIME_VALUE8	163
4.282	Builtin SELECT_FROM_LIST.....	164
4.283	Builtin select_from_list	164
4.284	Builtin select_from_menu	165
4.285	Builtin select_from_menu2	166
4.286	Builtin send	167
4.287	Builtin send_all_values.....	168
4.288	Builtin send_command	168
4.289	Builtin send_command_trans.....	168
4.290	Builtin send_on_exit	169
4.291	Builtin send_trans	170
4.292	Builtin send_value.....	170
4.293	Builtin send_value2.....	171
4.294	Builtin SET_NUMBER_OF_RETRIES.....	171
4.295	Builtin sgetval	172
4.296	Builtin ShortToByte	172
4.297	Builtin sin.....	172
4.298	Builtin sinh.....	173
4.299	Builtin sqrt	173
4.300	Builtin ssetval	173
4.301	Builtin strcmp	174
4.302	Builtin strleft.....	174
4.303	Builtin strlen.....	175
4.304	Builtin strlwr.....	175
4.305	Builtin strmid	175
4.306	Builtin strright.....	176
4.307	Builtin strstr	176
4.308	Builtin strtrim.....	176
4.309	Builtin strupr	177
4.310	Builtin tan	177
4.311	Builtin tanh.....	177
4.312	Builtin Time_To_Date.....	178
4.313	Builtin TIME_VALUE_to_Hour	178
4.314	Builtin TIME_VALUE_to_Minute.....	178
4.315	Builtin TIME_VALUE_to_Second	179
4.316	Builtin TIME_VALUE_to_seconds	179
4.317	Builtin TIME_VALUE_to_string	180
4.318	Builtin timet_to_string.....	180
4.319	Builtin timet_to_TIME_VALUE	181
4.320	Builtin timet_to_TIME_VALUE8	181
4.321	Builtin To_Date	181
4.322	Builtin To_Date_and_Time	182
4.323	Builtin To_Time.....	182

4.324	Builtin To_TIME_VALUE	183
4.325	Builtin To_TIME_VALUE8	183
4.326	Builtin trunc	184
4.327	Builtin VARID	184
4.328	Builtin vassign	185
4.329	Builtin WriteCommand	185
4.330	Builtin writeItemToDevice	185
4.331	Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	186
4.332	Builtin XMTR_ABORT_ON_ALL_DATA	186
4.333	Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS	187
4.334	Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES	187
4.335	Builtin XMTR_ABORT_ON_COMM_ERROR	188
4.336	Builtin XMTR_ABORT_ON_COMM_STATUS	188
4.337	Builtin XMTR_ABORT_ON_DATA	189
4.338	Builtin XMTR_ABORT_ON_DEVICE_STATUS	189
4.339	Builtin XMTR_ABORT_ON_NO_DEVICE	190
4.340	Builtin XMTR_ABORT_ON_RESPONSE_CODE	190
4.341	Builtin XMTR_IGNORE_ALL_COMM_STATUS	191
4.342	Builtin XMTR_IGNORE_ALL_DATA	191
4.343	Builtin XMTR_IGNORE_ALL_DEVICE_STATUS	192
4.344	Builtin XMTR_IGNORE_ALL_RESPONSE_CODES	192
4.345	Builtin XMTR_IGNORE_COMM_ERROR	193
4.346	Builtin XMTR_IGNORE_COMM_STATUS	193
4.347	Builtin XMTR_IGNORE_DATA	194
4.348	Builtin XMTR_IGNORE_DEVICE_STATUS	194
4.349	Builtin XMTR_IGNORE_NO_DEVICE	195
4.350	Builtin XMTR_IGNORE_RESPONSE_CODE	195
4.351	Builtin XMTR_RETRY_ON_ALL_COMM_STATUS	196
4.352	Builtin XMTR_RETRY_ON_ALL_DATA	196
4.353	Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS	197
4.354	Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE (removed)	197
4.355	Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES	197
4.356	Builtin XMTR_RETRY_ON_COMM_ERROR	198
4.357	Builtin XMTR_RETRY_ON_COMM_STATUS	198
4.358	Builtin XMTR_RETRY_ON_DATA	199
4.359	Builtin XMTR_RETRY_ON_DEVICE_STATUS	199
4.360	Builtin XMTR_RETRY_ON_NO_DEVICE	200
4.361	Builtin XMTR_RETRY_ON_RESPONSE_CODE	200
5	Builtins return codes	201
Annex A	(normative) Profiles of Builtins	203
A.1	Conventions for profiles of Builtins	203
A.2	Profiles for PROFIBUS and PROFINET	203
A.3	Profiles for FOUNDATION™ Fieldbus	212
A.4	Profiles for HART®	220
A.5	Profiles for Communication Servers	230
Annex ZA	(normative) Normative references to international publications with their corresponding European publications	239

Table 1 – Format for the Builtins lexical element tables	22
Table 2 – Contents of the lexical element table.....	23
Table 3 – Usage of Builtins	24
Table 4 – User interface Builtins	25
Table 5 – Communication Builtins	26
Table 6 – Action Builtins	28
Table 7 – Builtin _ERROR.....	28
Table 8 – Builtin _TRACE	29
Table 9 – Builtin _WARNING.....	29
Table 10 – Builtin abort.....	30
Table 11 – Builtin abort_on_all_comm_errors	30
Table 12 – Builtin ABORT_ON_ALL_COMM_STATUS	31
Table 13 – Builtin ABORT_ON_ALL_DEVICE_STATUS	31
Table 14 – Builtin ABORT_ON_ALL_RESPONSE_CODES	32
Table 15 – Builtin abort_on_all_response_codes	32
Table 16 – Builtin ABORT_ON_COMM_ERROR	32
Table 17 – Builtin abort_on_comm_error	33
Table 18 – Builtin ABORT_ON_COMM_STATUS	34
Table 19 – Builtin ABORT_ON_DEVICE_STATUS	34
Table 20 – Builtin ABORT_ON_NO_DEVICE	35
Table 21 – Builtin ABORT_ON_RESPONSE_CODE.....	36
Table 22 – Builtin abort_on_response_code	36
Table 23 – Builtin abortTransferPort.....	36
Table 24 – Builtin abs	37
Table 25 – Builtin ACKNOWLEDGE	37
Table 26 – Builtin acknowledge	38
Table 27 – Builtin acos.....	38
Table 28 – Builtin add_abort_method	38
Table 29 – Builtin add_abort_method	39
Table 30 – Builtin AddTime	39
Table 31 – Builtin asin	40
Table 32 – Builtin assign.....	40
Table 33 – Builtin assign_double.....	41
Table 34 – Builtin assign_float	41
Table 35 – Builtin assign_int	41
Table 36 – Builtin assign_var	42
Table 37 – Builtin assign2.....	43
Table 38 – Builtin atan	43
Table 39 – Builtin atof	44
Table 40 – Builtin atoi	44
Table 41 – Builtin browselidentity.....	44
Table 42 – Builtin BUILD_MESSAGE	45

Table 43 – Builtin ByteToDouble	45
Table 44 – Builtin ByteToFloat	46
Table 45 – Builtin ByteToLong	46
Table 46 – Builtin ByteToShort.....	47
Table 47 – Builtin cbrt.....	47
Table 48 – Builtin ceil	47
Table 49 – Builtin closeTransferPort.....	48
Table 50 – Builtin cos	48
Table 51 – Builtin cosh	48
Table 52 – Builtin dassign.....	49
Table 53 – Builtin DATE_AND_TIME_VALUE_to_string	49
Table 54 – Builtin Date_to_DayOfMonth	49
Table 55 – Builtin DATE_to_days	50
Table 56 – Builtin Date_to_Month	50
Table 57 – Builtin DATE_to_string.....	51
Table 58 – Builtin Date_To_Time	51
Table 59 – Builtin Date_to_Year.....	51
Table 60 – Builtin days_to_DATE	52
Table 61 – Builtin DELAY.....	52
Table 62 – Builtin delay	53
Table 63 – Builtin DELAY_TIME.....	53
Table 64 – Builtin delayfor	54
Table 65 – Builtin delayfor2	54
Table 66 – Builtin DICT_ID	55
Table 67 – Builtin dictionary_string	55
Table 68 – Builtin DiffTime.....	56
Table 69 – Builtin discard_on_exit.....	56
Table 70 – Builtin DISPLAY	57
Table 71 – Builtin display.....	57
Table 72 – Builtin display_bitenum.....	58
Table 73 – Builtin display_builtin_error.....	58
Table 74 – Builtin display_comm_error.....	59
Table 75 – Builtin display_comm_status.....	59
Table 76 – Builtin display_device_status	59
Table 77 – Builtin display_dynamics.....	60
Table 78 – Builtin display_dynamics2.....	61
Table 79 – Builtin display_message	61
Table 80 – Builtin display_message2	62
Table 81 – Builtin display_response_code.....	63
Table 82 – Builtin display_response_status	63
Table 83 – Builtin display_xmtr_status	64
Table 84 – Builtin DoubleToByte	64

Table 85 – Builtin drand	65
Table 86 – Builtin dseed	65
Table 87 – Builtin edit_device_value	66
Table 88 – Builtin edit_device_value2	67
Table 89 – Builtin edit_local_value	68
Table 90 – Builtin edit_local_value2	69
Table 91 – Builtin exp	69
Table 92 – Builtin ext_send_command	70
Table 93 – Builtin ext_send_command_trans	70
Table 94 – Builtin fail_on_all_comm_errors	71
Table 95 – Builtin fail_on_all_response_codes	72
Table 96 – Builtin fail_on_comm_error	72
Table 97 – Builtin fail_on_response_code	73
Table 98 – Builtin fassign	73
Table 99 – Builtin fGetByte	73
Table 100 – Builtin fgetval	74
Table 101 – Builtin float_value	74
Table 102 – Builtin FloatToByte	75
Table 103 – Builtin floor	75
Table 104 – Builtin fmod	75
Table 105 – Builtin fpclassifyd	76
Table 106 – Builtin From_DATE_AND_TIME_VALUE	76
Table 107 – Builtin From_TIME_VALUE	77
Table 108 – Builtin fsetval	77
Table 109 – Builtin ftoa	77
Table 110 – Builtin fvar_value	78
Table 111 – Builtin get_acknowledgement	78
Table 112 – Builtin get_acknowledgement2	79
Table 113 – Builtin get_block_instance_by_object_index	79
Table 114 – Builtin get_block_instance_by_tag	80
Table 115 – Builtin get_block_instance_count	80
Table 116 – Builtin get_comm_error	81
Table 117 – Builtin get_comm_error_string	81
Table 118 – Builtin get_date	82
Table 119 – Builtin get_date_lelem	82
Table 120 – Builtin get_date_lelem2	83
Table 121 – Builtin get_date_value	83
Table 122 – Builtin get_date_value2	84
Table 123 – Builtin GET_DD_REVISION	84
Table 124 – Builtin get_dds_error	85
Table 125 – Builtin GET_DEV_VAR_VALUE	85
Table 126 – Builtin get_dev_var_value	86

Table 127 – Builtin GET_DEVICE_REVISION	86
Table 128 – Builtin GET_DEVICE_TYPE	86
Table 129 – Builtin get_dictionary_string	87
Table 130 – Builtin get_double	87
Table 131 – Builtin get_double_lelem	88
Table 132 – Builtin get_double_lelem2	88
Table 133 – Builtin get_double_value	89
Table 134 – Builtin get_double_value2	90
Table 135 – Builtin get_enum_string	90
Table 136 – Builtin get_float	91
Table 137 – Builtin get_float_lelem	91
Table 138 – Builtin get_float_lelem2	92
Table 139 – Builtin get_float_value	92
Table 140 – Builtin get_float_value2	93
Table 141 – Builtin GET_LOCAL_VAR_VALUE	93
Table 142 – Builtin get_local_var_value	94
Table 143 – Builtin GET_MANUFACTURER	94
Table 144 – Builtin get_more_status	95
Table 145 – Builtin get_resolve_status	95
Table 146 – Builtin get_response_code	96
Table 147 – Builtin get_response_code_string	96
Table 148 – Builtin get_rspcode_string	97
Table 149 – Builtin get_rspcode_string_by_id	97
Table 150 – Builtin get_signed	98
Table 151 – Builtin get_signed_lelem	98
Table 152 – Builtin get_signed_lelem2	99
Table 153 – Builtin get_signed_value	99
Table 154 – Builtin get_signed_value2	100
Table 155 – Builtin get_status_code_string	100
Table 156 – Builtin get_status_string	101
Table 157 – Builtin get_stddict_string	101
Table 158 – Builtin get_string	102
Table 159 – Builtin get_string_lelem	103
Table 160 – Builtin get_string_lelem2	103
Table 161 – Builtin get_string_value	104
Table 162 – Builtin get_string_value2	104
Table 163 – Builtin GET_TICK_COUNT	105
Table 164 – Builtin get_transfer_status	105
Table 165 – Builtin get_unsigned	106
Table 166 – Builtin get_unsigned_lelem	106
Table 167 – Builtin get_unsigned_lelem2	107
Table 168 – Builtin get_unsigned_value	107

Table 169 – Builtin get_unsigned_value2	108
Table 170 – Builtin get_variable_string	109
Table 171 – Builtin GetCurrentDate	109
Table 172 – Builtin GetCurrentDateAndTime	109
Table 173 – Builtin GetCurrentTime	110
Table 174 – Builtin iassign	110
Table 175 – Builtin igetval	110
Table 176 – Builtin IGNORE_ALL_COMM_STATUS	111
Table 177 – Builtin IGNORE_ALL_DEVICE_STATUS	111
Table 178 – Builtin IGNORE_ALL_RESPONSE_CODES	112
Table 179 – Builtin IGNORE_COMM_ERROR	112
Table 180 – Builtin IGNORE_COMM_STATUS	113
Table 181 – Builtin IGNORE_DEVICE_STATUS	113
Table 182 – Builtin IGNORE_NO_DEVICE	114
Table 183 – Builtin IGNORE_RESPONSE_CODE	114
Table 184 – Builtin int_value	115
Table 185 – Builtin is_NaN	115
Table 186 – Builtin isetval	115
Table 187 – Builtin isOffline	116
Table 188 – Builtin ITEM_ID	116
Table 189 – Builtin itoa (version A)	116
Table 190 – Builtin itoa (version B)	117
Table 191 – Builtin ivar_value	117
Table 192 – Builtin lassign	118
Table 193 – Builtin lgetval	118
Table 194 – Builtin ListDeleteElementAt	119
Table 195 – Builtin ListDeleteElementAt2	119
Table 196 – Builtin ListInsert	120
Table 197 – Builtin ListInsert2	120
Table 198 – Builtin log	121
Table 199 – Builtin LOG_MESSAGE	121
Table 200 – Builtin log10	121
Table 201 – Builtin log2	122
Table 202 – Builtin long_value	122
Table 203 – Builtin LongToByte	122
Table 204 – Builtin lsetval	123
Table 205 – Builtin lvar_value	123
Table 206 – Builtin Make_Time	124
Table 207 – Builtin MEMBER_ID	124
Table 208 – Builtin MenuDisplay	125
Table 209 – Builtin method_abort	125
Table 210 – Builtin nan	126

Table 211 – Builtin NaN_value	126
Table 212 – Builtin nanf	127
Table 213 – Builtin ObjectReference	127
Table 214 – Builtin openTransferPort	128
Table 215 – Builtin pop_abort_method	128
Table 216 – Builtin pow	128
Table 217 – Builtin process_abort	129
Table 218 – Builtin push_abort_method	129
Table 219 – Builtin put_date	129
Table 220 – Builtin put_date_value	130
Table 221 – Builtin put_date_value2	131
Table 222 – Builtin put_double	131
Table 223 – Builtin put_double_value	132
Table 224 – Builtin put_double_value2	133
Table 225 – Builtin put_float	133
Table 226 – Builtin put_float_value	134
Table 227 – Builtin put_float_value2	134
Table 228 – Format options	135
Table 229 – Builtin PUT_MESSAGE	135
Table 230 – Builtin put_message	136
Table 231 – Builtin put_signed	136
Table 232 – Builtin put_signed_value	137
Table 233 – Builtin put_signed_value2	137
Table 234 – Builtin put_string	138
Table 235 – Builtin put_string_value	139
Table 236 – Builtin put_string_value2	139
Table 237 – Builtin put_unsigned	140
Table 238 – Builtin put_unsigned_value	141
Table 239 – Builtin put_unsigned_value2	141
Table 240 – Builtin re_read_file	142
Table 241 – Builtin re_write_file	142
Table 242 – Builtin read_value	143
Table 243 – Builtin read_value2	143
Table 244 – Builtin ReadCommand	144
Table 245 – Builtin readItemFromDevice	144
Table 246 – Builtin remove_abort_method	144
Table 247 – Builtin remove_abort_method	145
Table 248 – Builtin remove_all_abort_methods	145
Table 249 – Builtin resolve_array_ref	146
Table 250 – Builtin resolve_array_ref2	146
Table 251 – Builtin resolve_block_ref	147
Table 252 – Builtin resolve_block_ref2	148

Table 253 – Builtin resolve_list_ref.....	148
Table 254 – Builtin resolve_local_ref.....	149
Table 255 – Builtin resolve_local_ref2.....	149
Table 256 – Builtin resolve_param_list_ref	150
Table 257 – Builtin resolve_param_ref	150
Table 258 – Builtin resolve_param_ref2.....	151
Table 259 – Builtin resolve_record_ref	151
Table 260 – Builtin resolve_record_ref2	152
Table 261 – Builtin ret_double_value.....	153
Table 262 – Builtin ret_double_value2.....	153
Table 263 – Builtin ret_float_value	153
Table 264 – Builtin ret_float_value2	154
Table 265 – Builtin ret_signed_value.....	154
Table 266 – Builtin ret_signed_value2.....	155
Table 267 – Builtin ret_unsigned_value	155
Table 268 – Builtin ret_unsigned_value2	156
Table 269 – Builtin retry_on_all_comm_errors	156
Table 270 – Builtin RETRY_ON_ALL_COMM_STATUS	157
Table 271 – Builtin RETRY_ON_ALL_DEVICE_STATUS	157
Table 272 – Builtin RETRY_ON_ALL_RESPONSE_CODES.....	158
Table 273 – Builtin retry_on_all_response_codes	158
Table 274 – Builtin RETRY_ON_COMM_ERROR	159
Table 275 – Builtin retry_on_comm_error	159
Table 276 – Builtin RETRY_ON_COMM_STATUS	160
Table 277 – Builtin RETRY_ON_DEVICE_STATUS	160
Table 278 – Builtin RETRY_ON_NO_DEVICE	161
Table 279 – Builtin RETRY_ON_RESPONSE_CODE.....	161
Table 280 – Builtin retry_on_response_code	162
Table 281 – Builtin round	162
Table 282 – Builtin save_on_exit.....	163
Table 283 – Builtin save_values	163
Table 284 – Builtin seconds_to_TIME_VALUE.....	163
Table 285 – Builtin seconds_to_TIME_VALUE.....	164
Table 286 – Builtin SELECT_FROM_LIST	164
Table 287 – Builtin select_from_list.....	165
Table 288 – Builtin select_from_menu.....	166
Table 289 – Builtin select_from_menu2	167
Table 290 – Builtin send	167
Table 291 – Builtin send_all_values	168
Table 292 – Builtin send_command.....	168
Table 293 – Builtin send_command_trans	169
Table 294 – Builtin send_on_exit.....	169

Table 295 – Builtin send_trans	170
Table 296 – Builtin send_value	171
Table 297 – Builtin send_value2	171
Table 298 – Builtin SET_NUMBER_OF_RETRIES	172
Table 299 – Builtin sgetval	172
Table 300 – Builtin ShortToByte	172
Table 301 – Builtin sin	173
Table 302 – Builtin sinh	173
Table 303 – Builtin sqrt	173
Table 304 – Builtin ssetval	174
Table 305 – Builtin strcmp	174
Table 306 – Builtin strleft	174
Table 307 – Builtin strlen	175
Table 308 – Builtin strlwr	175
Table 309 – Builtin strmid	176
Table 310 – Builtin strright	176
Table 311 – Builtin strstr	176
Table 312 – Builtin strtrim	177
Table 313 – Builtin strupr	177
Table 314 – Builtin tan	177
Table 315 – Builtin tanh	178
Table 316 – Builtin Time_To_Date	178
Table 317 – Builtin TIME_VALUE_to_Hour	178
Table 318 – Builtin TIME_VALUE_to_Minute	179
Table 319 – Builtin TIME_VALUE_to_Second	179
Table 320 – Builtin TIME_VALUE_to_seconds	180
Table 321 – Builtin TIME_VALUE_to_string	180
Table 322 – Builtin timet_to_string	181
Table 323 – Builtin timet_to_TIME_VALUE	181
Table 324 – Builtin timet_to_TIME_VALUE8	181
Table 325 – Builtin To_Date	182
Table 326 – Builtin To_Date_and_Time	182
Table 327 – Builtin To_Time	183
Table 328 – Builtin To_TIME_VALUE	183
Table 329 – Builtin To_TIME_VALUE8	184
Table 330 – Builtin trunc	184
Table 331 – Builtin VARID	184
Table 332 – Builtin vassign	185
Table 333 – Builtin WriteCommand	185
Table 334 – Builtin writeltemToDevice	186
Table 335 – Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	186
Table 336 – Builtin XMTR_ABORT_ON_ALL_DATA	187

Table 337 – Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS	187
Table 338 – Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES	188
Table 339 – Builtin XMTR_ABORT_ON_COMM_ERROR	188
Table 340 – Builtin XMTR_ABORT_ON_COMM_STATUS	189
Table 341 – Builtin XMTR_ABORT_ON_DATA	189
Table 342 – Builtin XMTR_ABORT_ON_DEVICE_STATUS	190
Table 343 – Builtin XMTR_ABORT_ON_NO_DEVICE	190
Table 344 – Builtin XMTR_ABORT_ON_RESPONSE_CODE	191
Table 345 – Builtin XMTR_IGNORE_ALL_COMM_STATUS	191
Table 346 – Builtin XMTR_IGNORE_ALL_DATA	192
Table 347 – Builtin XMTR_IGNORE_ALL_DEVICE_STATUS	192
Table 348 – Builtin XMTR_IGNORE_ALL_RESPONSE_CODES	193
Table 349 – Builtin XMTR_IGNORE_COMM_ERROR	193
Table 350 – Builtin XMTR_IGNORE_COMM_STATUS	194
Table 351 – Builtin XMTR_IGNORE_DATA	194
Table 352 – Builtin XMTR_IGNORE_DEVICE_STATUS	195
Table 353 – Builtin XMTR_IGNORE_NO_DEVICE	195
Table 354 – Builtin XMTR_IGNORE_RESPONSE_CODE	196
Table 355 – Builtin XMTR_RETRY_ON_ALL_COMM_STATUS	196
Table 356 – Builtin XMTR_RETRY_ON_ALL_DATA	197
Table 357 – Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS	197
Table 358 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES	198
Table 359 – Builtin XMTR_RETRY_ON_COMM_ERROR	198
Table 360 – Builtin XMTR_RETRY_ON_COMM_STATUS	199
Table 361 – Builtin XMTR_RETRY_ON_DATA	199
Table 362 – Builtin XMTR_RETRY_ON_DEVICE_STATUS	200
Table 363 – Builtin XMTR_RETRY_ON_NO_DEVICE	200
Table 364 – Builtin XMTR_RETRY_ON_RESPONSE_CODE	201
Table 365 – Contents of the return codes description table	201
Table 366 – Return code descriptions	202
Table 367 – Return code descriptions	202
Table A.1 – Builtin profile for PROFIBUS and PROFINET	203
Table A.2 – Builtin profile for FOUNDATION fieldbus	212
Table A.3 – Builtin profile for HART	220
Table A.4 – Builtin profile for Communication Servers	230

INTRODUCTION

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

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL AND ELECTRONIC DEVICE DESCRIPTION LANGUAGE (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 documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61804-3:2015, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 3: EDDL syntax and semantics*

IEEE 754, *Standard for Binary Floating-Point Arithmetic for microprocessor systems*

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 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

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.

PI PROFIBUS & PROFINET ² International
PNO PROFIBUS Nutzerorganisation

4 EDDL Builtin library

4.1 General

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. 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.
- EDDL basic constructs and attributes referenced in this document are written in upper case letters.

Table 1 – Format for the Builtins lexical element tables

Parameter Name	Type	Direction	Usage	Description

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.

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 is implicitly passed by reference so that the called Builtin function can modify the value. The return value of a Builtin is passed by value, if any.
	I/O	An input output parameter is implicitly passed by reference so that the called Builtin function can modify the value.
	—	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).

4.3 Builtin categories

4.3.1 Overview

There are the following special categories of Builtins:

- User interface Builtins
- Communication Builtins
- Action Builtins

Restrictions apply to the usage of Builtins, depending on their category (see Table 3).

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.			

4.3.2 User interface Builtins

User interface Builtins display information and in addition some of them require interaction from the user (see Table 4).

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

4.3.3 Communication Builtins

The communication Builtins allow methods to communicate with the device or to control the communication with the device (see Table 5).

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
closeTransferPort
ext_send_command
ext_send_command_trans
get_transfer_status
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
openTransferPort
read_value
read_value2
ReadCommand
readItemFromDevice
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

Builtin name
RETRY_ON_RESPONSE_CODE
retry_on_response_code
send
send_all_values
send_command
send_command_trans
send_on_exit
send_trans
send_value
send_value2
WriteCommand
writelnItemToDevice
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_DATA
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

4.3.4 Action Builtins

The action Builtins allow for accessing the value of the variable from which the action was called (see Table 6). These Builtins may not be used in other methods or in actions called from a MENU.

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

4.4 Builtin _ERROR

Purpose

The Builtin _ERROR puts a textual information in a log. This information is put into the category error. The Builtin _TRACE and the Builtin _WARNING 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

`_ERROR(text)`

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

Table 7 – Builtin _ERROR

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

4.5 Builtin _TRACE

Purpose

The Builtin _TRACE puts a textual information in a log. This information is put into the category trace. The Builtin _ERROR and the Builtin _WARNING 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

_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 element of the Builtin abort is 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.

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	—	—	—

4.10 Builtin ABORT_ON_ALL_DEVICE_STATUS

Purpose

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

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 the send_command function for implementation of the masks. See 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_DEVICE_STATUS (VOID)

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

Table 13 – Builtin ABORT_ON_ALL_DEVICE_STATUS

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

4.11 Builtin ABORT_ON_ALL_RESPONSE_CODES

Purpose

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

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

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 the send command function for implementation of the masks. See 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_RESPONSE_CODES (VOID)

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

Table 14 – Builtin ABORT_ON_ALL_RESPONSE_CODES

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

4.12 Builtin abort_on_all_response_codes

Purpose

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

Lexical structure

abort_on_all_response_codes(VOID)

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

Table 15 – Builtin abort_on_all_response_codes

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

4.13 Builtin ABORT_ON_COMM_ERROR

Purpose

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

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 the send_command function for implementation of the masks. 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_COMM_ERROR(VOID)

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

Table 16 – Builtin ABORT_ON_COMM_ERROR

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

4.14 Builtin abort_on_comm_error

Purpose

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

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

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

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

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

Lexical structure

abort_on_comm_error(error)

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

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 366.

4.15 Builtin ABORT_ON_COMM_STATUS

Purpose

The Builtin ABORT_ON_COMM_STATUS will set the correct bit(s) in the comm status abort mask in such a way that the specified comm_status_value will cause the method to abort. 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 the send command function for implementation of the masks. See 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_COMM_STATUS(comm_status)

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

Table 18 – Builtin ABORT_ON_COMM_STATUS

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

4.16 Builtin ABORT_ON_DEVICE_STATUS

Purpose

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

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 the send command function for implementation of the masks. See 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_DEVICE_STATUS(device_status)

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

Table 19 – Builtin ABORT_ON_DEVICE_STATUS

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

4.17 Builtin ABORT_ON_NO_DEVICE

Purpose

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

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 the send command function for implementation of the masks. See 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_NO_DEVICE(VOID)

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

Table 20 – Builtin ABORT_ON_NO_DEVICE

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

4.18 Builtin ABORT_ON_RESPONSE_CODE

Purpose

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

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.

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

- response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- response_code_retry_mask = NO BITS SET
- cmd_48_response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- cmd_48_response_code_retry_mask = NO BITS SET
- device_status_abort_mask = DEVICE_MALFUNCTION
- device_status_retry_mask = NO BITS SET
- cmd_48_device_status_abort_mask = DEVICE_MALFUNCTION
- cmd_48_device_status_retry_mask = NO BITS SET
- comm_status_abort_mask = NO BITS SET
- comm_status_retry_mask = ALL BITS SET
- cmd_48_comm_status_abort_mask = NO BITS SET
- cmd_48_comm_status_retry_mask = ALL BITS SET
- no_device_abort_mask = ALL BITS SET
- comm_error_abort_mask = ALL BITS SET
- comm_error_retry_mask = NO BITS SET
- cmd_48_no_device_abort_mask = ALL BITS SET
- cmd_48_comm_error_abort_mask = ALL BITS SET
- cmd_48_comm_error_retry_mask = NO BITS SET
- cmd_48_data_abort_mask = NO BITS SET
- cmd_48_data_retry_mask = NO BITS SET

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_RESPONSE_CODE(response_code)

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

Table 21 – Builtin ABORT_ON_RESPONSE_CODE

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

4.19 Builtin abort_on_response_code

Purpose

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

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

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

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

Lexical structure

abort_on_response_code (code)

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

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 366.

4.20 Builtin abortTransferPort

Purpose

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

Lexical structure

abortTransferPort (handle)

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

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 367.

4.21 Builtin abs

Purpose

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

Lexical structure

`abs (x)`

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

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.

4.22 Builtin ACKNOWLEDGE

Purpose

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

Lexical structure

`ACKNOWLEDGE (prompt)`

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

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 367.

4.23 Builtin acknowledge

Purpose

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

Lexical structure

`acknowledge(prompt, global_var_ids)`

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

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 367.

4.24 Builtin acos

Purpose

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

Lexical structure

acos(x)

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

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.

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 367.

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. Builtins 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
<code>method_id</code>	unsigned long	I	m	The ID of the method.
<return>	long	O	m	One of the return codes specified in Table 366.

4.27 Builtin AddTime

Purpose

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

Lexical structure

```
AddTime(t0, seconds)
```

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

Table 30 – Builtin AddTime

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

4.28 Builtin asin

Purpose

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

Lexical structure

`asin(x)`

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

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.

4.29 Builtin assign

Purpose

The Builtin assign sets the value of a VARIABLE to another VARIABLE's value. It sets the value of the destination VARIABLE equal to the value of the source VARIABLE.

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

The parameters `dst_id`, `src_id`, `dst_member_id`, and `src_member_id` are specified using the Builtins ITEM_ID and MEMBER ID and the reference of the VARIABLE. Both the source VARIABLES and the destination VARIABLE shall have the same type.

Lexical structure

`assign(dst_id, dst_member_id, src_id, src_member_id)`

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

Table 32 – Builtin assign

Parameter name	Type	Direction	Usage	Description
<code>dst_id</code>	unsigned long	I	m	The item ID of the destination variable.
<code>dst_member_id</code>	unsigned long	I	m	The member ID of the destination variable.
<code>src_id</code>	unsigned long	I	m	The item ID of the source variable.
<code>src_member_id</code>	unsigned long	I	m	The member ID of the source variable.
<return>	long	O	m	One of the return codes specified in Table 366.

4.30 Builtin assign_double

Purpose

The Builtin assign_double assigns a value to a VARIABLE.

Lexical structure

```
assign_double(device_var, new_value)
```

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

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 367.

4.31 Builtin assign_float**Purpose**

The Builtin assign_float assigns a value to VARIABLE.

Lexical structure

```
assign_float(device_var, new_value)
```

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

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 367.

4.32 Builtin assign_int**Purpose**

The Builtin assign_int assigns a value to VARIABLE.

Lexical structure

```
assign_int(device_var, new_value)
```

The lexical elements of the Builtin assign_int are shown in Table 35.

Table 35 – Builtin assign_int

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	The name of the variable.
new_value	int	I	m	The new value of the referenced variable.
<return>	int	O	m	One of the return codes specified in Table 367.

4.33 Builtin assign_var

Purpose

The Builtin assign_var assigns a value of the source VARIABLE to the destination VARIABLE. Both variables shall have the same type.

Lexical structure

```
assign_var(destination_var, source_var)
```

The lexical elements of the Builtin assign_var are shown in Table 36.

Table 36 – Builtin assign_var

Parameter name	Type	Direction	Usage	Description
destination_var	reference	I	m	The name of the destination variable.
source_var	reference	I	m	The name of the source variable.
<return>	int	O	m	One of the return codes specified in Table 367.

4.34 Builtin assign2

Purpose

The Builtin assign2 sets the value of a VARIABLE to another VARIABLE's value. It sets the value of the destination VARIABLE equal to the value of the source VARIABLE.

The parameters dst_id, src_id, dst_member_id, and src_member_id are specified using the Builtins ITEM_ID and MEMBER ID and the reference of the VARIABLE. Both the source VARIABLES and the destination VARIABLE shall have the same type.

Lexical structure

```
assign2(dst_blk_id, dst_blk_num, dst_id, dst_member_id, src_blk_id, src_blk_num,  
src_id, src_member_id)
```

The lexical elements of the Builtin assign2 are shown in Table 37.

Table 37 – Builtin assign2

Parameter name	Type	Direction	Usage	Description
dst_blk_id	unsigned long	I	m	The item ID of the destination block.
dst_blk_num	unsigned long	I	m	The occurrence number of the destination block.
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_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 366.

4.35 Builtin atan

Purpose

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

Lexical structure

atan(x)

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

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.

4.36 Builtin atof

Purpose

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

Lexical structure

atof(str)

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

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.

4.37 Builtin atoi

Purpose

The Builtin converts a text into an integer value.

Lexical structure

`atoi(str)`

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

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.

4.38 Builtin browseIdentity

Purpose

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

Lexical structure

`browseIdentity(blob_item)`

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

Table 41 – Builtin browseIdentity

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 367.

4.39 Builtin BUILD_MESSAGE

Purpose

The Builtin BUILD_MESSAGE returns a string based on a format specification. 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

`BUILD_MESSAGE(string)`

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

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.

4.40 Builtin ByteToDouble

Purpose

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

Lexical structure

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

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

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.

4.41 Builtin ByteToFloat

Purpose

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

Lexical structure

ByteToFloat(in1, in2, in3, in4)

The lexical elements of the Builtin ByteToFloat are shown in Table 44.

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.

4.42 Builtin ByteToLong

Purpose

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

Lexical structure

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

The lexical elements of the Builtin ByteToLong are shown in Table 45.

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.

4.43 Builtin ByteToShort

Purpose

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

Lexical structure

`ByteToShort(in1, in2)`

The lexical elements of the Builtin ByteToShort are shown in Table 46.