

SENSORS & SYSTEMS

By far the better solution



INSTRUMENTATION
MICRO-EPSILON

Sensors

- Opto series
(ILD, ODC, IFD, ILR)
- Eddy series
- Capa series

Interfaces

- RS232
- IF2004 (RS422)
- IF2008 (RS422)
- USB
- TCP/IP

Data Acquisition Library

Micro-Epsilon

Data Acquisition Library
X9751165

Programming language interface for accessing digital sensors
2.1.3.7684

MICRO-EPSILON
MESSTECHNIK
GmbH & Co. KG
Königbacher Strasse 15

D-94496 Ortenburg

Tel. +49/85 42/ 168-0
Fax +49/85 42/ 168-90
e-mail info@micro-epsilon.de
www.micro-epsilon.de

Contents

1	Introduction	19
2	Installation	20
3	Accessing MEDAQLib in Visual C/C++	21
3.1	Setting up Visual Studio	21
3.1.1	Using MEDAQLib.lib, static approach	21
3.1.2	Using MEDAQLib.dll, dynamic approach	21
4	Using MEDAQLib	22
4.1	Block based acquisition	22
4.2	Polling	22
5	Samples	23
5.1	C#Example	23
5.2	DelphiExample	23
5.3	DLLExample	23
5.4	LabView	23
5.5	LibExample	23
5.6	UnicodeExample	23
5.7	VBAExample	24
5.8	VB6.0Example	24
5.9	VB2008Example	24
6	Function Reference	25
6.1	Opening the Driver	25
6.2	Releasing Driver	25
6.3	Set parameters	26
6.4	Retrieving sensor answer	31
6.5	Clear internal parameter buffer	36
6.6	Connecting to the sensor	36
6.7	Closing connection to sensor	37
6.8	Sending commands to the sensor	37
6.9	Polling data from sensor	38
6.10	Number of values available to read	39
6.11	Block wise data acquisition from sensor	40
6.12	Get additional error information	41
6.13	Get version of MEDAQLib dll	42
A	Parameters	44
A.1	Interface parameters	44
A.1.1	All Interfaces	44
A.1.2	RS232	48
A.1.3	IF2004	49
A.1.4	IF2008	50
A.1.5	TCP/IP	52
A.1.6	DriverX_USB	52
A.1.7	USBIO	53
A.2	Communication via SensorCommand	54
A.3	Sensor commands valid for each sensor	58
A.3.1	Use_Defaults	58
A.3.2	SettingsChanged	60

A.3.3	Get_DrvSetting	60
A.3.4	Cmd_Generic	62
A.3.5	Clear_Buffers	62
A.3.6	DataAvail_Event	63
B	Commands for ILR110x_115x	64
B.1	Get_Parameters (GAP)	64
B.2	Get_Energy (GDB)	67
B.3	Get_SerialNbr (GNR)	68
B.4	Get_ErrorStatus (GSI)	68
B.5	Get_Temperature (GTE)	68
B.6	Get_Version (GVE)	68
B.7	Set_Offset (IDO)	69
B.8	Set_VisibleLaser (IVL)	69
B.9	Set_HysteresisQ1 (IH1)	69
B.10	Set_HysteresisQ2 (IH2)	69
B.11	Set_LimitQ1-1 (IL1)	70
B.12	Set_LimitQ2-1 (IL2)	70
B.13	Set_LimitQA-1 (IL1)	70
B.14	Set_LimitQ1-2 (IL4)	70
B.15	Set_LimitQ2-2 (IL2)	71
B.16	Set_LimitQA-2 (IL6)	71
B.17	Set_ModeQ1 (IM1)	71
B.18	Set_ModeQ2 (IM2)	71
B.19	Set_NormQA (INA)	72
B.20	Set_NormQ1 (IN1)	72
B.21	Set_NormQ2 (IN2)	72
B.22	Set_ContinuousMode (ICM)	72
B.23	Exec_ContMeasure (ECM)	72
B.24	Trg_SingleMeasure (ESM)	73
B.25	Set_Stand-by (ISB)	73
B.26	Save_Parameters (EPW)	73
C	Commands for ILR118x	74
C.1	DistanceTracking (DT)	74
C.2	DistanceTracking7m (DS)	74
C.3	DistanceTracking10Hz (DW)	74
C.4	DistanceTracking50Hz (DX)	74
C.5	DistanceTriggered (DF)	74
C.6	DistanceMeasure (DM)	74
C.7	StopTracking (<ESC>)	74
C.8	Set_OutputFormat (SD)	75
C.9	Get_OutputFormat (SD)	75
C.10	Set_ScaleFactor (SF)	75
C.11	Get_ScaleFactor (SF)	75
C.12	Set_Offset (OF)	76
C.13	Get_Offset (OF)	76
C.14	CurrentDistAsOffset (SO)	76
C.15	Set_RemovalMeasVal (RM)	76
C.16	Get_RemovalMeasVal (RM)	77
C.17	Set_ErrorMode (SE)	77
C.18	Get_ErrorMode (SE)	78
C.19	Set_AlarmStart (AC)	78

C.20 Get_AlarmStart (AC)	78
C.21 Set_AlarmHysteresis (AH)	78
C.22 Get_AlarmHysteresis (AH)	79
C.23 Set_AlarmWidth (AW)	79
C.24 Get_AlarmWidth (AW)	79
C.25 Set_RangeBegin4mA (RB)	79
C.26 Get_RangeBegin4mA (RB)	80
C.27 Set_RangeEnd20mA (RE)	80
C.28 Get_RangeEnd20mA (RE)	80
C.29 Set_AverageValue (SA)	80
C.30 Get_AverageValue (SA)	81
C.31 Set_MeasureTime (ST)	81
C.32 Get_MeasureTime (ST)	81
C.33 Set_TriggerDelay (TD)	81
C.34 Get_TriggerDelay (TD)	82
C.35 Set_TriggerMode (TM)	82
C.36 Get_TriggerMode (TM)	82
C.37 Set_Baudrate (BR)	83
C.38 Get_Baudrate (BR)	83
C.39 Set_Autostart (AS)	83
C.40 Get_Autostart (AS)	84
C.41 Get_Info (ID)	84
C.42 Get_AllParameters (PA)	84
C.43 Get_Temperature (TP)	87
C.44 Reset_Parameters (PR)	87
D Commands for ILR1191	91
D.1 DistanceTracking (DT)	91
D.2 DistanceTriggered (DF)	91
D.3 DistanceMeasure (DM)	91
D.4 SpeedMeasure (VM)	91
D.5 SpeedTracking (VT)	91
D.6 StopTracking (<ESC>)	91
D.7 Set_OutputFormat (SD)	92
D.8 Get_OutputFormat (SD)	92
D.9 Set_TerminatingChar (TE)	92
D.10 Get_TerminatingChar (TE)	93
D.11 Set_FormatSSI (SC)	93
D.12 Get_FormatSSI (SC)	93
D.13 Set_ScaleFactor (SF)	94
D.14 Get_ScaleFactor (SF)	94
D.15 Set_Offset (OF)	94
D.16 Get_Offset (OF)	94
D.17 CurrentDistAsOffset (SO)	95
D.18 Set_ErrorMode (SE)	95
D.19 Get_ErrorMode (SE)	95
D.20 Set_MeasureWindow (MW)	95
D.21 Get_MeasureWindow (MW)	96
D.22 Set_AnalogOutLimits (QA)	96
D.23 Get_AnalogOutLimits (QA)	96
D.24 Set_Out1Parameters (Q1)	97
D.25 Get_Out1Parameters (Q1)	97
D.26 Set_Out2Parameters (Q2)	98

D.27 Get_Out2Parameters (Q2)	99
D.28 Set_AverageValue (SA)	99
D.29 Get_AverageValue (SA)	99
D.30 Set_TriggerDelay (TD)	100
D.31 Get_TriggerDelay (TD)	100
D.32 Set_Baudrate (BR)	100
D.33 Get_Baudrate (BR)	101
D.34 Set_Autostart (AS)	101
D.35 Get_Autostart (AS)	102
D.36 Set_MeasFreq (MF)	103
D.37 Get_MeasFreq (MF)	103
D.38 Set_PilotLaser (PL)	103
D.39 Get_PilotLaser (PL)	103
D.40 Get_Info (ID)	104
D.41 Get_AllParameters (PA)	104
D.42 Get_Temperature (TP)	108
D.43 Get_HWDiagnosis (HW)	108
D.44 Reset_Parameters (PR)	108
D.45 Trigger_ColdStart (DR)	112
D.46 Use_Defaults	113
D.47 Get_DrvSetting	114
E Commands for ILD1302	116
E.1 Get_Info	116
E.2 Get_Settings	119
E.3 Set_KeyLock	121
E.4 Dat_Out_Off	121
E.5 Dat_Out_On	121
E.6 Set_Av	122
E.7 Set_Baudrate	122
E.8 Set_ErrorHandler	122
E.9 Laser_Off	123
E.10 Laser_On	123
E.11 ASCII_Output	123
E.12 Set_OutputType	123
E.13 Reset_Boot	123
E.14 Set_Default	123
E.15 Set_OutputMode	123
E.16 Set_OutputTime	124
E.17 Set_SaveSettingsMode	124
E.18 Set_ExtInputMode	124
E.19 Set_TeachValue	124
E.20 Reset_TeachValue	125
E.21 Set_PeakSearching	125
E.22 Set_Threshold	125
E.23 Get_Video	125
E.24 Use_Defaults	125
E.25 Get_DrvSetting	126

F	Commands for ILD1401	127
F.1	Reset_Boot	127
F.2	Get_Version	127
F.3	Get_Info	127
F.4	Set_OutputChannel	128
F.5	SaveLastMV	128
F.6	Set_Median	129
F.7	Use_Defaults	129
F.8	Get_DrvSetting	129
G	Commands for ILD1401-Controller	130
G.1	Dat_Out_Off	130
G.2	Dat_Out_On	130
G.3	Set_ControllerReset	130
G.4	Get_FirmwareVersion	130
G.5	Get_Settings	131
G.6	Get_BatteryCharge	131
G.7	Get_Version	132
G.8	Get_ValueMedian	132
G.9	Get_ValueActual	132
G.10	Get_SensorRange	132
G.11	Reset_Boot	133
G.12	Set_Zero	133
G.13	Set_Unit	133
G.14	Set_Direction	133
G.15	Set_OutputChannel	133
H	Commands for ILD1402	134
H.1	Get_Info	134
H.2	Get_Settings	137
H.3	Set_KeyLock	140
H.4	Dat_Out_Off	140
H.5	Dat_Out_On	140
H.6	Set_Av	140
H.7	Set_Baudrate	141
H.8	Set_ErrorHandler	141
H.9	Set_Speed	141
H.10	Laser_Off	141
H.11	Laser_On	142
H.12	ASCII_Output	142
H.13	Set_OutputType	142
H.14	Reset_Boot	142
H.15	Set_Default	142
H.16	Set_CIMode_1401	142
H.17	Set_OutputMode	143
H.18	Set_OutputTime	143
H.19	Set_SaveSettingsMode	143
H.20	Set_ExtInputMode	143
H.21	Set_TeachValue	144
H.22	Reset_TeachValue	144
H.23	Set_PeakSearching	144
H.24	Set_Threshold	144
H.25	Get_Video	145

H.26 Use_Defaults	145
H.27 Get_DrvSetting	146
I Commands for ILD1700	147
I.1 Get_MeasValue	147
I.2 Get_Info	147
I.3 Get_Settings	150
I.4 Set_KeyLock	154
I.5 WriteFlashZero	154
I.6 Set_Av0	154
I.7 Set_Av1	154
I.8 Set_Av2	154
I.9 Set_Av3	154
I.10 Set_AvX	154
I.11 Dat_Out_Off	155
I.12 Dat_Out_On	155
I.13 Set_Av_T	155
I.14 Set_Limits	155
I.15 Set_Baudrate	156
I.16 Set_ErrorHandler	156
I.17 Set_Sync_TrgMode	156
I.18 Set_UpperLimit_F1	157
I.19 Set_LowerLimit_F1	157
I.20 Set_Speed	157
I.21 Laser_Off	157
I.22 Laser_On	157
I.23 ASCII_Output	157
I.24 Set_OutputType	157
I.25 Set_ErrorOutput	158
I.26 Reset_Boot	158
I.27 Set_Default	158
I.28 Set_VideoMode	158
I.29 Get_Video	158
I.30 Use_Defaults	159
I.31 Get_DrvSetting	160
J Commands for ILD1800	161
J.1 Get_Info	161
J.2 Get_Version	162
J.3 Zero	162
J.4 Set_Av0	162
J.5 Set_Av1	162
J.6 Set_Av2	162
J.7 Set_Av3	163
J.8 Set_AvX	163
J.9 Dat_Out_Off	163
J.10 Dat_Out_On	163
J.11 Displacement	163
J.12 Thickness	163
J.13 Multilayer	163
J.14 Refraction	163
J.15 Set_Av_T	164
J.16 Reset_Boot	164

K	Commands for ILD2000	165
K.1	Set_AvX	165
K.2	Save_EEPROM	165
L	Commands for ILD2200	166
L.1	Get_Info	166
L.2	Get_Settings	167
L.3	Get_Version	169
L.4	Set_KeyLock	169
L.5	Zero	169
L.6	Set_Av0	169
L.7	Set_Av1	169
L.8	Set_Av2	169
L.9	Set_Av3	169
L.10	Set_AvX	170
L.11	Dat_Out_Off	170
L.12	Dat_Out_On	170
L.13	Set_Av_T	170
L.14	Laser_Off	170
L.15	Laser_On	170
L.16	Transmit_Intensity	170
L.17	Reset_Boot	171
M	Commands for IFD2400	172
M.1	Set_ActiveSensor (SEN)	172
M.2	Get_ActiveSensor (SEN?)	172
M.3	Get_Range (SCA)	172
M.4	Acquire_DarkSig (DRK)	173
M.5	FastDark (FDK)	173
M.6	Set_SRIIndex (SRA)	173
M.7	Get_SRIIndex (SRA?)	173
M.8	Set_SumFreq (SHZ)	174
M.9	Get_SumFreq (SHZ?)	174
M.10	Set_DutyCycle (DCY)	174
M.11	Get_DutyCycle (DCY?)	174
M.12	Set_MeasureMode (MOD)	175
M.13	Get_MeasureMode (MOD?)	175
M.14	Set_RefractIndex (SRI)	175
M.15	Get_RefractIndex (SRI?)	175
M.16	Set_Threshold (THR)	176
M.17	Get_Threshold (THR?)	176
M.18	Set_KeyLock (LOC)	176
M.19	Get_KeyLock (LOC?)	176
M.20	Set_Averaging (AVR)	177
M.21	Get_Averaging (AVR?)	177
M.22	Set_SpectralAv (AVS)	177
M.23	Get_SpectralAv (AVS?)	177
M.24	Continue (CTN)	177
M.25	Start_Trigger (TRG)	178
M.26	End_Trigger	178
M.27	SingleShot_Trg (TRE)	178
M.28	Get_Status (STS)	178
M.29	Get_Version (VER)	180

M.30 Set_OutputData (SOD)	180
M.31 Get_OutputData (SOD?)	180
M.32 Set_Ascii (ASC)	181
M.33 Set_Binary (BIN)	181
M.34 Set_AnalogOut (ANA)	181
M.35 Get_AnalogOut (ANA?)	182
M.36 Save_Setup (SSU)	183
M.37 Get_CCD (CCD)	183
M.38 End_CCD	183
M.39 Get_Memory (DEB)	183
M.40 Upload_CalibTable	184
M.41 Use_Defaults	184
M.42 Get_DrvSetting	185
N Commands for IFD2401	187
N.1 Set_ActiveSensor (SEN)	187
N.2 Get_ActiveSensor (SEN?)	187
N.3 Get_Range (SCA)	187
N.4 Get_AllRanges (LUL)	188
N.5 Acquire_DarkSig (DRK)	188
N.6 FastDark (FDK)	188
N.7 Set_AutoDark (ADK)	188
N.8 Get_AutoDark (ADK?)	189
N.9 Set_SRIndex (SRA)	189
N.10 Get_SRIndex (SRA?)	189
N.11 Set_FreeSR (FRQ)	189
N.12 Get_FreeSR (FRQ?)	190
N.13 Set_Exposure (TEX)	190
N.14 Get_Exposure (TEX?)	190
N.15 Get_MinSR (FRM)	191
N.16 Set_DoubleFreq (DFA)	191
N.17 Get_DoubleFreq (DFA?)	191
N.18 Set_Frequencies (DFF)	191
N.19 Get_Frequencies (DFF?)	192
N.20 Set_IntensityMode (DFI)	192
N.21 Get_IntensityMode (DFI?)	192
N.22 Set_MeasureMode (MOD)	193
N.23 Get_MeasureMode (MOD?)	193
N.24 Set_RefractIndex (SRI)	193
N.25 Get_RefractIndex (SRI?)	193
N.26 Set_Threshold (MNP)	194
N.27 Get_Threshold (MNP?)	194
N.28 Set_HoldLastValid (HLV)	194
N.29 Get_HoldLastValid (HLV?)	194
N.30 Set_Threshold1 (SPP)	195
N.31 Get_Threshold1 (SPP?)	195
N.32 Set_Threshold2 (SDP)	195
N.33 Get_Threshold2 (SDP?)	195
N.34 Set_FirstPeakMode (MSP)	196
N.35 Get_FirstPeakMode (MSP?)	196
N.36 Set_DataScale (CEE)	196
N.37 Get_DataScale (CEE?)	196
N.38 Set_BarycenterSca (CEB)	197

N.39 Get_BarycenterSca (CEB?)	197
N.40 Set_BarycenterOff (CRB)	197
N.41 Get_BarycenterOff (CRB?)	197
N.42 Set_LampTest (SLP)	198
N.43 Get_LampTest (SLP?)	198
N.44 Set_LampTestThr (CSL)	198
N.45 Get_LampTestThr (CSL?)	198
N.46 Set_LEDIntensity (LED)	199
N.47 Get_LEDIntensity (LED?)	199
N.48 Set_AutoAdaptLED (AAL)	199
N.49 Get_AutoAdaptLED (AAL?)	199
N.50 Set_AdaptLEDThr (VTH)	200
N.51 Get_AdaptLEDThr (VTH?)	200
N.52 Set_IntLightSrc (CCL)	200
N.53 Get_IntLightSrc (CCL?)	200
N.54 Set_RefldxFFile (INF)	201
N.55 Get_RefldxFFile (INF?)	201
N.56 Upload_RefldxFFile	202
N.57 RecenterEncoder (RCD)	202
N.58 Set_MissingSignal (RSP)	203
N.59 Get_MissingSignal (RSP?)	203
N.60 Set_Reverse (RVS)	203
N.61 Get_Reverse (RVS?)	204
N.62 Set_Averaging (AVR)	204
N.63 Get_Averaging (AVR?)	204
N.64 Set_SpectralAv (AVS)	204
N.65 Get_SpectralAv (AVS?)	205
N.66 Continue (CTN)	205
N.67 Start_Trigger (TRG)	205
N.68 End_Trigger	205
N.69 SingleShot_Trg (TRE)	205
N.70 Set_TrgMode_Edge (TRS)	205
N.71 Set_TrgMode_State (TRN)	206
N.72 Set_ActiveEdge (TRF)	206
N.73 Get_ActiveEdge (TRF?)	206
N.74 Software_Trigger (STR)	206
N.75 Set_Watchdog (WDE)	206
N.76 Get_Watchdog (WDE?)	207
N.77 Set_WatchdogPrd (WDP)	207
N.78 Get_WatchdogPrd (WDP?)	207
N.79 Get_Status (STS)	207
N.80 Get_Version (VER)	209
N.81 Set_OutputData (SOD)	209
N.82 Get_OutputData (SOD?)	210
N.83 Set_Ascii (ASC)	210
N.84 Set_Binary (BIN)	210
N.85 Set_AnalogOut (ANA)	210
N.86 Get_AnalogOut (ANA?)	211
N.87 Set_AnalogZero (SOF)	212
N.88 Set_Baudrate (BAU)	212
N.89 Get_Baudrate (BAU?)	213
N.90 Save_Setup (SSU)	213
N.91 Reset (RST)	213

N.92 Get_CCD (CCD)	213
N.93 Get_DarkSig (SGD)	213
N.94 Get_WhiteRef (SGW)	214
N.95 Get_NormSig (SGN)	214
N.96 Get_CalibTable (SGC)	214
N.97 Start_Spectrum	214
N.98 Get_Spectrum	215
N.99 End_Spectrum	216
N.100 Upload_CalibTable	216
N.101 Use_Defaults	217
N.102 Get_DrvSetting	218
O Commands for IFD2430	220
O.1 Set_ActiveSensor (SEN)	220
O.2 Get_ActiveSensor (SEN?)	220
O.3 Get_Range (SCA)	220
O.4 Acquire_DarkSig (DRK)	221
O.5 FastDark (FDK)	221
O.6 Set_SRIndex (SRA)	221
O.7 Get_SRIndex (SRA?)	222
O.8 Set_FreeSR (FRQ)	222
O.9 Get_FreeSR (FRQ?)	223
O.10 Set_Exposure (TEX)	223
O.11 Get_Exposure (TEX?)	223
O.12 Set_MeasureMode (MOD)	223
O.13 Get_MeasureMode (MOD?)	224
O.14 Set_RefractIndex (SRI)	224
O.15 Get_RefractIndex (SRI?)	224
O.16 Set_Threshold (MNP)	224
O.17 Get_Threshold (MNP?)	225
O.18 Set_KeyLock (LOC)	225
O.19 Get_KeyLock (LOC?)	225
O.20 Set_Averaging (AVR)	225
O.21 Get_Averaging (AVR?)	226
O.22 Continue (CTN)	226
O.23 Start_Trigger (TRG)	226
O.24 End_Trigger	226
O.25 Get_Status (STS)	226
O.26 Get_Version (VER)	227
O.27 Set_OutputData (SOD)	228
O.28 Get_OutputData (SOD?)	228
O.29 Set_Ascii (ASC)	228
O.30 Set_Binary (BIN)	229
O.31 Set_IPAddress (IPA)	229
O.32 Get_IPAddress (IPA?)	229
O.33 Set_Baudrate (BAU)	229
O.34 Get_Baudrate (BAU?)	229
O.35 Save_Setup (SSU)	230
O.36 Get_CCD (CCD)	230
O.37 Get_DarkSig (SGD)	230
O.38 Get_WhiteRef (SGW)	230
O.39 Get_NormSig (SGN)	231
O.40 Get_CalibTable (SGC)	231

O.41 Upload_CalibTable	231
O.42 Use_Defaults	232
O.43 Get_DrvSetting	233
P Commands for IFD2431	234
P.1 Set_ActiveSensor (SEN)	234
P.2 Get_ActiveSensor (SEN?)	234
P.3 Get_Range (SCA)	234
P.4 Get_AllRanges (LUL)	235
P.5 Acquire_DarkSig (DRK)	235
P.6 FastDark (FDK)	235
P.7 Set_AutoDark (ADK)	235
P.8 Get_AutoDark (ADK?)	236
P.9 Set_SRIndex (SRA)	236
P.10 Get_SRIndex (SRA?)	236
P.11 Set_FreeSR (FRQ)	237
P.12 Get_FreeSR (FRQ?)	237
P.13 Set_Exposure (TEX)	237
P.14 Get_Exposure (TEX?)	238
P.15 Get_MinSR (FRM)	238
P.16 Set_MeasureMode (MOD)	238
P.17 Get_MeasureMode (MOD?)	238
P.18 Set_RefractIndex (SRI)	239
P.19 Get_RefractIndex (SRI?)	239
P.20 Set_Threshold (MNP)	239
P.21 Get_Threshold (MNP?)	239
P.22 Set_HoldLastValid (HLV)	240
P.23 Get_HoldLastValid (HLV?)	240
P.24 Set_Threshold1 (SPP)	240
P.25 Get_Threshold1 (SPP?)	240
P.26 Set_Threshold2 (SDP)	241
P.27 Get_Threshold2 (SDP?)	241
P.28 Set_FirstPeakMode (MSP)	241
P.29 Get_FirstPeakMode (MSP?)	241
P.30 Set_DataScale (CEE)	242
P.31 Get_DataScale (CEE?)	242
P.32 Set_BarycenterSca (CEB)	242
P.33 Get_BarycenterSca (CEB?)	242
P.34 Set_BarycenterOff (CRB)	243
P.35 Get_BarycenterOff (CRB?)	243
P.36 Set_RefIdxFile (INF)	243
P.37 Get_RefIdxFile (INF?)	244
P.38 Upload_RefIdxFile	245
P.39 RecenterEncoder (RCD)	245
P.40 Set_MissingSignal (RSP)	245
P.41 Get_MissingSignal (RSP?)	246
P.42 Set_Reverse (RVS)	246
P.43 Get_Reverse (RVS?)	246
P.44 Set_Binning (FBH)	246
P.45 Get_Binning (FBH?)	247
P.46 Set_Averaging (AVR)	247
P.47 Get_Averaging (AVR?)	247
P.48 Set_SpectralAv (AVS)	247

P.49 Get_SpectralAv (AVS?)	248
P.50 Continue (CTN)	248
P.51 Start_Trigger (TRG)	248
P.52 End_Trigger	248
P.53 SingleShot_Trg (TRE)	248
P.54 Set_TrgMode_Edge (TRS)	248
P.55 Set_TrgMode_State (TRN)	249
P.56 Set_ActiveEdge (TRF)	249
P.57 Get_ActiveEdge (TRF?)	249
P.58 Software_Trigger (STR)	249
P.59 Set_Watchdog (WDE)	249
P.60 Get_Watchdog (WDE?)	250
P.61 Set_WatchdogPrd (WDP)	250
P.62 Get_WatchdogPrd (WDP?)	250
P.63 Get_Status (STS)	250
P.64 Get_Version (VER)	252
P.65 Set_OutputData (SOD)	252
P.66 Get_OutputData (SOD?)	253
P.67 Set_Ascii (ASC)	253
P.68 Set_Binary (BIN)	253
P.69 Set_AnalogOut (ANA)	253
P.70 Get_AnalogOut (ANA?)	254
P.71 Set_AnalogZero (SOF)	255
P.72 Set_Baudrate (BAU)	255
P.73 Get_Baudrate (BAU?)	256
P.74 Save_Setup (SSU)	256
P.75 Reset (RST)	256
P.76 Get_CCD (CCD)	256
P.77 Get_DarkSig (SGD)	256
P.78 Get_WhiteRef (SGW)	257
P.79 Get_NormSig (SGN)	257
P.80 Get_CalibTable (SGC)	257
P.81 Start_Spectrum	257
P.82 Get_Spectrum	258
P.83 End_Spectrum	259
P.84 Upload_CalibTable	259
P.85 Use_Defaults	260
P.86 Get_DrvSetting	261
Q Commands for ODC2500	263
Q.1 Reset_Boot	263
Q.2 Get_Info	263
Q.3 Dat_Out_Off	264
Q.4 Dat_Out_On	264
Q.5 Choose_MeasProg	264
Q.6 Switch_Edge	264
Q.7 Read_OptionData	265
Q.8 Read_MeasProgData	268
Q.9 Write_OptionData	271
Q.10 Write_MeasProgData	273
Q.11 Save_OptionData	276
Q.12 Save_MeasProgData	276
Q.13 Read_MinMax	276

Q.14 Read_MinMaxReset	277
Q.15 Use_Defaults	278
Q.16 Get_DrvSetting	278
R Commands for ODC2600	279
R.1 Reset_Boot	279
R.2 Get_Info	279
R.3 Dat_Out_Off	280
R.4 Dat_Out_On	280
R.5 Choose_MeasProg	280
R.6 Switch_Edge	280
R.7 Read_OptionData	282
R.8 Read_MeasProgData	285
R.9 Write_OptionData	289
R.10 Write_MeasProgData	292
R.11 Save_OptionData	296
R.12 Save_MeasProgData	296
R.13 Triggermode_Reset	296
R.14 Triggermode_Trigger	296
R.15 Set_LightRef	296
R.16 Reset_LightRef	296
R.17 Read_MinMax	297
R.18 Read_MinMaxReset	297
R.19 Use_Defaults	298
R.20 Get_DrvSetting	298
S Commands for ESC4912	299
S.1 Set_SRIndex (SRA)	299
S.2 Get_SRIndex (SRA?)	299
S.3 Set_Trigger (TRG)	300
S.4 Get_Trigger (TRG?)	300
S.5 Get_Measure (GMD)	300
S.6 Set_AvrType (AVT)	300
S.7 Get_AvrType (AVT?)	300
S.8 Set_AvrNbr (AVN)	301
S.9 Get_AvrNbr (AVN?)	301
S.10 ChannelStatus (CHS)	301
S.11 Set_Channel (CHT)	301
S.12 Get_Channel (CHT?)	302
S.13 Set_LinMode (LIN)	302
S.14 Get_LinMode (LIN?)	302
S.15 Set_LinPoint (SLP)	303
S.16 Get_LinPoint (GLP)	303
S.17 Get_Status (STS)	304
S.18 Get_Version (VER)	305
S.19 Save_Setup (SSU)	305
S.20 Read_Setup (RSU)	305
S.21 Factory_Defaults (FDE)	305
S.22 Use_Defaults	307
S.23 Get_DrvSetting	308

T	Commands for DT6100	309
T.1	Set_SRIndex (SRA)	309
T.2	Get_SRIndex (SRA?)	309
T.3	Set_Trigger (TRG)	310
T.4	Get_Trigger (TRG?)	310
T.5	Get_Measure (GMD)	310
T.6	Set_Coefficient (SCO)	310
T.7	Get_Coefficient (GCO)	311
T.8	Set_AvrType (AVT)	311
T.9	Get_AvrType (AVT?)	311
T.10	Set_AvrNbr (AVN)	311
T.11	Get_AvrNbr (AVN?)	312
T.12	Set_LinMode (LIN)	312
T.13	Get_LinMode (LIN?)	312
T.14	Set_LinPoint (SLP)	312
T.15	Get_LinPoint (GLP)	313
T.16	Get_Status (STS)	313
T.17	Get_Version (VER)	314
T.18	Save_Setup (SSU)	314
T.19	Read_Setup (RSU)	314
T.20	Factory_Defaults (FDE)	315
T.21	Use_Defaults	316
T.22	Get_DrvSetting	316
U	Commands for KSS6380	317
U.1	Set_SRIndex (SRA)	317
U.2	Get_SRIndex (SRA?)	317
U.3	Set_Trigger (TRG)	317
U.4	Get_Trigger (TRG?)	318
U.5	Get_Measure (GMD)	318
U.6	Set_TempCoeff (STC)	318
U.7	Get_TempCoeff (GTC)	318
U.8	Get_UnlinEddyVal (GUE)	319
U.9	Set_AvrType (AVT)	319
U.10	Get_AvrType (AVT?)	319
U.11	Set_AvrNbr (AVN)	319
U.12	Get_AvrNbr (AVN?)	320
U.13	ChannelStatus (CHS)	320
U.14	Set_Channel (CHT)	320
U.15	Get_Channel (CHT?)	320
U.16	Set_LinPoint (SLP)	321
U.17	Get_LinPoint (GLP)	321
U.18	Get_Status (STS)	322
U.19	Get_Version (VER)	323
U.20	Save_Setup (SSU)	323
U.21	Read_Setup (RSU)	323
U.22	Factory_Defaults (FDE)	324
U.23	Use_Defaults	325
U.24	Get_DrvSetting	325

V	Commands for DT6500	327
V.1	Set_SRIndex (SRA)	327
V.2	Get_SRIndex (SRA?)	327
V.3	Set_Trigger (TRG)	328
V.4	Get_Trigger (TRG?)	328
V.5	Get_Measure (GMD)	328
V.6	Set_AvrType (AVT)	328
V.7	Get_AvrType (AVT?)	329
V.8	Set_AvrNbr (AVN)	329
V.9	Get_AvrNbr (AVN?)	329
V.10	ChannelStatus (CHS)	329
V.11	Set_Channel (CHT)	330
V.12	Get_Channel (CHT?)	330
V.13	Set_Display (DIS)	330
V.14	Get_Display (DIS?)	330
V.15	Set_MathFunction (SMF)	331
V.16	Get_MathFunction (GMF)	331
V.17	Clr_MathFunction (CMF)	332
V.18	Set_LinMode (LIN)	332
V.19	Get_LinMode (LIN?)	333
V.20	Set_LinPoint (SLP)	333
V.21	Get_LinPoint (GLP)	333
V.22	Get_Status (STS)	334
V.23	Get_Version (VER)	336
V.24	Save_Setup (SSU)	336
V.25	Read_Setup (RSU)	336
V.26	Factory_Defaults (FDE)	336
V.27	Use_Defaults	338
V.28	Get_DrvSetting	339
W	Commands for EncoderIF2004	340
W.1	Use_Defaults	340
W.2	Enc_ClearCounter	341
W.3	Enc_LoadCounter	341
W.4	Enc_LatchCounter	341
W.5	Enc_UnlockTraceC	342
W.6	Enc_IsFirstRef	342
W.7	Enc_IsSecondRef	342
W.8	Enc_GetLatchReg	342
W.9	Enc_SetLoadReg	342
W.10	IF2004_SystemReset	343
W.11	IF2004_CheckGate	343
W.12	Enc_CheckRef	343
W.13	IF2004_SensorReset12	343
W.14	IF2004_SensorReset34	343
W.15	Get_FPGAVersion	344
X	Commands for PCICard_IF2008	345
X.1	Use_Defaults	345
X.2	Get_DrvSetting	346
X.3	Use_Gate	347
X.4	Set_TimerFrequency	347
X.5	Set_EncoderInterpolation	348

X.6	Get_EncoderInterpolation	349
X.7	Set_EncoderDirection	349
X.8	Get_EncoderDirection	350
X.9	Set_EncoderMode	350
X.10	Get_EncoderMode	350
X.11	Set_EncoderLatchSource	351
X.12	Get_EncoderLatchSource	351
X.13	Set_EncoderPreload	352
X.14	Set_DigitalOutSource	352
X.15	Get_DigitalOutSource	353
X.16	Set_TxDSource	353
X.17	Get_TxDSource	353
X.18	Set_TriggerSource	354
X.19	Get_TriggerSource	354
X.20	Set_DigitalInLatchSource	354
X.21	Get_DigitalInLatchSource	355
X.22	Set_ADCLatchSource	355
X.23	Get_ADCLatchSource	356
X.24	Get_FPGAVersion	356
X.25	Is_Channel56Available	356
X.26	Is_ADCAvailable	357
X.27	Is_DigitalIOAvailable	357
X.28	Clear_Encoder	357
X.29	Load_Encoder	357
X.30	Latch_Encoder	358
X.31	EnableRef_Encoder	358
X.32	Get_EncoderValue	358
X.33	Get_EncoderReference	359
X.34	Get_ADCValue	359
X.35	Get_DigitalInValue	360
X.36	Get_RxDValue	360
X.37	Set_DigitalOutValue	360
X.38	Get_DigitalOutValue	360
X.39	Set_TxDValue	361
X.40	Get_TxDValue	361
X.41	Set_TriggerValue	361
X.42	Get_TriggerValue	361

1 Introduction

MEDAQLib is a software library for easy data acquisition and communication with digital Micro-Epsilon sensors. The library is independent from any communication protocol or hardware interface, i.e. all sensors or controllers are accessed from your program in the same way independent whether via TCP/IP or USB or serial communication.

Currently the following sensors are supported:

ILR sensors: optoNCDT110x, optoNCDT115x, optoNCDT118x, optoNCDT1191.

ILD sensors: optoNCDT1302, optoNCDT1401, optoNCDT1401-Controller, optoNCDT1402, optoNCDT1700, optoNCDT1800, optoNCDT2000, optoNCDT2200 (including optoNCDT2220).

IFD sensors: optoNCDT2400, optoNCDT2401, optoNCDT2430, optoNCDT2431.

ODC sensors: optoCONTROL2500, optoCONTROL2600.

Eddy sensors: eddyNCDT4912.

Capa sensors: capaNCDT6100, capaNCDT6500.

PCI cards: Encoder of IF2004 and IF2008 PCI card. Digital OUT/IN, Analog IN of IF2008 PCI card.

The sensors can be accessed through interfaces RS232, USB (Converter from RS422), USB (via DriverX or USBIO), IF2004 and IF2008 PCI card (RS422) and TCP/IP.

The software consists of a DLL to be imported into your data acquisition project. As programming languages C/C++, Visual Basic, Delphi and many other languages are supported. For C/C++ an additional include file is provided to get you started.

More than one sensor can be used over different interfaces at the same time (from one or several applications).

The functions of MEDAQLib are thread-safe, so you can call any function at the same time from concurrent threads.

Additionally Micro-Epsilon provides ICONNECT having special modules to access all the features of these sensors. ICONNECT is a programming environment giving you the possibility to develop applications without the need to "real programming".

2 Installation

MEDAQLib comes as one zip file. On extracting the zip file to some directory on your hard disk the following directory structure and files will be created:

Release\MEDAQLib.dll	the driver dll.
Release\MEDAQLib.lib	linker library for Visual C/C++ projects.
MEDAQLib.h	include file for C/C++ projects.
ErrCodes.h	list of all error codes.
Documentation\MEDAQLib.pdf	the reference guide you are currently reading.
Documentation\Version History.txt	changes between the different released versions.
Samples	examples showing usage of MEDAQLib.

Nothing has to be installed into some Windows directory. Nothing has to be changed in your Windows registry.

In addition the following drivers are included in the zip file:

Driver\DriverXUSB	for accessing the IFD2401/IFD2431 over USB.
Driver\IF2004	for accessing the IF2004 PCI card.
Driver\IF2008	for accessing the IF2008 PCI card.
Driver\USBIO	for accessing the ILD-1401-Controller over USB
Driver\FTDI_USB-RS422_Converter_V2.0	the driver for the RS422 to USB converter from FTDI, version 2.0

These drivers are also included in the driver CD supplied with your hardware.

3 Accessing MEDAQLib in Visual C/C++

3.1 Setting up Visual Studio

For accessing MEDAQLib from Visual Studio copy the file MEDAQLib.h into your project directory, add it to your project (Project->Add Files) and include it in your C/C++ source file as follows:

```
#include "MEDAQLib.h"
```

Now you are able to compile your project accessing MEDAQLib. For linking in MEDAQLib.dll there are two possibilities:

3.1.1 Using MEDAQLib.lib, static approach

Copy the file MEDAQLib.lib into your project directory and add it to your project (Project->Add Files). Now you are able to also link your project.

3.1.2 Using MEDAQLib.dll, dynamic approach

Do not link your project against MEDAQLib.lib. For accessing the functions you have to load the MEDAQLib.dll using

```
HINSTANCE hInstance = LoadLibrary("MEDAQLib.dll");
```

and get the functions pointers like

```
CREATESENSORINSTANCE pCreateSensorInstance= (CREATESENSORINSTANCE)  
GetProcAddress (hInstance, "CreateSensorInstance");
```

```
RELEASESENSORINSTANCE pReleaseSensorInstance= (RELEASESENSORIN-  
STANCE) GetProcAddress (hInstance, "ReleaseSensorInstance");
```

Please check any return codes after calling these functions, which is not shown here. Calling a function then is done like

```
DWORD result = pCreateSensorInstance(SENSOR_ILD2000);
```

4 Using MEDAQLib

MEDAQLib continuously collects all data from sensor in an own thread or timer and stores it into the internal ring buffer. No function call from user to MEDAQLib does interrupt reading the data, so no data is lost. For reading the data from MEDAQLib to host application, MEDAQLib supports two operation modes:

4.1 Block based acquisition

The measurement values are collected and processed in large data blocks. The advantage of this method is that the transfer cycles do not have to be so fast and often. Because of this normally there is no loss of data.

4.2 Polling

Using this method only the very last measurement value is returned to the caller. So the response time is very short, but you may miss some data on not polling often enough.

Both modes may be used simultaneously because polling for data does not change any internal buffer of MEDAQLib.

The function `DataAvail` can be used to retrieve the number of available values in buffer.

To wait for enough data, the user can call `DataAvail` in a loop (combined with a `Sleep` function) or he can use the `DataAvail` event ([A.3.6](#)).

5 Samples

For real examples please take a look at the given examples in subdirectory samples. They show you different possibilities on using MEDAQLib:

5.1 C#Example

This example shows how to access an ILD-1401 via TCP/IP from C#. Before using it please change the IP address to suit your settings.

5.2 DelphiExample

This example shows how to access an ILD-1401 via RS232 from Delphi. Before using it please change the COM port number to suit your settings.

5.3 DLLExample

This example shows how to access an ILD-2200 via IF2004 hardware card from Visual C++. In addition it shows how to access the functions from MEDAQLib without using MEDAQLib.lib, but using the dynamic approach using LoadLibrary and GetProcAddress.

5.4 LabView

This example shows how to access an ILD-1700 via RS232 from LabView 8.6 (and exported to 8.5 and 8.2). Before using it please change the COM port number to suit your settings. Several VI's with base functionality are included which can be used for own applications.

5.5 LibExample

This example shows how to access an ILD-1700 via RS232 from Visual C++. Before using it please change the COM port number to suit your settings. In contrast to the DLLExample is links against the MEDAQLib.lib library directly.

5.6 UnicodeExample

This example shows how to access an IFC2401 via USB (DriverX_USB) from Visual C++. It uses the MEDAQLib dynamically and can be compiled for ANSI and UNICODE. To wait for data it uses event handling.

5.7 VBAExample

This example shows how to access an ODC-2600 via RS232 from Excel speaking Visual Basic for Applications. Before using it please change the COM port number to suit your settings. To see the code please right click the "Table 1" tab at the bottom of Excel. In the context menu select "Show code". The second example (VBAExample2.xls) does the same but with sensor ILD1700.

5.8 VB6.0Example

This example shows how to access an IFC-2430 via TCP/IP using Visual Basic 6.0. Before using it please change the IP address to suit your settings.

5.9 VB2008Example

This example shows how to access an ILD1700 via IF2004 using Visual Basic 2008.

6 Function Reference

6.1 Opening the Driver

Name: CreateSensorInstance

CreateSensorInstance

Description:

Opens the driver and creates an instance for the sensor.

Attention! Please don't forget to release the instance at the end. Otherwise there will be memory and handle leaks.

Declaration:

```
DWORD CreateSensorInstance (ME_SENSOR sensor);
```

Parameter: ME_SENSOR sensor

sensor

Direction: [IN]

Valid values:

- SENSOR_ILR110x_115x (19)
- SENSOR_ILR118x (20)
- SENSOR_ILR1191 (21)
- SENSOR_ILD1302 (24)
- SENSOR_ILD1401 (1)
- CONTROLLER_ILD1401 (14)
- SENSOR_ILD1402 (23)
- SENSOR_ILD1700 (2)
- SENSOR_ILD1800 (3)
- SENSOR_ILD2000 (4)
- SENSOR_ILD2200 (5)
- SENSOR_IFD2400 (6)
- SENSOR_IFD2401 (12)
- SENSOR_IFD2430 (7)
- SENSOR_IFD2431 (13)
- SENSOR_ODC2500 (8)
- SENSOR_ODC2600 (9)
- CONTROLLER_ESC4912 (17)
- SENSOR_ASP5500 (11)
- SENSOR_DT6100 (16)
- CONTROLLER_DT6500 (15)
- CONTROLLER_KSS6380 (18)
- ENCODER_IF2004 (10)
- PCI_CARD_IF2008 (22)

Description: Type of sensor used.

Returns: Number of the created sensor instance or zero, if parameter sensor is not valid.

6.2 Releasing Driver

Name: ReleaseSensorInstance

ReleaseSensorInstance

Description:

Free the specified sensor instance.

Declaration:

```
ERR_CODE ReleaseSensorInstance (DWORD instanceHandle);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Returns:

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

6.3 Set parameters

Before connecting to the sensor (OpenSensor) or sending commands to the sensor (SensorCommand) parameters for both functions can be specified. Each parameter can be set by one of following functions.

Name: SetParameterInt

SetParameterInt

Description:

Set a 4 Byte integer parameter.

Declaration:

```
ERR_CODE SetParameterInt (DWORD instanceHandle, LPCTSTR paramName,
                          int paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPCTSTR paramName

paramName

Direction: [IN]

Valid value: All available values are listed at chapter [A.1](#) and [A.2](#) and following.

Description: Name of the parameter as string.

Parameter: int paramValue

paramValue

Direction: [IN]

Description: Value of the parameter.

Returns:

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterIntU**SetParameterIntU****Description:**

Set a 4 Byte integer parameter (Unicode version).

Declaration:

```

ERR_CODE SetParameterIntU (DWORD instanceHandle, LPCWSTR paramName,
    int paramValue);

```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as unicode string.**Parameter:** int paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterDWORD**SetParameterDWORD****Description:**

Set a 4 Byte unsigned integer parameter.

Declaration:

```

ERR_CODE SetParameterDWORD (DWORD instanceHandle, LPCTSTR paramName,
    DWORD paramValue);

```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as string.**Parameter:** DWORD paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterDWORDU**SetParameterDWORDU****Description:**

Set a 4 Byte unsigned integer parameter (Unicode version).

Declaration:

```
ERR_CODE SetParameterDWORDU (DWORD instanceHandle, LPCWSTR paramName,
                             DWORD paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as unicode string.**Parameter:** DWORD paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterDouble**SetParameterDouble****Description:**

Set a 8 Byte double parameter.

Declaration:

```
ERR_CODE SetParameterDouble (DWORD instanceHandle, LPCTSTR paramName,
                             double paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as string.**Parameter:** double paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterDoubleU**SetParameterDoubleU****Description:**

Set a 8 Byte double parameter (Unicode version).

Declaration:

```
ERR_CODE SetParameterDoubleU (DWORD instanceHandle, LPCWSTR paramName,
    double paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as unicode string.**Parameter:** double paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterString**SetParameterString****Description:**

Set a string (pointer to a character array) parameter.

Declaration:

```
ERR_CODE SetParameterString (DWORD instanceHandle, LPCTSTR paramName,
    LPCTSTR paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#) and [A.2](#) and following.**Description:** Name of the parameter as string.**Parameter:** LPCTSTR paramValue

paramValue

Direction: [IN]**Description:** Value of the parameter.**Returns:**

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterStringU**SetParameterStringU****Description:**

Set a string (pointer to a character array) parameter (Unicode version).

Declaration:

```
ERR_CODE SetParameterStringU (DWORD instanceHandle, LPCWSTR paramName,
                              LPCWSTR paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPCWSTR paramName

paramName

Direction: [IN]

Valid value: All available values are listed at chapter [A.1](#) and [A.2](#) and following.

Description: Name of the parameter as unicode string.

Parameter: LPCWSTR paramValue

paramValue

Direction: [IN]

Description: Value (as unicode) of the parameter.

Returns:

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Name: SetParameterStringL**SetParameterStringL****Description:**

Set a binary string (pointer to a character array, including zero characters) parameter with length.

Declaration:

```
ERR_CODE SetParameterStringL (DWORD instanceHandle, LPCTSTR paramName,
                              LPCTSTR paramValue, DWORD len);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPCTSTR paramName

paramName

Direction: [IN]

Valid value: All available values are listed at chapter [A.1](#) and [A.2](#) and following.

Description: Name of the parameter as string. Usefull for the parameter SP_CmdStr (in Cmd_Generic)

Parameter: DWORD len

len

Direction: [IN]

Description: Length of the binary string.

Parameter: LPCTSTR paramValue
Direction: [IN]
Description: Value of the parameter.

paramValue

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

6.4 Retrieving sensor answer

After sending a command to a sensor (SensorCommand) the answer can be retrieved at same manner as setting parameters.

Name: GetParameterInt**GetParameterInt****Description:**

Get a 4 Byte integer parameter.

Declaration:

```
ERR_CODE GetParameterInt (DWORD instanceHandle, LPCTSTR paramName,
                          int *paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPCTSTR paramName

paramName

Direction: [IN]

Valid value: All available values are listed at chapter [A.1](#).

Description: Name of the parameter as string.

Parameter: int *paramValue

paramValue

Direction: [OUT]

Description: Pointer to a variable retrieving the parameter

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.
 ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.
 ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL.

Name: GetParameterIntU**GetParameterIntU****Description:**

Get a 4 Byte integer parameter (Unicode version).

Declaration:

```
ERR_CODE GetParameterIntU (DWORD instanceHandle, LPCWSTR paramName,
    int *paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as unicode string.**Parameter:** int *paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable retrieving the parameter**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL or unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

Name: GetParameterDWORD**GetParameterDWORD****Description:**

Get a 4 Byte unsigned integer parameter.

Declaration:

```
ERR_CODE GetParameterDWORD (DWORD instanceHandle, LPCTSTR paramName,
    DWORD *paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as string.**Parameter:** DWORD *paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable retrieving the parameter**Returns:**

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL.

Name: GetParameterDWORDU**GetParameterDWORDU****Description:**

Get a 4 Byte unsigned integer parameter (Unicode version).

Declaration:

```

ERR_CODE GetParameterDWORDU (DWORD instanceHandle, LPCWSTR paramName,
    DWORD *paramValue);

```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as unicode string.**Parameter:** DWORD *paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable retrieving the parameter**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL or unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

Name: GetParameterDouble**GetParameterDouble****Description:**

Get a 8 Byte double parameter.

Declaration:

```

ERR_CODE GetParameterDouble (DWORD instanceHandle, LPCTSTR paramName,
    double *paramValue);

```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as string.**Parameter:** double *paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable retrieving the parameter**Returns:**

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL.

Name: GetParameterDoubleU**GetParameterDoubleU****Description:**

Get a 8 Byte double parameter (Unicode version).

Declaration:

```
ERR_CODE GetParameterDoubleU (DWORD instanceHandle, LPCWSTR paramName,
    double *paramValue);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCWSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as unicode string.**Parameter:** double * paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable retrieving the parameter**Returns:**

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue is NULL or unicode conversion failed.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

Name: GetParameterString**GetParameterString****Description:**

Get a string (character array) parameter.

Declaration:

```
ERR_CODE GetParameterString (DWORD instanceHandle, LPCTSTR paramName,
    LPTSTR paramValue, DWORD *maxLen);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** LPCTSTR paramName

paramName

Direction: [IN]**Valid value:** All available values are listed at chapter [A.1](#).**Description:** Name of the parameter as string.**Parameter:** LPTSTR paramValue

paramValue

Direction: [OUT]**Description:** Pointer to a variable (character buffer) retrieving the parameter.

Parameter: DWORD *maxLen

maxLen

Direction: [IN/OUT]

Description: The buffer must be allocated by the application. The size of the buffer is specified at maxLen. If the resulting string is larger than maxLen, it is truncated and ERR_NOMEMORY is returned. The real length of the string (maybe truncated) is returned in maxLen too. If LPTSTR paramValue is NULL, the length of the containing string is returned in maxLen.

Returns:

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if the pointer to paramValue or the pointer to maxLen is NULL.

ERR_NOMEMORY (-19) if the buffer is too short to hold the complete answer.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

Name: GetParameterStringU

GetParameterStringU

Description:

Get a string (character array) parameter (Unicode version).

Declaration:

```
ERR_CODE GetParameterStringU (DWORD instanceHandle, LPCWSTR paramName,
    LPWSTR paramValue, DWORD *maxLen);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPCWSTR paramName

paramName

Direction: [IN]

Valid value: All available values are listed at chapter [A.1](#).

Description: Name of the parameter as unicode string.

Parameter: LPWSTR paramValue

paramValue

Direction: [OUT]

Description: Pointer to a variable (unicode buffer) retrieving the parameter.

Parameter: DWORD *maxLen

maxLen

Direction: [IN/OUT]

Description: The buffer must be allocated by the application. The size of the buffer is specified at maxLen. If the resulting string is larger than maxLen, it is truncated and ERR_NOMEMORY is returned. The real length of the string (maybe truncated) is returned in maxLen too. If LPTSTR paramValue is NULL, the length of the containing string is returned in maxLen.

Returns:

ERR_NOERROR (0) on success.
 ERR_WRONG_PARAMETER (-18) if the pointer to paramValue or the pointer to maxLen is NULL or conversion to unicode failed.
 ERR_NOMEMORY (-19) if the buffer is too short to hold the complete answer.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.
 ERR_NOT_FOUND (-25) if the parameter (specified by paramName) is not found.

6.5 Clear internal parameter buffer

Before building up a new command using the SetParameter... functions, the internal parameter buffer should be cleared. This avoids parameter mismatch between several commands.

Name: ClearAllParameters**ClearAllParameters****Declaration:**

```
ERR_CODE ClearAllParameters (DWORD instanceHandle);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

6.6 Connecting to the sensor

Name: OpenSensor**OpenSensor****Description:**

Establish the connection to the sensor.

After connecting to the sensor, all internal parameters starting with SP_, CP_ and IP_ are cleared (like at function ClearAllParameters).

Attention! Please don't forget to close the connection at the end. Otherwise it can come to unpredictable behaviour when resources leave open.

Declaration:

```
ERR_CODE OpenSensor (DWORD instanceHandle);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.
 ERR_ALREADY_OPEN (-11) if the specified connection is already open.
 ERR_INTERFACE_NOT_SUPPORTED (-10) if the specified interface (parameter "IP_Interface") is not supported.
 ERR_CANNOT_CREATE_INTERFACE (-12) if the interface cannot be created.
 ERR_CANNOT_OPEN (-2) if interface cannot be opened.
 ERR_WRONG_PARAMETER (-18) if a parameter is not valid (e.g. out of range).
 ERR_NOMEMORY (-19) if there is not enough memory to allocate driver memory.
 ERR_APPLYING_PARAMS (-4) if parameters cannot applied to driver.

6.7 Closing connection to sensor**Name:** CloseSensor**CloseSensor****Description:**

Close the connection to the sensor.

Declaration:

```
ERR_CODE CloseSensor (DWORD instanceHandle);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

6.8 Sending commands to the sensor**Name:** SensorCommand**SensorCommand****Description:**

Before sending a command to the sensor the command and parameters must be specified with SetParameter... functions. After the command is executed the answer can be read using GetParameter... functions.
 At start of this function, all internal parameters starting with SA_, and IA_ are cleared (like at function ClearAllParameters).
 At end of this function, all internal parameters starting with SP_, CP_, and IP_ are cleared (like at function ClearAllParameters).

Declaration:

```
ERR_CODE SensorCommand (DWORD instanceHandle);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Returns:

ERR_NOERROR (0) on success.
 ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.
 ERR_NOT_OPEN (-3) if connection is not established
 ERR_UNKNOWN_SENSOR_COMMAND (-14) if parameter "S_Command" is not valid.
 ERR_BAUDRATE_TOO_LOW (-22) if at sensor IFD2430 over RS232 with Baudrate < 38400 the CCD line (Get_CCD) or other binary data (Get_DarkSig, Get_...) should be read.
 ERR_APPLYING_PARAMS (-4) if interface parameters changed by command (e.g. change baudrate) and cannot be applied.
 ERR_NO_ANSWER_RECEIVED (-20) if no answer but other data is received within answer time.
 ERR_TIMEOUT_READING_FROM_SENSOR (-8) if no answer is received within answer time.
 ERR_CLEARUNG_BUFFER (-6) for command "Clear_Buffers" if specified device could not be cleared.
 ERR_NOMEMORY (-19) if a data buffer could not be allocated.
 ERR_UNKNOWN_SENSOR_ANSWER (-15) if sensor answer is not an answer to a command.
 ERR_SENSOR_ANSWER_ERROR (-16) if sensor answer cannot be interpreted or sensor returned an error number.
 ERR_SENSOR_ANSWER_TOO_SHORT (-17) if sensor answer is too short.
 ERR_SENSOR_ANSWER_DOES_NOT_MATCH_COMMAND (-21) if answer does not match the command.

6.9 Polling data from sensor

Name: Poll**Poll****Description:**

Get the latest values from sensor.

Declaration:

```
ERR_CODE Poll (DWORD instanceHandle, int *rawData, double *scaledData,
               int maxValues);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]**Description:** Number of the sensor instance, previously returned by CreateSensorInstance.**Parameter:** int *rawData

rawData

Direction: [OUT]**Description:** Pointer to value (or array of values) to retrieve latest data frame from sensor as raw values.**Parameter:** double *scaledData

scaledData

Direction: [OUT]**Description:** Pointer to value (or array of values) to retrieve latest data frame from sensor scaled by sensor range. If rawData is null of scaledData is null, no data is transferred for this parameter.

Parameter: `int maxValues`

`maxValues`

Direction: [IN]

Description: Length of `rawData` and `scaledData`. Some sensors can measure more than one value per measure cycle (e.g. IFD's which measures distance, intensity, ... or ODC's which can measure two segments). In this case `maxValues` can be set up to frame size (values per measure cycle).

If `maxValues` is smaller than frame size only the first `maxValues` values are transferred.

If `maxValues` is greater than frame size, values of more than one frame are transferred.

Returns:

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if `instanceHandle` is not valid.

ERR_NOT_OPEN (-3) if connection is not established

ERR_NO_SENSORDATA_AVAILABLE (-13) if the receive buffer does not contain any value from sensor.

6.10 Number of values available to read

Name: `DataAvail`

DataAvail

Description:

The values available returned may be divided by frame size to calculate the number of frames available. A frame is the set of values transmitted by the sensor for each one measurement.

Declaration:

```
ERR_CODE DataAvail (DWORD instanceHandle, int *avail);
```

Parameter: `DWORD instanceHandle`

`instanceHandle`

Direction: [IN]

Description: Number of the sensor instance, previously returned by `CreateSensorInstance`.

Parameter: `int *avail`

`avail`

Direction: [OUT]

Description: Pointer to value to retrieve number of values available from sensor

Returns:

ERR_NOERROR (0) on success.

ERR_INSTANCE_NOT_EXIST (-24) if `instanceHandle` is not valid.

ERR_NOT_OPEN (-3) if connection is not established.

ERR_WRONG_PARAMETER (-18) if `avail` is null.

6.11 Block wise data acquisition from sensor

Name: TransferData

TransferData

Description:

Transfer the data from driver to application.

Declaration:

```
ERR_CODE TransferData (DWORD instanceHandle, int *rawData, double *scaledData,
    int maxValues, int *read);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: int *rawData

rawData

Direction: [OUT]

Description: Pointer to array of values to retrieve data from sensor as raw values

Parameter: double * scaledData

scaledData

Direction: [OUT]

Description: Pointer to array of values to retrieve data frame from sensor scaled by sensor range.
If rawData is null or scaledData is null, no data is transferred for this parameter. If both parameters are null all buffers used are emptied.

Parameter: int maxValues

maxValues

Direction: [IN]

Description: Length of rawData and scaledData. It should be a multiple of frame size, otherwise the rest of the last frame is lost, because TransferData always starts with frame start.

Parameter: int * read

read

Direction: [OUT]

Description: Real number of values transferred.

Returns:

ERR_NOERROR (0) on success.
ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.
ERR_NOT_OPEN (-3) if connection is not established.
ERR_NO_SENSORDATA_AVAILABLE (-13) if no values per data frame are specified in driver. Either use 'Use_Defaults' or read information from sensor to set this value.

6.12 Get additional error information

Name: GetError

GetError

Description:

Get the last error in driver.

This function can be used after an error return from function OpenSensor, CloseSensor, SensorCommand, DataAvail, TransferData and Poll.

For all functions above and error code ERR_INSTANCE_NOT_EXIST this function does not return an error text.

For function DataAvail and error code ERR_WRONG_PARAMETER this function does not return an error text.

Declaration:

```
ERR_CODE GetError (DWORD instanceHandle, LPTSTR errText, DWORD maxLen);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPTSTR errText

errText

Direction: [OUT]

Description: String buffer to get extended error string.

Parameter: DWORD maxLen

maxLen

Direction: [IN]

Description: Length of string buffer. If the error text is longer as maxLen it is truncated. The string is null terminated.

Returns:

ERR_NOERROR (0) on success.

ERR_OVERFLOW (-23) if there was a buffer overflow or data could not be interpreted.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Any other error value returned by previous function.

Name: GetErrorU

GetErrorU

Description:

Get the last error in driver (Unicode version). Rest of functionality is identical to GetError.

Declaration:

```
ERR_CODE GetErrorU (DWORD instanceHandle, LPWSTR errText, DWORD maxLen);
```

Parameter: DWORD instanceHandle

instanceHandle

Direction: [IN]

Description: Number of the sensor instance, previously returned by CreateSensorInstance.

Parameter: LPWSTR errText

errText

Direction: [OUT]

Description: Unicode string buffer to get extended error string.

Parameter: DWORD maxLen

maxLen

Direction: [IN]

Description: Length of string buffer. If the error text is longer as maxLen it is truncated. The string is null terminated.

Returns:

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if conversion to unicode failed.

ERR_OVERFLOW (-23) if there was a buffer overflow or data could not be interpreted.

ERR_INSTANCE_NOT_EXIST (-24) if instanceHandle is not valid.

Any other error value returned by previous function.

6.13 Get version of MEDAQLib dll

Name: GetDLLVersion

GetDLLVersion

Description:

Retrieves the version of the MEDAQLib dll.

This function can be called at any time, no sensor instance is needed. The version is stored in versionStr and is limited to length of maxLen (should be at least 11 bytes).

Declaration:

```
ERR_CODE GetDLLVersion (LPTSTR versionStr, DWORD maxLen);
```

Parameter: LPTSTR versionStr

versionStr

Direction: [OUT]

Description: String buffer to get version info.

Parameter: DWORD maxLen

maxLen

Direction: [IN]

Description: Length of string buffer. If the version info is longer as maxLen it is truncated. The string is null terminated.

Returns:

ERR_NOERROR (0) on success.

Name: GetDLLVersionU

GetDLLVersionU

Description:

Retrieves the version of the MEDAQLib dll (Unicode version). Rest of functionality is identical to GetDLLVersion.

Declaration:

```
ERR_CODE GetDLLVersionU (LPWSTR versionStr, DWORD maxLen);
```

Parameter: LPWSTR versionStr

versionStr

Direction: [OUT]

Description: Unicode string buffer to get version info.

Parameter: DWORD maxLen

maxLen

Direction: [IN]

Description: Length of string buffer. If the version info is longer as maxLen it is truncated. The string is null terminated.

Returns:

ERR_NOERROR (0) on success.

ERR_WRONG_PARAMETER (-18) if conversion to unicode failed.

Any other error value returned by previous function.

A Parameters

Parameters for opening the interface (Interface parameters, IP_...) are used in the Function OpenSensor.

Parameters for communicating with the sensor (Sensor parameters, SP_...) and answer from sensor (Sensor answer, SA_...) are used in Function SensorCommand.

Direction in parameters is defined as follows:

Down: From application to driver or sensor.

Up: From sensor or driver to application.

Parameters without a default value are obligatory and must be specified. For the other parameters, the default value is used if not specified.

When a parameter contains two periods (e.g: SP_X1..16), it stands for a sequence of parameters.

A.1 Interface parameters

A.1.1 All Interfaces

Parameter: String IP_Interface

IP_Interface

Direction: Down

Valid values:

"RS232"

"IF2004"

"IF2008"

"TCP/IP"

"DriverX_USB"

"USBIO"

Description: Interface type where the sensor is plugged on. If the RS422/USB converter is used "RS232" must be set, because a RS232 interface is emulated.

With an RS422/RS232 to Ethernet converter additional sensors (except IFD2430) can be connected via "TCP/IP".

Parameter: int IP_ScaleErrorValues

IP_ScaleErrorValues

Direction: Down

Valid values:

1 = "last valid value"

2 = "set to -1"

3 = "set to negative error value"

Default: 1

Description: Attention! Normal sensor values can get negative too, so check raw values for errors additionally.

If sensor values are not valid, they cannot be scaled. So invalid values can be set to the last valid value or can be set to -1 or can be set to a negative error value:

For ODC's:

A Parameters

-10	DSP No edge
-11	DSP At the beginning of the picture
-12	DSP At the end of the picture
-13	DSP Dark - bright edge
-14	DSP Bright - dark edge
-15	DSP Min. number of edges
-16	DSP Max. number of edges
-17	DSP Invalid measuring program
-18	DSP Segment 1st edge \geq 2nd edge
-19	DSP Segment number of edges $<$ last edge
-20	DSP Invalid working distance
-22	ARM Laser off
-23	ARM Invalid float
-24	ARM DMA setup error
-90	ASCII mode at IF2004 not supported

For ILD's (not every sensor supports all errors):

-1	F1 bad objekt (no objekt cognizable)
-2	F2 out of range + (to near at sensor)
-3	F3 out of range - (to far from sensor)
-4	F4 poor target (objekt not evaluable)
-5	F5 Laser off (external laser off)

For SENSOR_ILR118x:

For error values, scaled values are set to 0, raw values are set to positive error values:

A Parameters

15	E15 - Excessively poor reflexes. Distance sensor (Front edge) against target < 0.1m.
16	E16 - Excessively strong reflexes.
17	E17 - Too much steady light (for example sun).
18	E18 - Only in DX mode (50 Hz): Too much difference between measured and pre-calculated value.
19	E19 - Only in DX mode (50 Hz): Target motion speed > 10 m/s.
23	E23 - Temperature below -10 °C
24	E24 - Temperature above +60 °C
31	E31 - Faulty EEPROM checksum, hardware error.
51	E51 - Failure to set avalanche voltage of diode laser. 1. straylight or 2. hardware error.
52	E52 - Laser current too high / laser defective.
53	E53 - One or more parameters in the EEPROM not set (Consequence: Division by 0).
54	E54 - Hardware error (PLL).
55	E55 - Hardware error.
61	E61 - Used parameter is inadmissible, invalid command sent.
62	E62 - 1. Hardware error 2. wrong value in interface communication (Parity error SIO).
63	E63 - SIO overflow.
64	E64 - Framing-Error SIO.

For SENSOR_ILR1191:

For error values, scaled values are set to 0, raw values are set to positive error values:

2	E02 - No target.
4	E04 - Laser defect.

Parameter: int IP_RingBufferSize

IP_RingBufferSize

Direction: Down

Valid values:

Minimum: 512

Maximum: 10485760

Unit: Bytes

Default: SENSOR_ILR110x_115x: 8 KB,
 SENSOR_ILR118x: 4 KB, SENSOR_ILR1191: 40 KB,
 SENSOR_ILD1302: 10 KB,
 SENSOR_ILD1401: 4 KB, CONTROLLER_ILD1401: 20 KB,
 SENSOR_ILD1402: 20 KB,
 SENSOR_ILD1700: 30 KB, SENSOR_ILD1800: 20 KB,
 SENSOR_ILD2000: 40 KByte, SENSOR_ILD2200: 240 KB,
 SENSOR_IFD2400: 20 KB, SENSOR_IFD2401: 2 MB,
 SENSOR_IFD2430: 3 MB, SENSOR_IFD2431: 10 MB,
 SENSOR_ODC2500: 40 KB, SENSOR_ODC2600: 40 KB,
 CONTROLLER_ESC4912: 500 KB,
 SENSOR_DT6100: 62 KB, CONTROLLER_DT6500: 500 KB,

CONTROLLER_KSS6380: 32 KB,
ENCODER_IF2004: 80 KB, PCI_CARD_IF2008: 160 KB

Description: Data (values and sensor answer) is collected from sensor into a ring buffer. From there it is converted (value) or interpreted (sensor answer). If the ring buffer size is too small and data is not transferred fast enough an overflow occur. The buffer must be large enough to hold a complete sensor answer (especially IFD's with command Get_CCD or Get_DarkSig, ...).

Parameter: int IP_LogType

IP_LogType

Direction: Down

Valid values:

A bit combination of following values:
0= LOG_NONE (No logging)
1= HIGH_TYPE (User <--> MEDAQLib)
2= MIDDLE_TYPE (Sensor layer <--> Interface layer)
4= LOW_TYPE (MEDAQLib <--> Hardware driver)
8= ERROR_TYPE (Any errors reported by MEDAQLib)
16= DRIVER_TYPE (Hardware driver <--> Sys driver)

Default: LOG_NONE

Description: This parameter enables logging to file for debugging purposes.

Parameter: int IP_LogLevel

IP_LogLevel

Direction: Down

Valid values:

A bit combination of following values:
1= EMERGENCY_LEVEL (logging emerging errors)
2= CRITICAL_LEVEL (logging critical errors)
4= ERROR_LEVEL (logging errors which occurs)
8= WARNING_LEVEL (logging warnings from MEDAQLib)
16= NOTICE_LEVEL (logging notices)
32= TRACE_LEVEL (logging function calls)
64= DATA_LEVEL (logging data in binary mode)

Description: This parameter specifies the kind of event to log.

If IP_LogType is set to LOG_NONE, this parameter is not obligatory.

Parameter: String IP_LogFile

IP_LogFile

Direction: Down

Valid values:

File name of log file.

Description: This parameter specifies the file where to log.

If IP_LogType is set to LOG_NONE, this parameter is not obligatory.

Parameter: int IP_LogAppend

IP_LogAppend

Direction: Down

Valid values:

0= FALSE
1= TRUE

Default: 1

Description: This parameter specifies if the logfile should be cleared at opening or if the new data should be appended to file.

A Parameters

Parameter: int IP_LogFlush

IP_LogFlush

Direction: Down

Valid values:

0 = FALSE

1 = TRUE

Default: 0

Description: This parameter specifies if the logfile should be flushed after each output. In this case, it is sure that all information is stored to logfile before proceeding. But depending on the storage device it can slow down the MEDAQLib.

Parameter: int IP_MaxPacketSize

IP_MaxPacketSize

Direction: Down

Valid values:

Minimum: 1

Maximum: INT_MAX (2147483647)

Unit: Bytes

Default: SENSOR_ILD1401: 1, otherwise INT_MAX

Description: Maximum size of a block which can be transferred to the sensor at once. Because of a small receive FIFO in ILD1401 only 1 Byte after another can be sent to sensor with a break between.

Parameter: int IP_PacketDelay

IP_PacketDelay

Direction: Down

Valid values:

Minimum: 0

Maximum: INT_MAX (2147483647)

Unit: ms

Default: SENSOR_ILD1401: 1 (RS232) or 3 (TCP/IP), otherwise 0

Description: Break time between sending two blocks to a sensor (see IP_MaxPacketSize).

A.1.2 RS232

Following sensors supports this interface:

SENSOR_ILD1401, SENSOR_ILD1800, SENSOR_ODC2500, SENSOR_ODC2600, SENSOR_IFD2400, SENSOR_IFD2430, SENSOR_ASP5500, SENSOR_IFD2401, SENSOR_IFD2431, SENSOR_ILR110x_115x, SENSOR_ILR118x, SENSOR_ILR1191 (native).

SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ILD2200 (additional, e.g. RS422_USB-Converter and RS232 high level interface).

Parameter: String IP_Port

IP_Port

Direction: Down

Valid values:

"COM1"

"COM2"

...

Description: Name of the serial interface. Before opening the interface using CreateFile, the string is prefixed with "\\.\\".

A Parameters

Parameter: int IP_Baudrate Direction: Down Valid values: Minimum: 0 Unit: Baud Default: SENSOR_ILR110x_115x: 38400, SENSOR_ILR118x: 9600, SENSOR_ILD1401: 38400, SENSOR_ILD2200: 691500, otherwise 115200 Description: Baudrate of the serial connection.	IP_Baudrate
Parameter: int IP_Stopbits Direction: Down Valid values: 0= ONESTOPBIT 1= ONE5STOPBITS 2= TWOSTOPBITS Default: SENSOR_ODC2500 and SENSOR_ODC2600: 2 (RS232) or 0 (others), otherwise 0 Description: Number of stop bits of the serial connection.	IP_Stopbits
Parameter: int IP_Parity Direction: Down Valid values: 0= NOPARITY 1= ODDPARITY 2= EVENPARITY 3= MARKPARITY 4= SPACEPARITY Default: SENSOR_ILR110x_115x: 2, otherwise 0 Description: Parity of the serial connection.	IP_Parity
Parameter: int IP_ByteSize Direction: Down Valid values: 7 8 Unit: Bit Default: SENSOR_ILR110x_115x: 7, otherwise 8 Description: Number of data bits per byte of the serial connection.	IP_ByteSize

A.1.3 IF2004

Following sensors supports this interface:

SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ILD1800, SENSOR_ILD2200, SENSOR_ODC2500, SENSOR_ODC2600, SENSOR_ILD2000, ENCODER_IF2004 (native).

Parameter: int IP_CardInstance Direction: Down Valid values: Minimum: 0 Maximum: 15 Default: 0 Description: Instance number of the IF2004 Interface card. The cards are enumerated by the OS and the only way to distinguish is the card instance number. It does not change at least there are no changes at the PCI bus.	IP_CardInstance
---	-----------------

A Parameters

Parameter: int IP_ChannelNumber

IP_ChannelNumber

Direction: Down

Valid values:

Minimum: ENCODER_IF2004: -1, otherwise 0

Maximum: 3

Default: ENCODER_IF2004: 3, otherwise obligatory

Description: Channel number on IF2004 Interface card. If the Encoder on the IF2004 card should be used to store values synchronously to a sensor, the channel number 3 is reserved for it. Otherwise the FIFO cannot be used for Encoder. Sensors can be carried on each channel.

Parameter: int IP_UseGate

IP_UseGate

Direction: Down

Valid values:

0= FALSE

1= TRUE

Default: 0

Description: The gate input of the card (5V TTL signal) can be used to lock or free the FIFO for data from sensors. This parameter affects always two channels (0+1 or 2+3) because they are on the same connector. The encoder can be locked for FIFO too.

Parameter: int IP_Baudrate

IP_Baudrate

Direction: Down

Valid values:

SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ODC2500 and SENSOR_ODC2600: 115200

SENSOR_ILD1800, SENSOR_ILD2000, SENSOR_ODC2500 and SENSOR_ODC2600: 691200

SENSOR_ILD2200: 691200 or 1250000

Unit: Baud

Default: SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700: 115200, otherwise 691200

Description: Speed of the RS422 serial connection. Only the ODC sensors can be used with different baud rates.

A.1.4 IF2008

Following sensors supports this interface:

SENSOR_ILD1401 (for sensor ILD1402 in compatibility mode).

SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ILD2200, SENSOR_ODC2500, SENSOR_ODC2600, PCI_CARD_IF2008, SENSOR_IFD2401, SENSOR_IFD2431, SENSOR_ILR110x_115x, SENSOR_ILR118x, SENSOR_ILR1191 (native).

Parameter: int IP_CardInstance

IP_CardInstance

Direction: Down

Valid values:

Minimum: 0

Maximum: 15

Default: 0

Description: Instance number of the IF2008 Interface card. The cards are enumerated by the OS and the only way to distinguish is the card instance number. It does not change at least there are no changes at the PCI bus.

A Parameters

Parameter: int IP_ChannelNumber

IP_ChannelNumber

Direction: Down

Valid values:

Condition: SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ILD2200, SENSOR_ODC2500, SENSOR_ODC2600, SENSOR_IFD2401, SENSOR_IFD2431, SENSOR_ILR110x_115x, SENSOR_ILR118x, SENSOR_ILR1191

0= Sensor 1 (Base Board, Connector X1)

1= Sensor 2 (Base Board, Connector X1)

2= Sensor 3 (Base Board, Connector X2)

3= Sensor 4 (Base Board, Connector X2)

4= Sensor 5 (Extension Board, Connector X1)

5= Sensor 6 (Extension Board, Connector X1)

Valid values:

Condition: PCI_CARD_IF2008

-1= No data acquisition

6= Encoder 1

7= Encoder 2

8= Digital IN

9= Digital Rx/D

10= ADC 1 (Analog/Digital converter)

11= ADC 2 (Analog/Digital converter)

Description: Channel number on IF2008 Interface card. Attention! Sensor 5 and 6 are only available if IF2008E extension card is installed. Digital IN is only available if IF2008E extension card or IF2008IO extension slot is installed. ADC is only available if IF2008E extension card is installed. -1 means, no data channel is written to FIFO and can be read using TransferData or Poll. This mode can be used if IF2008 should only be parametrized.

Parameter: int IP_Baudrate

IP_Baudrate

Direction: Down

Valid values:

SENSOR_ILD1401: 38400

SENSOR_ILD1302, SENSOR_ILD1402: 115200, 57600, 38400, 19200, 9600

SENSOR_ILD1700: 115200, 57600, 19200, 9600

SENSOR_ILD2200: 691200, 1250000

SENSOR_IFD2401, SENSOR_IFD2431: 460800, 230400, 115200, 57600, 38400, 19200, 9600

SENSOR_ODC2500, SENSOR_ODC2600: 115200, 691200

SENSOR_ILR110x_115x: 57600, 38400, 19200, 9600, 4800

SENSOR_ILR118x: 38400, 19200, 9600, 4800, 2800

SENSOR_ILR1191: 460800, 230400, 115200, 57600, 38400, 19200, 9600

Unit: Baud

Default: SENSOR_ILD1401, SENSOR_ILR118x: 38400, SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_IFD2401, SENSOR_IFD2431, SENSOR_ILR1191: 115200, SENSOR_ILD2200, SENSOR_ODC2500, SENSOR_ODC2600: 691200, SENSOR_ILR110x_115x: 57600

Description: Speed of the RS422 serial connection.

A Parameters

Parameter: int IP_Parity

IP_Parity

Direction: Down

Valid values:

SENSOR_ILR110x_115x: 2= EVENPARITY

SENSOR_ODC2500, SENSOR_ODC2600: 0= NOPARITY, 2= EVENPARITY

otherwise: 0= NOPARITY

Default: SENSOR_ILR110x_115x: 2, otherwise 0

Description: Parity of the RS422 serial connection.

A.1.5 TCP/IP

Following sensors supports this interface:

SENSOR_ILD1800 (additional, e.g. RS232 to TCP/IP comm server and RS232 high level interface).

SENSOR_ILD1401, SENSOR_ILD1302, SENSOR_ILD1402, SENSOR_ILD1700, SENSOR_ILD2200, SENSOR_ODC2500, SENSOR_ODC2600, SENSOR_IFD2400, SENSOR_ASP5500, SENSOR_IFD2401, SENSOR_IFD2431, SENSOR_ILR110x_115x, SENSOR_ILR118x, SENSOR_ILR1191 (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

SENSOR_IFD2430, CONTROLLER_ESC4912, SENSOR_DT6100, CONTROLLER_KSS6380, CONTROLLER_DT6500 (native).

Parameter: String IP_RemoteAddr

IP_RemoteAddr

Direction: Down

Valid value: IP address

Description: IP address of the remote sensor (server). It has to be change in any case because the default address is not valid!

Parameter: int IP_RemotePort

IP_RemotePort

Direction: Down

Valid values:

Minimum: 1

Maximum: 65535

Default: SENSOR_IFD2430: 50/1000, otherwise 10001

Description: TCP port of the remote sensor (server). For IFD2430 this parameter is not used. Because sensor uses two ports (port 50 for data, port 1000 for commands), two interfaces (with this fixed ports) are generated by the driver.

A.1.6 DriverX_USB

Following sensors supports this interface:

SENSOR_IFD2401, SENSOR_IFD2431 (native).

Attention! This driver allows opening the same device twice, but communication does not work in this case.

Parameter: int IP_DeviceInstance

IP_DeviceInstance

Direction: Down

Valid values:

A Parameters

Minimum: 0
Maximum: 255

Default: 0

Description: Instance number of the USB device. The devices are enumerated by the OS and the only way to distinguish is the device instance number. It does not change at least there are no changes at the USB bus (plug/unplug devices).

Parameter: int IP_UsbReadBufCnt

IP_UsbReadBufCnt

Direction: Down

Valid values:

Minimum: 2
Maximum: 32

Default: 4

Description: Number of buffers for read operations on USB.

Parameter: int IP_UsbReadBufSize

IP_UsbReadBufSize

Direction: Down

Valid values:

Minimum: 512
Maximum: 65536

Unit: Bytes

Default: 512

Description: Buffer size for read operations on USB. The value is always ceiled to the next power of two (512, 1024, 2048, ..., 32768, 65526).

A.1.7 USBIO

Following sensors supports this interface:

CONTROLLER_ILD1401 (native).

Parameter: int IP_DeviceInstance

IP_DeviceInstance

Direction: Down

Valid values:

Minimum: 0
Maximum: 255

Default: 0

Description: Instance number of the USB device. The devices are enumerated by the OS and the only way to distinguish is the device instance number. It does not change at least there are no changes at the USB bus (plug/unplug devices).

Parameter: int IP_UsbReadBufCnt

IP_UsbReadBufCnt

Direction: Down

Valid values:

Minimum: CONTROLLER_ILD1401: 0, otherwise 2
Maximum: 32

Default: CONTROLLER_ILD1401: 0 (fixed), otherwise 4

Description: Number of buffers for read operations on USB.

A Parameters

Parameter: int IP_UsbReadBufSize IP_UsbReadBufSize
Direction: Down
Valid values:
 Minimum: 2
 Maximum: 65536
Unit: Bytes
Default: CONTROLLER_ILD1401: 10, otherwise 512
Description: Buffer size for read operations on USB. The value is always ceiled to the next even number (2, 4, 6, ..., 65534, 65536).

A.2 Communication via SensorCommand

The function SensorCommand always must have one obligatory parameter:

Parameter: String S_Command S_Command
Direction: Down
Valid for sensor: all
Valid value: See commands for sensors.
Default: "" (nothing, not valid)
Description: The command to execute is specified by this parameter. Some commands are for the driver, the most commands are processed by the sensor.

The following parameters are used for each call to SensorCommand so they are described only once:

Parameter: int CP_SensorAnswerTimeout CP_SensorAnswerTimeout
Direction: Down
Valid for sensor: all
Valid values:
 Minimum: 0
 Maximum: INT_MAX (2147483647)
Unit: ms
Default: Depending on sensor and command.
Description: The timeout waiting for the complete answer from sensor.

Parameter: String SA_CompleteAnswer SA_CompleteAnswer
Direction: Up
Valid for sensor: all
Description: The raw (not interpreted) answer (to a command) from the sensor is stored here.

Parameter: int SA_ErrorNumber SA_ErrorNumber
Direction: Up
Valid for sensor:
 SENSOR_ILD1302
 SENSOR_ILD1401
 SENSOR_ILD1402
 SENSOR_ILD1700
 SENSOR_ILD1800
 SENSOR_ILD2200
 SENSOR_ODC2500
 SENSOR_ODC2600
Valid values:
 Minimum: 0
 Maximum: 255
Description: Error number returned by any sensor if a command was not successful.

Parameter: String SA_ErrorText

SA_ErrorText

Direction: Up

Valid for sensor:

SENSOR_ILR118x
 SENSOR_ILR1191
 SENSOR_ILD1302
 SENSOR_ILD1402
 SENSOR_ILD1700
 SENSOR_ILD1800
 SENSOR_ILD2000
 SENSOR_ILD2200
 SENSOR_IFD2400
 SENSOR_IFD2430
 SENSOR_IFD2401
 SENSOR_IFD2431
 SENSOR_ODC2500
 SENSOR_ODC2600
 CONTROLLER_ESC4912
 SENSOR_DT6100
 CONTROLLER_DT6500

Description: Clear text for specific error number.

Following error numbers and strings can be set by ILD1401:

0x02	"Command error"
0x03	"Faulty number of parameters"
0x04	"Timeout"

Following error numbers and strings can be set by ILD1302, ILD1402, ILD1700, ILD1800 and ILD2200:

0x01	"Command unknown"
0x02	"Incorrect parameter value"
0x03	"Invalid parameter"
0x04	"Timeout"
0x05	"Command failed"
0x06	"Warning for averaging type and averaging number"

Following error strings can be set by ILR110x_115x:

"NAK" (Error in command)

Following error strings can be set by ILR118x:

A Parameters

"E15" (Excessively poor reflexes. Distance sensor (Front edge) against target < 0.1m.)
"E16" (Excessively strong reflexes.)
"E17" (Too much steady light (for example sun).)
"E18" (Only in DX mode (50 Hz): Too much difference between measured and pre-calculated value.)
"E19" (Only in DX mode (50 Hz): Target motion speed > 10 m/s.)
"E23" (Temperature below -10 °C)
"E24" (Temperature above +60 °C)
"E31" (Faulty EEPROM checksum, hardware error.)
"E51" (Failure to set avalanche voltage of diode laser. 1. straylight or 2. hardware error.)
"E52" (Laser current too high / laser defective.)
"E53" (One or more parameters in the EEPROM not set (Consequence: Division by 0).)
"E54" (Hardware error (PLL).)
"E55" (Hardware error.)
"E61" (Used parameter is inadmissible, invalid command sent.)
"E62" (1. Hardware error 2. wrong value in interface communication (Parity error SIO).)
"E63" (SIO overflow.)
"E64" (Framing-Error SIO.)

Following error strings can be set by ILR1191:

"?" (Unknown command)

Following error strings can be set by ILD2000:

"Timeout"
"Wrong checksum"
"Too much data"
"Error writing to EEPROM"

Following error strings can be set by IFD2400, IFD2430, IDF2401 and IFD2431:

"error"
"not valid"

Following error numbers and strings can be set by ODC2500 and ODC2600:

A Parameters

0x01	"Error destination"
0x02	"Error source"
0x03	"Error length"
0x04	"Too much data received"
0x06	"Flash access error"
0x07	"Error erase flash"
0x08	"Error flash sector"
0x09	"Error Video"
0x0a	"Error on writing to the RAM"
0x0b	"Incorrect data transmitted, see 'Valid values'"
0x0c	"Incorrect measurement program number"
0x0d	"Error light reference tuning"

Following error strings can be set by ESC4912, DT6100 and DT6500:

"UNKNOWN COMMAND"
"WRONG PARAMETER"
"TIMEOUT"
"ERROR NO CH1"
"ERROR DATARATE TO HIGH"

Following parameters affects the driver and interface (when communicating with the sensor):

Parameter: int IP_ClearRingBuffer

IP_ClearRingBuffer

Direction: Down

Valid Interface: all

Valid for sensor: all

Valid values:

0 = FALSE

1 = TRUE

Default: 1

Description: Clears the ring buffer before sending the command to the sensor. The containing data is discarded. So the next data in the ring buffer is the sensor answer. For IF2004 card, the ring buffer is cleared after reading the answer too.

Parameter: int IP_ClearSendBuffer

IP_ClearSendBuffer

Direction: Down

Valid Interface: all

Valid for sensor: all

Valid values:

0 = FALSE

1 = TRUE

Default: 1

Description: Clears the send buffer (PurgeComm (PURGE_TXCLEAR)) before sending the command to the sensor.

Parameter: int IP_ClearReceiveBuffer

IP_ClearReceiveBuffer

Direction: Down

Valid Interface:

RS232

IF2004

Valid for sensor: all

Valid values:

0= FALSE

1= TRUE

Default: 1

Description: Clears the receive buffer (PurgeComm (PURGE_RXCLEAR) for RS232, reading the FIFO for IF2004 and IF2008) before sending the command to the sensor.

This example shows how to set measure speed of ILD1700

```
SetParameterString (instance, "S_Command", "Set_Speed");
SetParameterInt (instance, "SP_Speed", 1); \ 1 is 1.25 kHz
err= SensorCommand (instance);
/* error handling, if err!=ERR_SUCCESS*/
```

The next example shows how to get information from IFD2401

```
int sensor;
SetParameterString (instance, "S_Command", "Get_Status");
err= SensorCommand (instance);
/* error handling, if err!=ERR_SUCCESS*/
GetParameterInt (instance, "SA_Sensor", &sensor);
```

A.3 Sensor commands valid for each sensor

Commands are stored in Parameter S_Command. Following chapters describe commands and the parameters for each command.

A.3.1 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Hint! The normal way to tell the driver the sensor information is calling the command Get_Settings or Get_Info (depending on sensor). This command should only be used if bidirectional communication with the sensor is not possible or must be avoided.

The following parameters are interface parameters and already described at chapter [A.1](#): IP_ScaleErrorValues and IP_RingBufferSize. They are valid for each sensor and interface and can be changed using this command (not only when opening the driver).

The other parameters are sensor parameters (used by the sensor interface part of MEDAQLib).

Parameters only valid for a specific sensor are not described here but at command Use_Defaults of specific sensor chapter.

A Parameters

Parameter: int IP_CheckRingBufferOverflow

Direction: Down

Valid for sensor: all

Valid values:

0 = off

1 = on

Description: If it is set to 1, the ring buffer is monitored. If there is an overflow, an error is returned by GetError function.

IP_CheckRingBufferOverflow

Parameter: int IP_DataOn

Direction: Down

Valid for sensor: all

Valid values:

0 = off

1 = on

Description: Tells the driver if sensor is sending data. If it is 0 no timeout check is performed. If data (values, no answer) from sensor is received it is automatically turned on.

IP_DataOn

Parameter: double IP_Samplerate

Direction: Down

Valid for sensor: all

Valid values:

Minimum: 0

Unit: Hz

Description: Tells the driver the samplerate of the sensor.

IP_Samplerate

Parameter: double IP_Datarate

Direction: Down

Valid for sensor: all

Valid values:

Minimum: 0

Unit: Hz

Description: Tells the driver the datarate (output rate) of the sensor. It is used for timeout check. If it is zero, no timeout check is performed.

IP_Datarate

Parameter: double IP_Range

Direction: Down

Valid for sensor:

SENSOR_ILD1302

SENSOR_ILD1401

SENSOR_ILD1402

SENSOR_ILD1700

SENSOR_ILD1800

SENSOR_ILD2000

SENSOR_ILD2200

SENSOR_IFD2400

SENSOR_IFD2401

SENSOR_IFD2430

SENSOR_IFD2431

SENSOR_DT6100

Valid values:

Minimum: 0

Unit: mm (for SENSOR_ILD...), μm (for SENSOR_IFD...), either mm or μm or any other (for SENSOR_DT6100)

Description: Tells the driver the range of sensor. It is used to scale the raw sensor values into mm or μm . If it is zero, no scaling is done.

IP_Range

Parameter: int IP_ASCII

IP_ASCII

Direction: Down

Valid for sensor:

SENSOR_ILD1302
 SENSOR_ILD1402
 SENSOR_ILD1700
 SENSOR_IFD2400
 SENSOR_IFD2401
 SENSOR_IFD2430
 SENSOR_IFD2431
 SENSOR_ODC2600

Valid values:

0 = Binary
 1 = ASCII

Description: Tells the driver if the sensor is sending data in ASCII or binary format. It is used to at conversion of data bytes into values.

A.3.2 SettingsChanged

Checks if sensor or driver settings have changed since last call of Get_DrvSetting.

Parameter: int IA_SettingsChanged

IA_SettingsChanged

Direction: Up

Valid values:

0 = FALSE
 1 = TRUE

Description: TRUE when settings have changed.

A.3.3 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

The following parameters returned, are already described at chapter [A.1](#) (but instead of IP_... for interface parameter is called now IA_... for interface answer)

: IA_ScaleErrorValues and IA_RingBufferSize.

They are returned for each sensor and interface.

The other return values does result from sensor parameters (used by the sensor interface part of MEDAQLib).

Parameters only valid for a specific sensor are not described here but at command Get_DrvSetting of specific sensor chapter.

Parameter: int IA_CheckRingBufferOverflow

IA_CheckRingBufferOverflow

Direction: Up

Valid for sensor: all

Valid values:

0 = off
 1 = on

Description: If it is set to 1, the ring buffer is monitored. If there is an overflow, an error is returned by GetError function.

A Parameters

Parameter: int IA_DataOn

IA_DataOn

Direction: Up

Valid for sensor: all

Valid values:

0 = off

1 = on

Description: Specifies if the driver expects data from sensor. If not no timeout check is performed. If data (values, no answer) from sensor is received it is automatically turned on.

Parameter: double IA_Samplerate

IA_Samplerate

Direction: Up

Valid for sensor: all

Unit: Hz

Description: The samplerate of the sensor used by driver. 0 means the samplerate is not known.

Parameter: double IA_Datarate

IA_Datarate

Direction: Up

Valid for sensor: all

Unit: Hz

Description: The datarate (output rate) of the sensor used by driver to check timeout from sensor. 0 means the datarate is not known.

Parameter: double IA_Range

IA_Range

Direction: Up

Valid for sensor:

SENSOR_ILD1302

SENSOR_ILD1401

CONTROLLER_ILD1401

SENSOR_ILD1402

SENSOR_ILD1700

SENSOR_ILD1800

SENSOR_ILD2000

SENSOR_ILD2200

SENSOR_IFD2400

SENSOR_IFD2401

SENSOR_IFD2430

SENSOR_IFD2431

SENSOR_ODC2500

SENSOR_ODC2600

SENSOR_DT6100

ENCODER_IF2004

Unit: mm (for SENSOR_ILD...), μm (for SENSOR_IFD...), either mm or μm or any other (for SENSOR_DT6100 or ENCODER_IF2004)

Description: The range of sensor. It is used to scale the raw sensor values. If it is zero, no scaling is done.
For ENCODER_IF2004, it is the distance per count of the encoder used by the driver for scaling data.

Parameter: int IA_ASCII

IA_ASCII

Direction: Up

Valid for sensor:

SENSOR_ILD1302
 SENSOR_ILD1402
 SENSOR_ILD1700
 SENSOR_IFD2400
 SENSOR_IFD2401
 SENSOR_IFD2430
 SENSOR_IFD2401
 SENSOR_ODC2600

Valid values:

0 = Binary
 1 = ASCII

Description: Returns the conversion mode which is used by the driver to convert data bytes into values.

A.3.4 Cmd_Generic

With Cmd_Generic, any data can be sent to the sensor.

Parameter: String SP_CmdStr

SP_CmdStr

Direction: Down

Description: The command string as it is sent to the sensor. E.g. for ILD sensors it always starts with "+++\rILD1" ('\r' is carriage return, 0x0d).

Parameter: int CP_SensorAnswerTimeout

CP_SensorAnswerTimeout

Direction: Down

Valid values:

Minimum: 0
Maximum: INT_MAX (2147483647)

Unit: ms

Default: 500

Description: The timeout waiting for the complete answer from sensor.

Parameter: String SA_CompleteAnswer

SA_CompleteAnswer

Direction: Up

Description: The raw (not interpreted) answer (to a command) from the sensor is stored here.

A.3.5 Clear_Buffers

This command is executed by the driver and does not affect the sensor. It erases the ring buffer and the input and output buffer of the interface.

Parameter: int SP_AllDevices

SP_AllDevices

Direction: Down

Valid values:

0 = FALSE
 1 = TRUE

Default: 0

Description: If it is set to 1, not only this instance but all instances created from this driver are cleared. It can be used to synchronize data acquisition from several sensors because after clearing all buffers the next data from all sensors have same timestamp.

A.3.6 DataAvail_Event

This command registers an event which will be set when new data is available. The event can be used to wait for new data in your application.

Parameter: int IP_EventOnAvailableValues

Direction: Down

IP_EventOnAvailableValues

Valid values:

Minimum: -1

Maximum: INT_MAX (2147483647)

Default: -1

Description: If it is set to -1, the event is not set by this condition.

Otherwise the event is set if at least so many values are available as specified here.

Parameter: double IP_EventOnBufferFillsize

IP_EventOnBufferFillsize

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Default: 1.0

Description: If it is set to 1.0, the event is not set by this condition.

Otherwise the event is set if the ring buffer fill size is at least as high as specified here.

For setting the ring buffer size please refer to chapter [A.1.1](#).

Parameter: DWORD IA_DataAvailEvent

IA_DataAvailEvent

Direction: Up

Description: The event handle for data avail.

If IP_EventOnAvailableValues is -1 (default) and IP_EventOnBufferFillsize is 1.0 (default), the event is released and the value is 0.

Example how to use the event:

```
SetParameterString (instance, "S_Command", "DataAvail_Event");
SetParameterInt (instance, "IP_EventOnAvailableValues", 1024);
DWORD err= SensorCommand (instance);
/* error handling, if err!=ERR_SUCCESS */
HANDLE event= NULL;
GetParameterDWORD (instance, "IA_DataAvailEvent", &event);

while (true)
{
    /* timeout is 1000 ms */
    if (WaitForSingleObject (event, 1000)==WAIT_OBJECT_0)
        err= TransferData (instance, raw, scaled, 1024, &read);
    /* error handling, verify that read==1024, process data */
}
```

B Commands for ILR110x_115x

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

B.1 Get_Parameters (GAP)

Retrieve all parameters from the sensor.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Firmware version.

Parameter: int SA_PilotLaser

SA_PilotLaser

Direction: Up

Valid values:

0= off

1= on

Description: Pilot laser behaviour.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

4800

9600

19200

38400

57600

Description: Sensor baudrate.

Parameter: int SA_SensorDatabits

SA_SensorDatabits

Direction: Up

Valid values:

7

8

Description: Sensor data bits.

Parameter: int SA_SensorStopbits

SA_SensorStopbits

Direction: Up

Valid values:

1

2

Description: Sensor stop bits.

Parameter: int SA_ContinuousMode Direction: Up Valid values: 0= continuous 1= single Description: Sensor is sending data continuous or only single values.	SA_ContinuousMode
Parameter: int SA_Q1Value Direction: Up Valid values: 0= low 1= high Description: The actual state of output Q1.	SA_Q1Value
Parameter: int SA_ModeQ1 Direction: Up Valid values: 0= not active 1= switching point 2= switching points Description: The mode of output Q1.	SA_ModeQ1
Parameter: int SA_LimitQ1-1 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 1 of output Q1.	SA_LimitQ1-1
Parameter: int SA_LimitQ1-2 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 2 of output Q1.	SA_LimitQ1-2
Parameter: int HysteresisQ1 Direction: Up Valid values: Minimum: 0 Maximum: 999 Description: The hysteresis of output Q1.	HysteresisQ1
Parameter: int SA_NormQ1 Direction: Up Valid values: 0= normal 1= inverted Description: Specifies if output Q1 is normal or inverted.	SA_NormQ1
Parameter: int SA_Q2Value Direction: Up Valid values: 0= low 1= high Description: The actual state of output Q2.	SA_Q2Value

Parameter: int SA_ModeQ2 Direction: Up Valid values: 0= not active 1= switching point 2= switching points Description: The mode of output Q2.	SA_ModeQ2
Parameter: int SA_LimitQ2-1 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 1 of output Q2.	SA_LimitQ2-1
Parameter: int SA_LimitQ2-2 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 2 of output Q2.	SA_LimitQ2-2
Parameter: int HysteresisQ2 Direction: Up Valid values: Minimum: 0 Maximum: 999 Description: The hysteresis of output Q2.	HysteresisQ2
Parameter: int SA_NormQ2 Direction: Up Valid values: 0= normal 1= inverted Description: Specifies if output Q2 is normal or inverted.	SA_NormQ2
Parameter: int SA_AnalogValue Direction: Up Valid values: Minimum: 0 Maximum: 4095 Description: The actual value of analog output.	SA_AnalogValue
Parameter: int SA_LimitQA-1 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 1 of analog output.	SA_LimitQA-1
Parameter: int SA_LimitQA-2 Direction: Up Valid values: Minimum: -48000 Maximum: 96000 Description: The limit 2 of analog output.	SA_LimitQA-2

Parameter: int SA_NormQA Direction: Up Valid values: 0 = normal 1 = inverted Description: Specifies if analog output is normal or inverted.	SA_NormQA
Parameter: int SA_OutputFormat Direction: Up Valid values: 0 = mm 1 = inch*100 Description: Output format of measured values.	SA_OutputFormat
Parameter: int SA_Offset Direction: Up Valid values: Minimum: -48000 Maximum: 48000 Description: Offset value.	SA_Offset
Parameter: int SA_Password Direction: Up Valid values: 0 = disabled 1 = enabled Description: Password protection for sensor menu.	SA_Password
Parameter: int SA_ErrorStatus Direction: Up Valid values: Minimum: 0x00 Maximum: 0xff Description: Error status as 8 bit field. 00000000: no error. 00000010: PLL UNLOCKED - Counter error. 00000100: LOW VOLT - Error in supply voltage: voltage too low (or error in measurement of supply voltage). 00101000: OVERTEMP - Temperature too high (above 85 °C inside); Measurement switched off. 00010000: Dist (mm) >Maximum - No target in range or sensor badly aligne. 00100000: Temperature warning (below -10 °C or above 70 °C). 01000000: BLINDING - External light too strong or internal error. 10000000: LAS.ERR. - Measurement laser faulty.	SA_ErrorStatus

B.2 Get_Energy (GDB)

Get the amount of receiving by sensor.

Parameter: int SA_Energy Direction: Up Unit: dB Valid values: Minimum: -120 Maximum: 0 Description: Energy value.	SA_Energy
--	-----------

B.3 Get_SerialNbr (GNR)

Get the serial number of sensor.

Parameter: String SA_SerialNumber
Direction: Up
Description: Serial number.

SA_SerialNumber

B.4 Get_ErrorStatus (GSI)

Get error status from the sensor.

Parameter: int SA_ErrorStatus
Direction: Up
Valid values:
 Minimum: 0x00
 Maximum: 0xff

SA_ErrorStatus

Description: Error status as 8 bit field.
00000000: no error.
00000010: PLL UNLOCKED - Counter error.
00000100: LOW VOLT - Error in supply voltage: voltage too low (or error in measurement of supply voltage).
00101000: OVERTEMP - Temperature too high (above 85 °C inside); Measurement switched off.
00010000: Dist (mm) >Maximum - No target in range or sensor badly aligned.
00100000: Temperature warning (below -10 °C or above 70 °C).
01000000: BLINDING - External light too strong or internal error.
10000000: LAS.ERR. - Measurement laser faulty.

B.5 Get_Temperature (GTE)

Retrieve the temperature inside of the sensor.

Parameter: int SA_Temperature
Direction: Up
Unit: °C
Description: Sensor temperature.

SA_Temperature

B.6 Get_Version (GVE)

Get the version of sensor firmware.

Parameter: String SA_Version
Direction: Up
Description: Firmware version.

SA_Version

B.7 Set_Offset (IDO)

Set the offset which is added by the sensor to distance values.

Parameter: int SP_Offset

SP_Offset

Direction: Down

Valid values:

Minimum: -12000 [mm] or -48000 [100*inch]

Maximum: 12000 [mm] or 48000 [100*inch]

Description: Offset value.

B.8 Set_VisibleLaser (IVL)

Set the behaviour of the pilot laser of the sensor.

Parameter: int SP_PilotLaser

SP_PilotLaser

Direction: Down

Valid values:

0= off

1= on

Description: Pilot laser behaviour.

B.9 Set_HysteresisQ1 (IH1)

Hysteresis setting around the switching point Q1 in [mm] or [100*inch].

Parameter: int SP_HysteresisQ1

SP_HysteresisQ1

Direction: Down

Valid values:

Minimum: 0 [mm] / [100*inch]

Maximum: 254 [mm] or 999 [100*inch]

Description: Hysteresis Q1.

B.10 Set_HysteresisQ2 (IH2)

Hysteresis setting around the switching point Q2 in [mm] or [100*inch].

Parameter: int SP_HysteresisQ2

SP_HysteresisQ2

Direction: Down

Valid values:

Minimum: 0 [mm] / [100*inch]

Maximum: 254 [mm] or 999 [100*inch]

Description: Hysteresis Q2.

B.11 Set_LimitQ1-1 (IL1)

Setting of the first switch point of Q1 in [mm] or [100*inch].

Parameter: int SP_LimitQ1-1

SP_LimitQ1-1

Direction: Down

Valid values:

Minimum: Offset [mm] / [100*inch]

Maximum: 12000+Offset [mm] or 48000+Offset [100*inch]

Description: Limit Q1-1.

B.12 Set_LimitQ2-1 (IL2)

Setting of the first switch point of Q2 in [mm] or [100*inch].

Parameter: int SP_LimitQ2-1

SP_LimitQ2-1

Direction: Down

Valid values:

Minimum: Offset [mm] / [100*inch]

Maximum: 12000+Offset [mm] or 48000+Offset [100*inch]

Description: Limit Q2-1.

B.13 Set_LimitQA-1 (IL1)

Setting of the 0% point of the analog characteristic. Only valid for sensors ILR1100 and ILR1150 (with analog output).

Parameter: int SP_LimitQA-1

SP_LimitQA-1

Direction: Down

Valid values:

Minimum: Offset

Maximum: 12000+Offset

Description: Limit QA-1.

B.14 Set_LimitQ1-2 (IL4)

Setting of the second switch point of Q1 in [mm] or [100*inch].

Parameter: int SP_LimitQ1-2

SP_LimitQ1-2

Direction: Down

Valid values:

Minimum: Offset [mm] / [100*inch]

Maximum: 12000+Offset [mm] or 48000+Offset [100*inch]

Description: Limit Q1-2.

B.15 Set_LimitQ2-2 (IL2)

Setting of the second switch point of Q2 in [mm] or [100*inch].

Parameter: int SP_LimitQ2-2

SP_LimitQ2-2

Direction: Down

Valid values:

Minimum: Offset [mm] / [100*inch]

Maximum: 12000+Offset [mm] or 48000+Offset [100*inch]

Description: Limit Q2-2.

B.16 Set_LimitQA-2 (IL6)

Setting of the 100% point of the analog characteristic. Only valid for sensors ILR1100 and ILR1150 (with analog output).

Parameter: int SP_LimitQA-2

SP_LimitQA-2

Direction: Down

Valid values:

Minimum: Offset

Maximum: 12000+Offset

Description: Limit QA-2.

B.17 Set_ModeQ1 (IM1)

Set the mode of output Q1.

Parameter: int SP_ModeQ1

SP_ModeQ1

Direction: Down

Valid values:

0= not active

1= switching point

2= switching points

Description: The mode of output Q1.

B.18 Set_ModeQ2 (IM2)

Set the mode of output Q2.

Parameter: int SP_ModeQ2

SP_ModeQ2

Direction: Down

Valid values:

0= not active

1= switching point

2= switching points

Description: The mode of output Q2.

B.19 Set_NormQA (INA)

Set the norm of analog output. Only valid for sensors ILR1100 and ILR1150 (with analog output).

Parameter: int SP_NormQA

SP_NormQA

Direction: Down

Valid values:

0 = normal

1 = inverted

Description: Specifies if analog output is normal or inverted.

B.20 Set_NormQ1 (IN1)

Set the norm of output Q1.

Parameter: int SP_NormQ1

SP_NormQ1

Direction: Down

Valid values:

0 = normal

1 = inverted

Description: Specifies if output Q1 is normal or inverted.

B.21 Set_NormQ2 (IN2)

Set the norm of output Q2.

Parameter: int SP_NormQ2

SP_NormQ2

Direction: Down

Valid values:

0 = normal

1 = inverted

Description: Specifies if output Q2 is normal or inverted.

B.22 Set_ContinuousMode (ICM)

Set the measurement mode.

Parameter: int SP_ContinuousMode

SP_ContinuousMode

Direction: Down

Valid values:

0 = continuous

1 = single

Description: Sensor is sending data continuous or only single values.

B.23 Exec_ContMeasure (ECM)

Continuous measurement output is set and triggered by the next request for measured values (ESM).

B.24 Trg_SingleMeasure (ESM)

Request for measured value with single measurement output.

B.25 Set_Stand-by (ISB)

Set the sensor in stand-by mode or reactivates it.

Parameter: int SP_Stand-by

SP_Stand-by

Direction: Down

Valid values:

0 = operation

1 = stand-by

Description: Sensor stand-by mode.

B.26 Save_Parameters (EPW)

Store all actual parameters in sensor memory.

C Commands for ILR118x

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

C.1 DistanceTracking (DT)

Start distance tracking mode.

C.2 DistanceTracking7m (DS)

Start distance tracking (7 m) mode.

C.3 DistanceTracking10Hz (DW)

Start distance tracking (10 Hz) mode. Only valid for ILR1181.

C.4 DistanceTracking50Hz (DX)

Start distance tracking (10 Hz) mode. Only valid for ILR1182.

C.5 DistanceTriggered (DF)

Start distance tracking (with external trigger) mode.

C.6 DistanceMeasure (DM)

Measure one distance value.

C.7 StopTracking (<ESC>)

Stop any tracking mode.

C.8 Set_OutputFormat (SD)

Set the output format how values are sent from sensor.

Parameter: int SP_OutputFormat

SP_OutputFormat

Direction: Down

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

Mode binary is currently not supported by MEDAQLib. So if it is selected, no values can be read.

C.9 Get_OutputFormat (SD)

Get the output format how values are sent from sensor.

Parameter: int SA_OutputFormat

SA_OutputFormat

Direction: Up

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

C.10 Set_ScaleFactor (SF)

Set the scaling factor how the sensor scale distance values.

Parameter: double SP_ScaleFactor

SP_ScaleFactor

Direction: Down

Valid values:

Minimum: -1000.0

Maximum: 1000.0

Description: Scaling factor.

C.11 Get_ScaleFactor (SF)

Get the scaling factor how the sensor scale distance values.

Parameter: double SA_ScaleFactor

SA_ScaleFactor

Direction: Up

Valid values:

Minimum: -1000.0

Maximum: 1000.0

Description: Scaling factor.

C.12 Set_Offset (OF)

Set the offset which is added by the sensor to distance values.

Parameter: double SP_Offset

SP_Offset

Direction: Down

Valid values:

Minimum: -150000.0

Maximum: 150000.0

Description: Offset value.

C.13 Get_Offset (OF)

Get the offset which is added by the sensor to distance values.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -150000.0

Maximum: 150000.0

Description: Offset value.

C.14 CurrentDistAsOffset (SO)

Set the current distance value as offset.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -150000.0

Maximum: 150000.0

Description: Offset value which is set.

C.15 Set_RemovalMeasVal (RM)

Set how invalid measure values should be treated by the sensor.

Parameter: int SP_PrecedingValues

SP_PrecedingValues

Direction: Down

Valid values:

Minimum: 1

Maximum: 10

Description: Designates the number of preceding measuring values that will be evaluated in the case of non-conforming measurement.

Parameter: double SP_ValidRange

SP_ValidRange

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Defines the range of permissible values. If this range is exceeded in negative or positive direction, the respective measuring value will be corrected accordingly.

Parameter: int SP_InvalidValues

SP_InvalidValues

Direction: Down

Valid values:

Minimum: 0

Maximum: 100

Description: Stands for the number of values that are out of the permissible value range; in the event of out-of-tolerance values arriving in succession, the most recently corrected value will be included in the correction process for the next out-of-tolerance value.

C.16 Get_RemovalMeasVal (RM)

Get how invalid measure values should be treated by the sensor.

Parameter: int SA_PrecedingValues

SA_PrecedingValues

Direction: Up

Valid values:

Minimum: 1

Maximum: 10

Description: Designates the number of preceding measuring values that will be evaluated in the case of non-conforming measurement.

Parameter: int SA_ValidRange

SA_ValidRange

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Defines the range of permissible values. If this range is exceeded in negative or positive direction, the respective measuring value will be corrected accordingly.

Parameter: int SA_InvalidValues

SA_InvalidValues

Direction: Up

Valid values:

Minimum: 0

Maximum: 100

Description: Stands for the number of values that are out of the permissible value range; in the event of out-of-tolerance values arriving in succession, the most recently corrected value will be included in the correction process for the next out-of-tolerance value.

C.17 Set_ErrorMode (SE)

Set the behaviour of digital and analog outputs in case of an error.

Parameter: int SP_ErrorMode

SP_ErrorMode

Direction: Down

Valid values:

0= last valid value

1= switch to bounds

2= switch to negated bounds

Description: Error mode.

C.18 Get_ErrorMode (SE)

Get the behaviour of digital and analog outputs in case of an error.

Parameter: int SA_ErrorMode

SA_ErrorMode

Direction: Up

Valid values:

0= last valid value

1= switch to bounds

2= switch to negated bounds

Description: Error mode.

C.19 Set_AlarmStart (AC)

Sets the beginning of the distance range, for which the switching output will be turned active.

Parameter: double SP_AlarmStart

SP_AlarmStart

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm start.

C.20 Get_AlarmStart (AC)

Gets the beginning of the distance range, for which the switching output will be turned active.

Parameter: double SA_AlarmStart

SA_AlarmStart

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm start.

C.21 Set_AlarmHysteresis (AH)

Set the switching hysteresis at the beginning and the end point of the active range of the switching output.

Parameter: double SP_AlarmHysteresis

SP_AlarmHysteresis

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm hysteresis.

C.22 Get_AlarmHysteresis (AH)

Get the switching hysteresis at the beginning and the end point of the active range of the switching output.

Parameter: double SA_AlarmHysteresis

SA_AlarmHysteresis

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm hysteresis.

C.23 Set_AlarmWidth (AW)

Set the length of the active range for the switching output.

Parameter: double SP_AlarmWidth

SP_AlarmWidth

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm width.

C.24 Get_AlarmWidth (AW)

Get the length of the active range for the switching output.

Parameter: double SA_AlarmWidth

SA_AlarmWidth

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Alarm width.

C.25 Set_RangeBegin4mA (RB)

Set the starting point of the distance range that is provided at the analog output.

Parameter: double SP_RangeBegin

SP_RangeBegin

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Range begin.

C.26 Get_RangeBegin4mA (RB)

Get the starting point of the distance range that is provided at the analog output.

Parameter: double SA_RangeBegin

SA_RangeBegin

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Range begin.

C.27 Set_RangeEnd20mA (RE)

Set the end point of the distance range that is provided at the analog output.

Parameter: double SP_RangeEnd

SP_RangeEnd

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Range end.

C.28 Get_RangeEnd20mA (RE)

Get the end point of the distance range that is provided at the analog output.

Parameter: double SA_RangeEnd

SA_RangeEnd

Direction: Up

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Range end.

C.29 Set_AverageValue (SA)

Set the average value for floating averaging.

Parameter: int SP_Average

SP_Average

Direction: Down

Valid values:

Minimum: 1

Maximum: 20

Description: Average value.

C.30 Get_AverageValue (SA)

Get the average value for floating averaging.

Parameter: int SA_Average

SA_Average

Direction: Up

Valid values:

Minimum: 1

Maximum: 20

Description: Average value.

C.31 Set_MeasureTime (ST)

Set an index for measure time of one distance value.

Parameter: int SP_MeasureTime

SP_MeasureTime

Direction: Down

Valid values:

Minimum: 0

Maximum: 25

Description: Measure time index.

C.32 Get_MeasureTime (ST)

Get an index for measure time of one distance value.

Parameter: int SA_MeasureTime

SA_MeasureTime

Direction: Up

Valid values:

Minimum: 0

Maximum: 25

Description: Measure time index.

C.33 Set_TriggerDelay (TD)

Set the behaviour of the trigger input.

Parameter: int SP_TriggerDelay

SP_TriggerDelay

Direction: Down

Unit: ms

Valid values:

Minimum: 0

Maximum: 9999

Description: Trigger delay.

Parameter: int SP_TriggerEdge

SP_TriggerEdge

Direction: Down

Valid values:

0= falling

1= rising

Description: Trigger edge.

C.34 Get_TriggerDelay (TD)

Get the behaviour of the trigger input.

Parameter: int SA_TriggerDelay

SA_TriggerDelay

Direction: Up

Unit: ms

Valid values:

Minimum: 0

Maximum: 9999

Description: Trigger delay.

Parameter: int SA_TriggerEdge

SA_TriggerEdge

Direction: Up

Valid values:

0= falling

1= rising

Description: Trigger edge.

C.35 Set_TriggerMode (TM)

Set parameters for the auto-start trigger function which allows external triggering of the auto-start command that was set via parameter AS.

Parameter: int SP_AutostartTrigger

SP_AutostartTrigger

Direction: Down

Valid values:

0= off

1= on

Description: Autostart trigger.

Parameter: int SP_AutostartEdge

SP_AutostartEdge

Direction: Down

Valid values:

0= falling

1= rising

Description: Autostart trigger edge.

C.36 Get_TriggerMode (TM)

Get parameters for the auto-start trigger function which allows external triggering of the auto-start command that was set via parameter AS.

Parameter: int SA_AutostartTrigger

SA_AutostartTrigger

Direction: Up

Valid values:

0= off

1= on

Description: Autostart trigger.

Parameter: int SA_AutostartEdge

SA_AutostartEdge

Direction: Up

Valid values:

0= falling

1= rising

Description: Autostart trigger edge.

C.37 Set_Baudrate (BR)

Set the baudrate of the sensors serial interface.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled first (sensor command StopTracking).

Parameter: int SP_SensorBaudrate

SP_SensorBaudrate

Direction: Down

Valid values:

2400

4800

9600

19200

38400

Description: Sensor baudrate.

C.38 Get_Baudrate (BR)

Get the baudrate of the sensors serial interface.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

2400

4800

9600

19200

38400

Description: Sensor baudrate.

C.39 Set_Autostart (AS)

Set which function will be carried out when power becomes available to the sensor.

Parameter: int SP_AutostartCommand

SP_AutostartCommand

Direction: Down

Valid values:

0= DT

1= DS

2= DW

3= DX

4= DF

5= DM

6= TP

7= ID

Description: Autostart command.

C.40 Get_Autostart (AS)

Get which function will be carried out when power becomes available to the sensor.

Parameter: int SA_AutostartCommand

SA_AutostartCommand

Direction: Up

Valid values:

0= DT

1= DS

2= DW

3= DX

4= DF

5= DM

6= TP

7= ID

-1= unknown

Description: Autostart command.

C.41 Get_Info (ID)

Retrieve information (like serial number) of the sensor.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Firmware version.

C.42 Get_AllParameters (PA)

Retrieve all parameters from the sensor.

Parameter: int SA_Average

SA_Average

Direction: Up

Valid values:

Minimum: 1

Maximum: 20

Description: Average value.

Parameter: int SA_OutputFormat

SA_OutputFormat

Direction: Up

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

Parameter: int SA_MeasureTime

SA_MeasureTime

Direction: Up

Valid values:

Minimum: 0

Maximum: 25

Description: Measure time index.

Parameter: double SA_ScaleFactor Direction: Up Valid values: Minimum: -1000.0 Maximum: 1000.0 Description: Scaling factor.	SA_ScaleFactor
Parameter: int SA_ErrorMode Direction: Up Valid values: 0= last valid value 1= switch to bounds 2= switch to negated bounds Description: Error mode.	SA_ErrorMode
Parameter: double SA_AlarmStart Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm start.	SA_AlarmStart
Parameter: double SP_AlarmHysteresis Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm hysteresis.	SP_AlarmHysteresis
Parameter: double SA_AlarmWidth Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm width.	SA_AlarmWidth
Parameter: double SA_RangeBegin Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Range begin.	SA_RangeBegin
Parameter: double SA_RangeEnd Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Range end.	SA_RangeEnd
Parameter: int SA_PrecedingValues Direction: Up Valid values: Minimum: 1 Maximum: 10 Description: Designates the number of preceding measuring values that will be evaluated in the case of non-conforming measurement.	SA_PrecedingValues

Parameter: int SA_ValidRange Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Defines the range of permissible values. If this range is exceeded in negative or positive direction, the respective measuring value will be corrected accordingly.	SA_ValidRange
Parameter: int SA_InvalidValues Direction: Up Valid values: Minimum: 0 Maximum: 100 Description: Stands for the number of values that are out of the permissible value range; in the event of out-of-tolerance values arriving in succession, the most recently corrected value will be included in the correction process for the next out-of-tolerance value.	SA_InvalidValues
Parameter: int SA_TriggerDelay Direction: Up Unit: ms Valid values: Minimum: 0 Maximum: 9999 Description: Trigger delay.	SA_TriggerDelay
Parameter: int SA_TriggerEdge Direction: Up Valid values: 0= falling 1= rising Description: Trigger edge.	SA_TriggerEdge
Parameter: int SA_AutostartTrigger Direction: Up Valid values: 0= off 1= on Description: Autostart trigger.	SA_AutostartTrigger
Parameter: int SA_AutostartEdge Direction: Up Valid values: 0= falling 1= rising Description: Autostart trigger edge.	SA_AutostartEdge
Parameter: int SA_SensorBaudrate Direction: Up Valid values: 2400 4800 9600 19200 38400 Description: Sensor baudrate.	SA_SensorBaudrate

Parameter: int SA_AutostartCommand

SA_AutostartCommand

Direction: Up

Valid values:

0= DT

1= DS

2= DW

3= DX

4= DF

5= DM

6= TP

7= ID

-1= unknown

Description: Autostart command.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -150000.0

Maximum: 150000.0

Description: Offset value which is set.

C.43 Get_Temperature (TP)

Retrieve the temperature inside of the sensor.

Parameter: double SA_Temperature

SA_Temperature

Direction: Up

Unit: °C

Description: Sensor temperature.

C.44 Reset_Parameters (PR)

Reset all parameters of sensor to factory defaults and return new parameters.

Parameter: int SA_Average

SA_Average

Direction: Up

Valid values:

Minimum: 1

Maximum: 20

Description: Average value.

Parameter: int SA_OutputFormat

SA_OutputFormat

Direction: Up

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

Parameter: int SA_MeasureTime Direction: Up Valid values: Minimum: 0 Maximum: 25 Description: Measure time index.	SA_MeasureTime
Parameter: double SA_ScaleFactor Direction: Up Valid values: Minimum: -1000.0 Maximum: 1000.0 Description: Scaling factor.	SA_ScaleFactor
Parameter: int SA_ErrorMode Direction: Up Valid values: 0= last valid value 1= switch to bounds 2= switch to negated bounds Description: Error mode.	SA_ErrorMode
Parameter: double SA_AlarmStart Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm start.	SA_AlarmStart
Parameter: double SP_AlarmHysteresis Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm hysteresis.	SP_AlarmHysteresis
Parameter: double SA_AlarmWidth Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Alarm width.	SA_AlarmWidth
Parameter: double SA_RangeBegin Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Range begin.	SA_RangeBegin
Parameter: double SA_RangeEnd Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Range end.	SA_RangeEnd

Parameter: int SA_PrecedingValues Direction: Up Valid values: Minimum: 1 Maximum: 10 Description: Designates the number of preceding measuring values that will be evaluated in the case of non-conforming measurement.	SA_PrecedingValues
Parameter: int SA_ValidRange Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Defines the range of permissible values. If this range is exceeded in negative or positive direction, the respective measuring value will be corrected accordingly.	SA_ValidRange
Parameter: int SA_InvalidValues Direction: Up Valid values: Minimum: 0 Maximum: 100 Description: Stands for the number of values that are out of the permissible value range; in the event of out-of-tolerance values arriving in succession, the most recently corrected value will be included in the correction process for the next out-of-tolerance value.	SA_InvalidValues
Parameter: int SA_TriggerDelay Direction: Up Unit: ms Valid values: Minimum: 0 Maximum: 9999 Description: Trigger delay.	SA_TriggerDelay
Parameter: int SA_TriggerEdge Direction: Up Valid values: 0 = falling 1 = rising Description: Trigger edge.	SA_TriggerEdge
Parameter: int SA_AutostartTrigger Direction: Up Valid values: 0 = off 1 = on Description: Autostart trigger.	SA_AutostartTrigger
Parameter: int SA_AutostartEdge Direction: Up Valid values: 0 = falling 1 = rising Description: Autostart trigger edge.	SA_AutostartEdge

Parameter: int SA_SensorBaudrate Direction: Up Valid values: 2400 4800 9600 19200 38400 Description: Sensor baudrate.	SA_SensorBaudrate
Parameter: int SA_AutostartCommand Direction: Up Valid values: 0= DT 1= DS 2= DW 3= DX 4= DF 5= DM 6= TP 7= ID -1= unknown Description: Autostart command.	SA_AutostartCommand
Parameter: double SA_Offset Direction: Up Valid values: Minimum: -150000.0 Maximum: 150000.0 Description: Offset value which is set.	SA_Offset

D Commands for ILR1191

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

D.1 DistanceTracking (DT)

Start distance tracking mode.

D.2 DistanceTriggered (DF)

Start distance tracking (with external trigger) mode.

D.3 DistanceMeasure (DM)

Measure one distance value.

D.4 SpeedMeasure (VM)

Measure one speed (velocity) value.

Attention! To leave that mode with StopTracking, there is a delay of average value (SA) / measure frequency (MF) seconds.

For example, if average value is 20 and measure frequency is 10 Hz, it takes 2 seconds to leave this mode. In extreme case, this could be 30000/1 seconds, more than 8 hours.

D.5 SpeedTracking (VT)

Start speed (velocity) tracking mode.

D.6 StopTracking (<ESC>)

Stop any tracking mode.

D.7 Set_OutputFormat (SD)

Set the output format how values are sent from sensor.

Parameter: int SP_OutputFormat

SP_OutputFormat

Direction: Down

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

Parameter: int SP_OutputContent

SP_OutputContent

Direction: Down

Valid values:

0= Measuring value

1= Measuring value, signal strength

2= Measuring value, sensor temperature

3= Measuring value, signal strength, sensor temperature

Description: Set which data is transmitted by sensor.

D.8 Get_OutputFormat (SD)

Get the output format how values are sent from sensor.

Parameter: int SA_OutputFormat

SA_OutputFormat

Direction: Up

Valid values:

0= decimal

1= hex

2= binary

Description: Output format of values.

Parameter: int SA_OutputContent

SA_OutputContent

Direction: Up

Valid values:

0= Measuring value

1= Measuring value, signal strength

2= Measuring value, sensor temperature

3= Measuring value, signal strength, sensor temperature

Description: Get which data is transmitted by sensor.

D.9 Set_TerminatingChar (TE)

Set the termination character of each measurement.

Parameter: int SP_TerminationChar

SP_TerminationChar

Direction: Down

Valid values:

0= <CRLF>

- 1 = <CR>
- 2 = <LF>
- 3 = <STX>
- 4 = <ETX>
- 5 = Tabulator
- 6 = Space
- 7 = Comma
- 8 = Colon
- 9 = Semicolon

Description: Termination character.

D.10 Get_TerminatingChar (TE)

Get the termination character of each measurement.

Parameter: int SA_TerminationChar

SA_TerminationChar

Direction: Up

Valid values:

- 0 = <CRLF>
- 1 = <CR>
- 2 = <LF>
- 3 = <STX>
- 4 = <ETX>
- 5 = Tabulator
- 6 = Space
- 7 = Comma
- 8 = Colon
- 9 = Semicolon

Description: Termination character.

D.11 Set_FormatSSI (SC)

Set the transmission format of SSI output.

Parameter: int SP_SSIFormat

SP_SSIFormat

Direction: Down

Valid values:

- 0 = binary
- 1 = grey code

Description: SSI transmission format.

D.12 Get_FormatSSI (SC)

Get the transmission format of SSI output.

Parameter: int SA_SSIFormat

SA_SSIFormat

Direction: Up

Valid values:

- 0 = binary
- 1 = grey code

Description: SSI transmission format.

D.13 Set_ScaleFactor (SF)

Set the scaling factor how the sensor scale distance values.

Parameter: double SP_ScaleFactor

SP_ScaleFactor

Direction: Down

Valid values:

Minimum: -10.0

Maximum: 10.0

Description: Scaling factor.

D.14 Get_ScaleFactor (SF)

Get the scaling factor how the sensor scale distance values.

Parameter: double SA_ScaleFactor

SA_ScaleFactor

Direction: Up

Valid values:

Minimum: -10.0

Maximum: 10.0

Description: Scaling factor.

D.15 Set_Offset (OF)

Set the offset which is added by the sensor to distance values.

Parameter: double SP_Offset

SP_Offset

Direction: Down

Valid values:

Minimum: -5000.0

Maximum: 5000.0

Description: Offset value.

D.16 Get_Offset (OF)

Get the offset which is added by the sensor to distance values.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -5000.0

Maximum: 5000.0

Description: Offset value.

D.17 CurrentDistAsOffset (SO)

Set the current distance value as offset.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -5000.0

Maximum: 5000.0

Description: Offset value which is set.

D.18 Set_ErrorMode (SE)

Set the behaviour of digital and analog outputs in case of an error.

Parameter: int SP_ErrorMode

SP_ErrorMode

Direction: Down

Valid values:

0 = last valid value

1 = switch to bounds

2 = switch to negated bounds

Description: Error mode.

D.19 Get_ErrorMode (SE)

Get the behaviour of digital and analog outputs in case of an error.

Parameter: int SA_ErrorMode

SA_ErrorMode

Direction: Up

Valid values:

0 = last valid value

1 = switch to bounds

2 = switch to negated bounds

Description: Error mode.

D.20 Set_MeasureWindow (MW)

Set a metrological range by definition of a starting point x and an end point as limits for output of measured values.

Parameter: double SP_WindowMin

SP_WindowMin

Direction: Down

Valid values:

Minimum: -FLT_MAX (-3.402823466e+38)

Maximum: FLT_MAX (3.402823466e+38)

Description: Window minimum value.

Parameter: double SP_WindowMax SP_WindowMax
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Window maximum value.

D.21 Get_MeasureWindow (MW)

Get a metrological range by definition of a starting point x and an end point as limits for output of measured values.

Parameter: double SA_WindowMin SA_WindowMin
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Window minimum value.

Parameter: double SA_WindowMax SA_WindowMax
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Window maximum value.

D.22 Set_AnalogOutLimits (QA)

Set parameters of the analog output QA .

Parameter: double SP_RangeBegin SP_RangeBegin
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Lower limit value.

Parameter: double SP_RangeEnd SP_RangeEnd
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Upper limit value.

D.23 Get_AnalogOutLimits (QA)

Get parameters of the analog output QA.

Parameter: double SA_RangeBegin SA_RangeBegin
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Lower limit value.

Parameter: double SA_RangeEnd SA_RangeEnd
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Upper limit value.

D.24 Set_Out1Parameters (Q1)

Set parameters of the switching outputs Q1.

Parameter: double SP_Q1Start SP_Q1Start
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Start value.

Parameter: double SP_Q1Width SP_Q1Width
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Width.

Parameter: double SP_Q1Hysteresis SP_Q1Hysteresis
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Hysteresis.

Parameter: int SP_Q1Negation SP_Q1Negation
Direction: Down
Valid values:
 0 = false
 1 = true
Description: Negation.

D.25 Get_Out1Parameters (Q1)

Get parameters of the switching outputs Q1.

Parameter: double SA_Q1Start SA_Q1Start
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Start value.

Parameter: double SA_Q1Width SA_Q1Width
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Width.

Parameter: double SA_Q1Hysteresis SA_Q1Hysteresis
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Hysteresis.

Parameter: int SA_Q1Negation SA_Q1Negation
Direction: Up
Valid values:
 0 = false
 1 = true
Description: Negation.

D.26 Set_Out2Parameters (Q2)

Set parameters of the switching outputs Q2.

Parameter: double SP_Q2Start SP_Q2Start
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Start value.

Parameter: double SP_Q2Width SP_Q2Width
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Width.

Parameter: double SP_Q2Hysteresis SP_Q2Hysteresis
Direction: Down
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Hysteresis.

Parameter: int SP_Q2Negation SP_Q2Negation
Direction: Down
Valid values:
 0 = false
 1 = true
Description: Negation.

D.27 Get_Out2Parameters (Q2)

Get parameters of the switching outputs Q2.

Parameter: double SA_Q2Start Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Start value.	SA_Q2Start
Parameter: double SA_Q2Width Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Width.	SA_Q2Width
Parameter: double SA_Q2Hysteresis Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Hysteresis.	SA_Q2Hysteresis
Parameter: int SA_Q2Negation Direction: Up Valid values: 0 = false 1 = true Description: Negation.	SA_Q2Negation

D.28 Set_AverageValue (SA)

Set the average value for block wise averaging.

Parameter: int SP_Average Direction: Down Valid values: Minimum: 1 Maximum: 30000 Description: Average value.	SP_Average
--	------------

D.29 Get_AverageValue (SA)

Get the average value for block wise averaging.

Parameter: int SA_Average Direction: Up Valid values: Minimum: 1 Maximum: 30000 Description: Average value.	SA_Average
--	------------

D.30 Set_TriggerDelay (TD)

Set the behaviour of the trigger input.

Parameter: double SP_TriggerDelay

SP_TriggerDelay

Direction: Down

Unit: ms

Valid values:

Minimum: 0.0

Maximum: 300.0

Description: Trigger delay.

Parameter: int SP_TriggerEdge

SP_TriggerEdge

Direction: Down

Valid values:

0= falling

1= rising

Description: Trigger edge.

D.31 Get_TriggerDelay (TD)

Get the behaviour of the trigger input.

Parameter: double SA_TriggerDelay

SA_TriggerDelay

Direction: Up

Unit: ms

Valid values:

Minimum: 0.0

Maximum: 300.0

Description: Trigger delay.

Parameter: int SA_TriggerEdge

SA_TriggerEdge

Direction: Up

Valid values:

0= falling

1= rising

Description: Trigger edge.

D.32 Set_Baudrate (BR)

Set the baudrate of the sensors serial interface.

Attention! If baudrate is set to 9600, the command ID? (currently set to supported) will timeout.

If Set_Autostart is set to ID?, the sensor reboots cyclic and must be send back to manufacturer.

Attention! If baudrate is set to a baudrate, which the computer does not support (e.g. 230400 or 460800), it cannot be changed back and must be send back to manufacturer.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled first (sensor command StopTracking).

Parameter: int SP_SensorBaudrate

SP_SensorBaudrate

Direction: Down

Valid values:

9600
19200
38400
57600
115200
230400
460800

Description: Sensor baudrate.

D.33 Get_Baudrate (BR)

Get the baudrate of the sensors serial interface.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

9600
19200
38400
57600
115200
230400
460800

Description: Sensor baudrate.

D.34 Set_Autostart (AS)

Set which function will be carried out when power becomes available to the sensor.

Parameter: int SP_AutostartCommand

SP_AutostartCommand

Direction: Down

Valid values:

0= ID
1= DM
2= DT
3= DF
4= VM
5= VT
6= TP
7= HW
8= PA
9= MF
10= DT
11= SA
12= SF
13= MW
14= OF
15= SE

16= Q1
 17= Q2
 18= QA
 19= BR
 20= SD
 21= TE
 22= BB
 23= AB
 24= SC
 25= PL
 26= AS

Description: Autostart command.

D.35 Get_Autostart (AS)

Get which function will be carried out when power becomes available to the sensor.

Parameter: int SA_AutostartCommand

SA_AutostartCommand

Direction: Up

Valid values:

0= ID
 1= DM
 2= DT
 3= DF
 4= VM
 5= VT
 6= TP
 7= HW
 8= PA
 9= MF
 10= DT
 11= SA
 12= SF
 13= MW
 14= OF
 15= SE
 16= Q1
 17= Q2
 18= QA
 19= BR
 20= SD
 21= TE
 22= BB
 23= AB
 24= SC
 25= PL
 26= AS
 -1= unknown

Description: Autostart command.

D.36 Set_MeasFreq (MF)

Set the measure frequency of the sensor.

Parameter: int SP_MeasFrequency

SP_MeasFrequency

Direction: Down

Valid values:

Minimum: 1

Maximum: 2000

Description: Measure frequency.

D.37 Get_MeasFreq (MF)

Get the measure frequency of the sensor.

Parameter: int SA_MeasFrequency

SA_MeasFrequency

Direction: Up

Valid values:

Minimum: 1

Maximum: 2000

Description: Measure frequency.

D.38 Set_PilotLaser (PL)

Set the behaviour of the pilot laser of the sensor.

Parameter: int SP_PilotLaser

SP_PilotLaser

Direction: Down

Valid values:

0= off

1= on

2= flashing (2 Hz)

3= flashing (5 Hz)

Description: Pilot laser behaviour.

D.39 Get_PilotLaser (PL)

Get the behaviour of the pilot laser of the sensor.

Parameter: int SA_PilotLaser

SA_PilotLaser

Direction: Up

Valid values:

0= off

1= on

2= flashing (2 Hz)

3= flashing (5 Hz)

Description: Pilot laser behaviour.

D.40 Get_Info (ID)

Retrieve information (like serial number) of the sensor.

Parameter: String SA_Version SA_Version
Direction: Up
Description: Firmware version.

D.41 Get_AllParameters (PA)

Retrieve all parameters from the sensor.

Parameter: int SA_MeasFrequency SA_MeasFrequency
Direction: Up
Valid values:
 Minimum: 1
 Maximum: 2000
Description: Measure frequency.

Parameter: double SA_TriggerDelay SA_TriggerDelay
Direction: Up
Unit: ms
Valid values:
 Minimum: 0.0
 Maximum: 300.0
Description: Trigger delay.

Parameter: int SA_TriggerEdge SA_TriggerEdge
Direction: Up
Valid values:
 0= falling
 1= rising
Description: Trigger edge.

Parameter: int SA_Average SA_Average
Direction: Up
Valid values:
 Minimum: 1
 Maximum: 30000
Description: Average value.

Parameter: double SA_ScaleFactor SA_ScaleFactor
Direction: Up
Valid values:
 Minimum: -10.0
 Maximum: 10.0
Description: Scaling factor.

Parameter: double SA_WindowMin SA_WindowMin
Direction: Up
Valid values:
 Minimum: -FLT_MAX (-3.402823466e+38)
 Maximum: FLT_MAX (3.402823466e+38)
Description: Window minimum value.

Parameter: double SA_WindowMax Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Window maximum value.	SA_WindowMax
Parameter: double SA_Offset Direction: Up Valid values: Minimum: -5000.0 Maximum: 5000.0 Description: Offset value which is set.	SA_Offset
Parameter: int SA_ErrorMode Direction: Up Valid values: 0= last valid value 1= switch to bounds 2= switch to negated bounds Description: Error mode.	SA_ErrorMode
Parameter: double SA_Q1Start Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Start value.	SA_Q1Start
Parameter: double SA_Q1Width Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Width.	SA_Q1Width
Parameter: double SA_Q1Hysteresis Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Hysteresis.	SA_Q1Hysteresis
Parameter: int SA_Q1Negation Direction: Up Valid values: 0= false 1= true Description: Negation.	SA_Q1Negation
Parameter: double SP_Q2Start Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Start value.	SP_Q2Start

Parameter: double SP_Q2Width Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Width.	SP_Q2Width
Parameter: double SP_Q2Hysteresis Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Hysteresis.	SP_Q2Hysteresis
Parameter: int SP_Q2Negation Direction: Down Valid values: 0= false 1= true Description: Negation.	SP_Q2Negation
Parameter: double SA_RangeBegin Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Lower limit value.	SA_RangeBegin
Parameter: double SA_RangeEnd Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Upper limit value.	SA_RangeEnd
Parameter: int SA_SensorBaudrate Direction: Up Valid values: 9600 19200 38400 57600 115200 230400 460800 Description: Sensor baudrate.	SA_SensorBaudrate
Parameter: int SA_OutputFormat Direction: Up Valid values: 0= decimal 1= hex 2= binary Description: Output format of values.	SA_OutputFormat

Parameter: int SA_OutputContent

SA_OutputContent

Direction: Up

Valid values:

- 0= Measuring value
- 1= Measuring value, signal strength
- 2= Measuring value, sensor temperature
- 3= Measuring value, signal strength, sensor temperature

Description: Get which data is transmitted by sensor.

Parameter: int SA_TerminationChar

SA_TerminationChar

Direction: Up

Valid values:

- 0= <CRLF>
- 1= <CR>
- 2= <LF>
- 3= <STX>
- 4= <ETX>
- 5= Tabulator
- 6= Space
- 7= Comma
- 8= Colon
- 9= Semicolon

Description: Termination character.

Parameter: int SA_SSIFormat

SA_SSIFormat

Direction: Up

Valid values:

- 0= binary
- 1= grey code

Description: SSI transmission format.

Parameter: int SA_PilotLaser

SA_PilotLaser

Direction: Up

Valid values:

- 0= off
- 1= on
- 2= flashing (2 Hz)
- 3= flashing (5 Hz)

Description: Pilot laser behaviour.

Parameter: int SA_AutostartCommand

SA_AutostartCommand

Direction: Up

Valid values:

- 0= ID
- 1= DM
- 2= DT
- 3= DF
- 4= VM
- 5= VT
- 6= TP
- 7= HW
- 8= PA
- 9= MF
- 10= DT

11= SA
12= SF
13= MW
14= OF
15= SE
16= Q1
17= Q2
18= QA
19= BR
20= SD
21= TE
22= BB
23= AB
24= SC
25= PL
26= AS
-1= unknown

Description: Autostart command.

D.42 Get_Temperature (TP)

Retrieve the temperature inside of the sensor.

Parameter: double SA_Temperature

SA_Temperature

Direction: Up

Unit: °C

Description: Sensor temperature.

D.43 Get_HWDiagnosis (HW)

Retrieve internal sensor diagnostic information.

Parameter: String SA_Dignosis

SA_Dignosis

Direction: Up

Description: Sensor diagnostic information.

D.44 Reset_Parameters (PR)

Reset all parameters of sensor to factory defaults and return new parameters.

Parameter: int SA_MeasFrequency

SA_MeasFrequency

Direction: Up

Valid values:

Minimum: 1

Maximum: 2000

Description: Measure frequency.

Parameter: double SA_TriggerDelay Direction: Up Unit: ms Valid values: Minimum: 0.0 Maximum: 300.0 Description: Trigger delay.	SA_TriggerDelay
Parameter: int SA_TriggerEdge Direction: Up Valid values: 0= falling 1= rising Description: Trigger edge.	SA_TriggerEdge
Parameter: int SA_Average Direction: Up Valid values: Minimum: 1 Maximum: 30000 Description: Average value.	SA_Average
Parameter: double SA_ScaleFactor Direction: Up Valid values: Minimum: -10.0 Maximum: 10.0 Description: Scaling factor.	SA_ScaleFactor
Parameter: double SA_WindowMin Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Window minimum value.	SA_WindowMin
Parameter: double SA_WindowMax Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Window maximum value.	SA_WindowMax
Parameter: double SA_Offset Direction: Up Valid values: Minimum: -5000.0 Maximum: 5000.0 Description: Offset value which is set.	SA_Offset
Parameter: int SA_ErrorMode Direction: Up Valid values: 0= last valid value 1= switch to bounds 2= switch to negated bounds Description: Error mode.	SA_ErrorMode

Parameter: double SA_Q1Start Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Start value.	SA_Q1Start
Parameter: double SA_Q1Width Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Width.	SA_Q1Width
Parameter: double SA_Q1Hysteresis Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Hysteresis.	SA_Q1Hysteresis
Parameter: int SA_Q1Negation Direction: Up Valid values: 0= false 1= true Description: Negation.	SA_Q1Negation
Parameter: double SP_Q2Start Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Start value.	SP_Q2Start
Parameter: double SP_Q2Width Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Width.	SP_Q2Width
Parameter: double SP_Q2Hysteresis Direction: Down Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Hysteresis.	SP_Q2Hysteresis
Parameter: int SP_Q2Negation Direction: Down Valid values: 0= false 1= true Description: Negation.	SP_Q2Negation

Parameter: double SA_RangeBegin Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Lower limit value.	SA_RangeBegin
Parameter: double SA_RangeEnd Direction: Up Valid values: Minimum: -FLT_MAX (-3.402823466e+38) Maximum: FLT_MAX (3.402823466e+38) Description: Upper limit value.	SA_RangeEnd
Parameter: int SA_SensorBaudrate Direction: Up Valid values: 9600 19200 38400 57600 115200 230400 460800 Description: Sensor baudrate.	SA_SensorBaudrate
Parameter: int SA_OutputFormat Direction: Up Valid values: 0= decimal 1= hex 2= binary Description: Output format of values.	SA_OutputFormat
Parameter: int SA_OutputContent Direction: Up Valid values: 0= Measuring value 1= Measuring value, signal strength 2= Measuring value, sensor temperature 3= Measuring value, signal strength, sensor temperature Description: Get which data is transmitted by sensor.	SA_OutputContent
Parameter: int SA_TerminationChar Direction: Up Valid values: 0= <CRLF> 1= <CR> 2= <LF> 3= <STX> 4= <ETX> 5= Tabulator 6= Space 7= Comma 8= Colon 9= Semicolon Description: Termination character.	SA_TerminationChar

Parameter: int SA_SSIFormat

SA_SSIFormat

Direction: Up

Valid values:

0= binary

1= grey code

Description: SSI transmission format.

Parameter: int SA_PilotLaser

SA_PilotLaser

Direction: Up

Valid values:

0= off

1= on

2= flashing (2 Hz)

3= flashing (5 Hz)

Description: Pilot laser behaviour.

Parameter: int SA_AutostartCommand

SA_AutostartCommand

Direction: Up

Valid values:

0= ID

1= DM

2= DT

3= DF

4= VM

5= VT

6= TP

7= HW

8= PA

9= MF

10= DT

11= SA

12= SF

13= MW

14= OF

15= SE

16= Q1

17= Q2

18= QA

19= BR

20= SD

21= TE

22= BB

23= AB

24= SC

25= PL

26= AS

-1= unknown

Description: Autostart command.

D.45 Trigger_ColdStart (DR)

Reboots the sensor and executes the autostart command.

D.46 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_OutputFormat

IP_OutputFormat

Direction: Down

Valid values:

0= decimal

1= hex

2= binary

Description: Tells the driver the output format of values.

Parameter: int IP_OutputContent

IP_OutputContent

Direction: Down

Valid values:

0= Measuring value

1= Measuring value, signal strength

2= Measuring value, sensor temperature

3= Measuring value, signal strength, sensor temperature

Description: Tells the driver which data is transmitted by sensor.

Parameter: int IP_DistanceTracking

IP_DistanceTracking

Direction: Down

Valid values:

0= no

1= yes

Description: Tells the driver if sensor is in DistanceTracking (DT) mode.

Parameter: int IP_SpeedMeasure

IP_SpeedMeasure

Direction: Down

Valid values:

0= no

1= yes

Description: Tells the driver if sensor is in SpeedMeasure (VM) mode.

Parameter: int IP_SpeedTracking

IP_SpeedTracking

Direction: Down

Valid values:

0= no

1= yes

Description: Tells the driver if sensor is in SpeedTracking (VT) mode.

Parameter: int IP_TerminationChar

IP_TerminationChar

Direction: Down

Valid values:

0= <CRLF>

1= <CR>

2= <LF>

3= <STX>

4= <ETX>

5= Tabulator

6= Space

7= Comma

8= Colon

9= Semicolon

Description: Tells the driver which termination char (TE) the sensor is using.

D.47 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_OutputFormat

IA_OutputFormat

Direction: Up

Valid values:

0= decimal

1= hex

2= binary

Description: Setting used by driver which output format is used.

Parameter: int IA_OutputContent

IA_OutputContent

Direction: Up

Valid values:

0= Measuring value

1= Measuring value, signal strength

2= Measuring value, sensor temperature

3= Measuring value, signal strength, sensor temperature

Description: Setting used by driver which data is transmitted by sensor.

Parameter: int IA_DistanceTracking

IA_DistanceTracking

Direction: Up

Valid values:

0= no

1= yes

Description: Setting used by driver if sensor is in DistanceTracking (DT) mode.

Parameter: int IA_SpeedMeasure

IA_SpeedMeasure

Direction: Up

Valid values:

0= no

1= yes

Description: Setting used by driver if sensor is in SpeedMeasure (VM) mode.

Parameter: int IA_SpeedTracking

IA_SpeedTracking

Direction: Up

Valid values:

0= no

1= yes

Description: Setting used by driver if sensor is in SpeedTracking (VT) mode.

Parameter: int IA_TerminationChar

IA_TerminationChar

Direction: Up

Valid values:

0= <CRLF>

1= <CR>

2= <LF>

3= <STX>

4= <ETX>

5= Tabulator

6= Space

7= Comma

8= Colon

9= Semicolon

Description: Setting used by driver which termination char (TE) the sensor is using.

E Commands for ILD1302

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (additional, e.g. RS422_USB-Converter and RS232 high level interface).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

E.1 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_Sensor	SA_Sensor
Direction: Up	
Description: Name of the sensor.	
Parameter: String SA_SensorType	SA_SensorType
Direction: Up	
Description: Type of the sensor.	
Parameter: String SA_ArticleNumber	SA_ArticleNumber
Direction: Up	
Description: Article number of the sensor.	
Parameter: String SA_Option	SA_Option
Direction: Up	
Description: Option of the sensor.	
Parameter: String SA_SerialNumber	SA_SerialNumber
Direction: Up	
Description: Serial number of the sensor.	
Parameter: double SA_Range	SA_Range
Direction: Up	
Valid values:	
20	
50	
100	
200 (or other on further versions)	
Unit: mm	
Description: Range of the sensor.	
Parameter: String SA_Softwareversion	SA_Softwareversion
Direction: Up	
Description: Software version of the sensor.	
Parameter: String SA_BootLoaderVer	SA_BootLoaderVer
Direction: Up	
Description: Boot loader version of the sensor.	

Parameter: String SA_Date Direction: Up Description: Software release date of the sensor.	SA_Date
Parameter: int SA_OutputType Direction: Up Valid values: 0 = current (4..20mA) 1 = RS422 Description: Data output (only values, not answer) interface of the sensor.	SA_OutputType
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0 = hold last value 1 = error values 2..99 = hold last valid for n values Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler
Parameter: int SA_AvType Direction: Up Valid values: 0 = moving 1 = Median Description: The averaging type.	SA_AvType
Parameter: int SA_MovingCount Direction: Up Valid values: Minimum: 1 Maximum: 128 Description: The moving averaging value, if AvType is moving.	SA_MovingCount
Parameter: int SA_MedianIndex Direction: Up Valid values: 3 5 7 9 Description: The Median value, if AvType is Median.	SA_MedianIndex
Parameter: int SA_ASCII Direction: Up Valid values: 0 = off (binary 2 bytes/value) 1 = on (ASCII 6 bytes/value) Description: Returns the mode the sensor is sending data (only values).	SA_ASCII
Parameter: int SA_OutputMode Direction: Up Valid values: 0 = continuous 1 = timed 2 = triggered Description: The output mode of the sensor.	SA_OutputMode

Parameter: int SA_Keylock Direction: Up Valid values: 0= off (keys enabled) 1= on (keys locked) 2= auto (locked after 5 minutes) Description: The keypad state at the sensor.	SA_Keylock
Parameter: int SA_SaveSettingsMode Direction: Up Valid values: 0= temporary in RAM 1= persistent in Flash Description: The mode if parameters should be temporary or stored persistent.	SA_SaveSettingsMode
Parameter: int SA_ExtInputMode Direction: Up Valid values: 0= used for teaching 1= used as trigger Description: Specifies if the external input is used for teaching or as trigger input.	SA_ExtInputMode
Parameter: int SA_PeakSearching Direction: Up Valid values: 0= global maximum 1= first peak 2= last peak Description: Specifies how the peak searching algorithm does work.	SA_PeakSearching
Parameter: int SA_Threshold Direction: Up Valid values: 0= lower 1= normal 2= higher 3= highest Description: Specifies the spectral threshold.	SA_Threshold
Parameter: double SA_TeachValue1 Direction: Up Valid values: Minimum: 0.0 Maximum: 16368.0 Description: The lower teach limit.	SA_TeachValue1
Parameter: double SA_TeachValue2 Direction: Up Valid values: Minimum: 0.0 Maximum: 16368.0 Description: The higher teach limit.	SA_TeachValue2

E.2 Get_Settings

Retrieve detailed information about the sensor.

Parameter: int SA_OutputType Direction: Up Valid values: 0= current (4..20mA) 1= RS422 Description: Data output (only values, not answer) interface of the sensor.	SA_OutputType
Parameter: double SA_TeachValue1 Direction: Up Valid values: Minimum: 0.0 Maximum: 16368.0 Description: The lower teach limit.	SA_TeachValue1
Parameter: double SA_TeachValue2 Direction: Up Valid values: Minimum: 0.0 Maximum: 16368.0 Description: The higher teach limit.	SA_TeachValue2
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0= hold last value 1= error values 2..99= hold last valid for n values Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler
Parameter: int SA_AvType Direction: Up Valid values: 0= moving 1= Median Description: The averaging type.	SA_AvType
Parameter: int SA_MovingCount Direction: Up Valid values: Minimum: 1 Maximum: 128 Description: The moving averaging value, if AvType is moving.	SA_MovingCount
Parameter: int SA_MedianIndex Direction: Up Valid values: 3 5 7 9 Description: The Median value, if AvType is Median.	SA_MedianIndex

Parameter: int SA_SensorBaudrate Direction: Up Valid values: 0= 115200 Baud 1= 57600 Baud 2= 38400 Baud 3= 19200 Baud 4= 9600 Baud Description: The serial connection baudrate of the sensor.	SA_SensorBaudrate
Parameter: int SA_ASCII Direction: Up Valid values: 0= off (binary 2 bytes/value) 1= on (ASCII 6 bytes/value) Description: Returns the mode the sensor is sending data (only values).	SA_ASCII
Parameter: int SA_OutputMode Direction: Up Valid values: 0= continuous 1= timed 2= triggered Description: The output mode of the sensor.	SA_OutputMode
Parameter: int SA_OutputTime Direction: Up Valid values: Minimum: 0 Maximum: 65536 Unit: ms Description: Data output time of the sensor. It is used for timeout check.	SA_OutputTime
Parameter: int SA_Keylock Direction: Up Valid values: 0= off (keys enabled) 1= on (keys locked) 2= auto (locked after 5 minutes) Description: The keypad state at the sensor.	SA_Keylock
Parameter: int SA_SaveSettingsMode Direction: Up Valid values: 0= temporary in RAM 1= persistent in Flash Description: The mode if parameters should be temporary or stored persistent.	SA_SaveSettingsMode
Parameter: int SA_ExtInputMode Direction: Up Valid values: 0= used for teaching 1= used as trigger Description: Specifies if the external input is used for teaching or as trigger input.	SA_ExtInputMode

Parameter: int SA_PeakSearching

SA_PeakSearching

Direction: Up

Valid values:

0= global maximum

1= first peak

2= last peak

Description: Specifies how the peak searching algorithm does work.

Parameter: int SA_Threshold

SA_Threshold

Direction: Up

Valid values:

0= lower

1= normal

2= higher

2= highest

Description: Specifies the spectral threshold.

Parameter: int SA_Reserved1

SA_Reserved1

Direction: Up

Description: Reserved for further use.

Parameter: int SA_Reserved2

SA_Reserved2

Direction: Up

Description: Reserved for further use.

E.3 Set_KeyLock

Locks/Unlocks the keypad of sensor.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

2= auto (locked after 5 minutes)

Description: The keypad state at the sensor.

E.4 Dat_Out_Off

Switch off data output from sensor.

E.5 Dat_Out_On

Switch on data output from sensor.

E.6 Set_Av

Set averaging type and value of sensor.

Parameter: int SP_AvType

SP_AvType

Direction: Down

Valid values:

0= moving

1= Median

Description: The averaging type.

Parameter: int SP_MovingCount

SP_MovingCount

Direction: Down

Valid values:

Minimum: 1

Maximum: 128

Description: The moving averaging value, if AvType is moving.

Parameter: int SP_MedianIndex

SP_MedianIndex

Direction: Down

Valid values:

3

5

7

9

Description: The Median value, if AvType is Median.

E.7 Set_Baudrate

Set the baudrate of the serial interface of sensor.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled first (sensor command Dat_Out_Off).

Parameter: int SP_SensorBaudrate

SP_SensorBaudrate

Direction: Down

Valid values:

0= 115200 Baud

1= 57600 Baud

2= 38400 Baud

3= 19200 Baud

4= 9600 Baud

Description: The serial connection baudrate of the sensor.

E.8 Set_ErrorHandler

Set the behaviour on invalid values at sensor.

Parameter: int SP_ErrorHandler

SP_ErrorHandler

Direction: Down

Valid values:

0= hold last value

1= error values

2..99= hold last valid for n values

Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

E.9 Laser_Off

Switch the laser off.

E.10 Laser_On

Switch the laser on.

E.11 ASCII_Output

Set digital data transfer (only values, no sensor answer) to ASCII or binary.

Parameter: int SP_ASCII

SP_ASCII

Direction: Down

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

E.12 Set_OutputType

Set the output type of sensor.

Parameter: int SP_OutputType

SP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= RS422

Description: Data output (only values, not answer) interface of the sensor.

E.13 Reset_Boot

Resets the sensor.

E.14 Set_Default

Resets the sensor to factory settings.

E.15 Set_OutputMode

Set the output mode of sensor.

Parameter: int SP_OutputMode

SP_OutputMode

Direction: Down

Valid values:

0= continuous

1= timed

2= triggered

Description: Data output mode of the sensor.

E.16 Set_OutputTime

Set the output time of sensor.

Parameter: int SP_OutputTime

SP_OutputTime

Direction: Down

Valid values:

Minimum: 0

Maximum: 65536

Unit: ms

Description: Data output time of the sensor.

E.17 Set_SaveSettingsMode

Set the save settings mode of sensor.

Parameter: int SP_SaveSettingsMode

SP_SaveSettingsMode

Direction: Down

Valid values:

0 = temporary in RAM

1 = persistent in Flash

Description: The mode if parameters should be temporary or stored persistent.

E.18 Set_ExtInputMode

Set the mode of external input at sensor.

Parameter: int SP_ExtInputMode

SP_ExtInputMode

Direction: Down

Valid values:

0 = used for teaching

1 = used as trigger

Description: Specifies if the external input is used for teaching or as trigger input.

E.19 Set_TeachValue

Set the teaching values at sensor.

Parameter: double SP_TeachValue1

SP_TeachValue1

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 16368.0

Description: The lower teach limit.

Parameter: double SP_TeachValue2

SP_TeachValue2

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 16368.0

Description: The higher teach limit.

E.20 Reset_TeachValue

Reset the teaching values at sensor.

E.21 Set_PeakSearching

Set the mode of external input at sensor.

Parameter: int SP_PeakSearching

SP_PeakSearching

Direction: Down

Valid values:

0= global maximum

1= first peak

2= last peak

Description: Specifies how the peak searching algorithm does work.

E.22 Set_Threshold

Set the spectral threshold of sensor.

Parameter: int SP_Threshold

SP_Threshold

Direction: Down

Valid values:

0= lower

1= normal

2= higher

2= highest

Description: Specifies the spectral threshold.

E.23 Get_Video

Get recent video signal from sensor.

Parameter: String (binary, with 0x00) SA_VideoSignal

SA_VideoSignal

Direction: Up

Valid values:

256 bytes

convertable to 128 WORDS.

Description: Raw video signal

E.24 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_OutputType

IP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= RS422

Description: Tells the driver the data output (only values, not answer) interface of the sensor. It is used for timeout check.

Parameter: int IP_OutputMode

IP_OutputMode

Direction: Down

Valid values:

0= continuous

1= timed

2= triggered

Description: Tells the driver the data output mode of the sensor. It is used for timeout check.

Parameter: int IP_OutputTime

IP_OutputTime

Direction: Down

Valid values:

Minimum: 1

Maximum: 65535

Unit: ms

Description: Tells the driver the data output time of the sensor. It is used for timeout check.

E.25 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_OutputType

IA_OutputType

Direction: Up

Valid values:

0= current (4..20mA)

1= RS422

Description: The data output (only values, not answer) interface of the sensor assumed by the driver for timeout check.

Parameter: int IA_OutputMode

IA_OutputMode

Direction: Up

Valid values:

0= continuous

1= timed

2= triggered

Description: The data output mode of the sensor.

Parameter: int IA_OutputTime

IA_OutputTime

Direction: Up

Valid values:

Minimum: 1

Maximum: 65535

Unit: ms

Description: The data output time of the sensor.

F Commands for ILD1401

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (for sensor ILD1402 in compatibility mode).

F.1 Reset_Boot

Resets the sensor. This command has no parameters.

F.2 Get_Version

Retrieve the sensor software version.

Parameter: String SA_Softwareversion

SA_Softwareversion

Direction: Up

Description: Software version of the sensor.

F.3 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_ArticleNumber

SA_ArticleNumber

Direction: Up

Description: Article number of the sensor

Parameter: String SA_Option

SA_Option

Direction: Up

Description: Option of the sensor

Parameter: String SA_SerialNumber

SA_SerialNumber

Direction: Up

Description: Serial number of the sensor

Parameter: double SA_Range

SA_Range

Direction: Up

Valid values:

5

10

20

50

100

200

250 (or other on further versions)

Unit: mm

Description: Range of the sensor

Parameter: String SA_Softwareversion Direction: Up Description: Software version of the sensor	SA_Softwareversion
Parameter: String SA_Date Direction: Up Description: Software release date of the sensor	SA_Date
Parameter: int SA_OutputType Direction: Up Valid values: 0= analog 1= RS232 Description: Data output (only values, not answer) interface of the sensor	SA_OutputType
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0= hold last value 1= error values Description: If the sensor cannot measure values, it can output the last valid value or it can output an error value (only at analog output).	SA_ErrorHandler
Parameter: int SA_Median_OnOff Direction: Up Valid values: 0= none 1= Median 3 Description: The sensor can perform averaging (Median over 3 values).	SA_Median_OnOff

F.4 Set_OutputChannel

Set the output channel of the sensor.

Parameter: int SP_OutputType Direction: Down Valid values: 0= analog 1= RS232 Description: Specifies data output (only values, not answer) interface of the sensor.	SP_OutputType
--	---------------

F.5 SaveLastMV

Specifies the error handling of sensor if it cannot measure values.

Parameter: int SP_ErrorHandler Direction: Down Valid values: 0= hold last value 1= error values Description: If the sensor cannot measure values, it can output the last valid value or it can output an error value (only at analog output).	SP_ErrorHandler
--	-----------------

F.6 Set_Median

Set the internal averaging mode of the sensor.

Parameter: int SP_Median_OnOff

SP_Median_OnOff

Direction: Down

Valid values:

0= none

1= Median 3

Description: The sensor can perform averaging (Median over 3 values).

F.7 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_Option

IP_Option

Direction: Down

Valid values:

Minimum: 0

Maximum: 999

Description: Tells the driver the sensor option. It is used for data scaling.

F.8 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_Option

IA_Option

Direction: Up

Valid values:

Minimum: 0

Maximum: 999

Description: The option of the sensor assumed by the driver for data scaling.

G Commands for ILD1401-Controller

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

USBIO (native).

G.1 Dat_Out_Off

Controller_ILD1401 does not send data periodically, so the driver has to poll for data in an own thread. This command stops polling data from the controller.

The normal polling intervall is up to 1000 Hz (depending on computer and USB speed), but it can be changed using the sensor command Use_Defaults (parameter IP_Samplerate).

G.2 Dat_Out_On

Starts polling data from controller.

G.3 Set_ControllerReset

Resets the controller. This command has no parameters.

G.4 Get_FirmwareVersion

Returns the version of the controller.

Parameter: String SA_Major	SA_Major
Direction: Up	
Description: Firmware major version.	
Parameter: String SA_Minor	SA_Minor
Direction: Up	
Description: Firmware minor version.	
Parameter: String SA_Version	SA_Version
Direction: Up	
Description: Firmware version.	
Parameter: String SA_Revision	SA_Revision
Direction: Up	
Description: Firmware revision.	

G.5 Get_Settings

Get the settings and state of controller and sensor.

Parameter: int SA_OutOfRange SA_OutOfRange
Direction: Up
Valid values:
0= in range
1= out of range
Description: The sensor is in range or out of range.

Parameter: int SA_OutputType SA_OutputType
Direction: Up
Valid values:
0= digital
1= analog
Description: The output type of the sensor.

Parameter: int SA_ZeroSet SA_ZeroSet
Direction: Up
Valid values:
0= off
1= on
Description: Original measured value or with offset.

Parameter: int SA_Direction SA_Direction
Direction: Up
Valid values:
0= normal
1= inverse
Description: The measured value can be output normal or subtracted from whole measure range.

Parameter: int SA_Unit SA_Unit
Direction: Up
Valid values:
0= metric
1= inch
Description: The measured value can be output in mm or inch.

G.6 Get_BatteryCharge

Get the battery charge state of controller.

Parameter: double SA_BatteryCharge SA_BatteryCharge
Direction: Up
Valid values:
Minimum: 0
Maximum: 100
Unit: %
Description: The charge state of battery.

G.7 Get_Version

Retrieve the sensor software version.

Parameter: String SA_Softwareversion

SA_Softwareversion

Direction: Up

Description: Software version of the sensor.

G.8 Get_ValueMedian

Retrieve the averaged measured value.

Parameter: double SA_ValueMedian

SA_ValueMedian

Direction: Up

Valid values:

Minimum: -measure range

Maximum: +measure range

Unit: mm or inch

Description: Averaged measured value.

G.9 Get_ValueActual

Retrieve the recent measured value.

Parameter: double SA_ValueActual

SA_ValueActual

Direction: Up

Valid values:

Minimum: -measure range

Maximum: +measure range

Unit: mm or inch

Description: Recent measured value.

G.10 Get_SensorRange

Retrieve the measure range of sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Valid values:

5

10

20

50

100

200

250 (or other on further versions)

Unit: mm

Description: Range of the sensor.

G.11 Reset_Boot

Resets the sensor. This command has no parameters.

G.12 Set_Zero

Set the actual value to 0. The offset is subtracted from all following values.

G.13 Set_Unit

Set the unit in controller.

Parameter: int SP_Unit

SP_Unit

Direction: Down

Valid values:

0 = metric

1 = inch

Description: The measured value can be output in mm or inch.

G.14 Set_Direction

Set the direction of measured values.

Parameter: int SP_Direction

SP_Direction

Direction: Down

Valid values:

0 = normal

1 = inverse

Description: The measured value can be output normal or subtracted from whole measure range.

G.15 Set_OutputChannel

Set the output channel of the sensor.

Parameter: int SP_OutputType

SP_OutputType

Direction: Down

Valid values:

0 = digital

1 = analog

Description: The output type of the sensor.

H Commands for ILD1402

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (additional, e.g. RS422_USB-Converter and RS232 high level interface).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

H.1 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_Sensor	SA_Sensor
Direction: Up	
Description: Name of the sensor.	
Parameter: String SA_SensorType	SA_SensorType
Direction: Up	
Description: Type of the sensor.	
Parameter: String SA_ArticleNumber	SA_ArticleNumber
Direction: Up	
Description: Article number of the sensor.	
Parameter: String SA_Option	SA_Option
Direction: Up	
Description: Option of the sensor.	
Parameter: String SA_SerialNumber	SA_SerialNumber
Direction: Up	
Description: Serial number of the sensor.	
Parameter: double SA_Range	SA_Range
Direction: Up	
Valid values:	
5	
10	
20	
50	
100	
200	
250 (or other on further versions)	
Unit: mm	
Description: Range of the sensor.	
Parameter: String SA_Softwareversion	SA_Softwareversion
Direction: Up	
Description: Software version of the sensor.	

Parameter: String SA_BootLoaderVer Direction: Up Description: Boot loader version of the sensor.	SA_BootLoaderVer
Parameter: String SA_Date Direction: Up Description: Software release date of the sensor.	SA_Date
Parameter: int SA_OutputType Direction: Up Valid values: 0= current (4..20mA) 1= RS422 Description: Data output (only values, not answer) interface of the sensor.	SA_OutputType
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0= hold last value 1= error values 2..99= hold last valid for n values Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler
Parameter: int SA_AvType Direction: Up Valid values: 0= moving 1= Median Description: The averaging type.	SA_AvType
Parameter: int SA_MovingCount Direction: Up Valid values: Minimum: 1 Maximum: 128 Description: The moving averaging value, if AvType is moving.	SA_MovingCount
Parameter: int SA_MedianIndex Direction: Up Valid values: 3 5 7 9 Description: The Median value, if AvType is Median.	SA_MedianIndex
Parameter: int SA_Speed Direction: Up Valid values: 0= 1.5kHz 1= 1.0kHz 2= 750Hz 3= 375Hz 4= 50Hz Description: The output speed of the sensor.	SA_Speed

Parameter: int SA_ASCII Direction: Up Valid values: 0= off (binary 2 bytes/value) 1= on (ASCII 6 bytes/value) Description: Returns the mode the sensor is sending data (only values).	SA_ASCII
Parameter: int SA_OutputMode Direction: Up Valid values: 0= continuous 1= timed 2= triggered Description: The output mode of the sensor.	SA_OutputMode
Parameter: int SA_Keylock Direction: Up Valid values: 0= off (keys enabled) 1= on (keys locked) 2= auto (locked after 5 minutes) Description: The keypad state at the sensor.	SA_Keylock
Parameter: int SA_SaveSettingsMode Direction: Up Valid values: 0= temporary in RAM 1= persistant in Flash Description: The mode if parameters should be temporay or stored persistent.	SA_SaveSettingsMode
Parameter: int SA_ExtInputMode Direction: Up Valid values: 0= used for teaching 1= used as trigger Description: Specifies if the external input is used for teaching or as trigger input.	SA_ExtInputMode
Parameter: int SA_PeakSearching Direction: Up Valid values: 0= global maximum 1= first peak 2= last peak Description: Specifies how the peak searching algorithm does work.	SA_PeakSearching
Parameter: int SA_Threshold Direction: Up Valid values: 0= lower 1= normal 2= higher 2= highest Description: Specifies the spectral threshold.	SA_Threshold

Parameter: double SA_TeachValue1 SA_TeachValue1
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 16368.0
Description: The lower teach limit.

Parameter: double SA_TeachValue2 SA_TeachValue2
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 16368.0
Description: The higher teach limit.

H.2 Get_Settings

Retrieve detailed information about the sensor.

Parameter: int SA_OutputType SA_OutputType
Direction: Up
Valid values:
 0= current (4..20mA)
 1= RS422
Description: Data output (only values, not answer) interface of the sensor.

Parameter: double SA_TeachValue1 SA_TeachValue1
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 16368.0
Description: The lower teach limit.

Parameter: double SA_TeachValue2 SA_TeachValue2
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 16368.0
Description: The higher teach limit.

Parameter: int SA_ErrorHandler SA_ErrorHandler
Direction: Up
Valid values:
 0= hold last value
 1= error values
 2..99= hold last valid for n values
Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

Parameter: int SA_AvType SA_AvType
Direction: Up
Valid values:
 0= moving
 1= Median
Description: The averaging type.

Parameter: int SA_MovingCount

SA_MovingCount

Direction: Up

Valid values:

Minimum: 1

Maximum: 128

Description: The moving averaging value, if AvType is moving.

Parameter: int SA_MedianIndex

SA_MedianIndex

Direction: Up

Valid values:

3

5

7

9

Description: The Median value, if AvType is Median.

Parameter: int SA_Speed

SA_Speed

Direction: Up

Valid values:

0= 1.5kHz

1= 1.0kHz

2= 750Hz

3= 375Hz

4= 50Hz

Description: The output speed of the sensor.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

0= 115200 Baud

1= 57600 Baud

2= 38400 Baud

3= 19200 Baud

4= 9600 Baud

Description: The serial connection baudrate of the sensor.

Parameter: int SA_ASCII

SA_ASCII

Direction: Up

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

Parameter: int SA_OutputMode

SA_OutputMode

Direction: Up

Valid values:

0= continuous

1= timed

2= triggered

Description: The output mode of the sensor.

Parameter: int SA_OutputTime Direction: Up Valid values: Minimum: 0 Maximum: 65536 Unit: ms Description: Data output time of the sensor. It is used for timeout check.	SA_OutputTime
Parameter: int SA_Keylock Direction: Up Valid values: 0 = off (keys enabled) 1 = on (keys locked) 2 = auto (locked after 5 minutes) Description: The keypad state at the sensor.	SA_Keylock
Parameter: int SA_SaveSettingsMode Direction: Up Valid values: 0 = temporary in RAM 1 = persistent in Flash Description: The mode if parameters should be temporary or stored persistent.	SA_SaveSettingsMode
Parameter: int SA_ExtInputMode Direction: Up Valid values: 0 = used for teaching 1 = used as trigger Description: Specifies if the external input is used for teaching or as trigger input.	SA_ExtInputMode
Parameter: int SA_PeakSearching Direction: Up Valid values: 0 = global maximum 1 = first peak 2 = last peak Description: Specifies how the peak searching algorithm does work.	SA_PeakSearching
Parameter: int SA_Threshold Direction: Up Valid values: 0 = lower 1 = normal 2 = higher 3 = highest Description: Specifies the spectral threshold.	SA_Threshold
Parameter: int SA_Reserved1 Direction: Up Description: Reserved for further use.	SA_Reserved1
Parameter: int SA_Reserved2 Direction: Up Description: Reserved for further use.	SA_Reserved2

H.3 Set_KeyLock

Locks/Unlocks the keypad of sensor.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

2= auto (locked after 5 minutes)

Description: The keypad state at the sensor.

H.4 Dat_Out_Off

Switch off data output from sensor.

H.5 Dat_Out_On

Switch on data output from sensor.

H.6 Set_Av

Set averaging type and value of sensor.

Parameter: int SP_AvType

SP_AvType

Direction: Down

Valid values:

0= moving

1= Median

Description: The averaging type.

Parameter: int SP_MovingCount

SP_MovingCount

Direction: Down

Valid values:

Minimum: 1

Maximum: 128

Description: The moving averaging value, if AvType is moving.

Parameter: int SP_MedianIndex

SP_MedianIndex

Direction: Down

Valid values:

3

5

7

9

Description: The Median value, if AvType is Median.

H.7 Set_Baudrate

Set the baudrate of the serial interface of sensor.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled first (sensor command Dat_Out_Off).

Parameter: int SP_SensorBaudrate

SP_SensorBaudrate

Direction: Down

Valid values:

- 0= 115200 Baud
- 1= 57600 Baud
- 2= 38400 Baud
- 3= 19200 Baud
- 4= 9600 Baud

Description: The serial connection baudrate of the sensor.

H.8 Set_ErrorHandler

Set the behaviour on invalid values at sensor.

Parameter: int SP_ErrorHandler

SP_ErrorHandler

Direction: Down

Valid values:

- 0= hold last value
- 1= error values
- 2..99= hold last valid for n values

Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

H.9 Set_Speed

Set the data acquisition speed of the sensor.

Parameter: int SP_Speed

SP_Speed

Direction: Down

Valid values:

- 0= 1.5kHz
- 1= 1.0kHz
- 2= 750Hz
- 3= 375Hz
- 4= 50Hz

Description: The output speed of the sensor.

H.10 Laser_Off

Switch the laser off.

H.11 Laser_On

Switch the laser on.

H.12 ASCII_Output

Set digital data transfer (only values, no sensor answer) to ASCII or binary.

Parameter: int SP_ASCII

SP_ASCII

Direction: Down

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

H.13 Set_OutputType

Set the output type of sensor.

Parameter: int SP_OutputType

SP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= RS422

Description: Data output (only values, not answer) interface of the sensor.

H.14 Reset_Boot

Resets the sensor.

H.15 Set_Default

Resets the sensor to factory settings.

H.16 Set_CIMode_1401

Set the sensor in compatibility mode for ILD1401. No other commands expect Set_CIMode_1402 will work now.

Attention! If the interface is IF2004, the sensor cannot be accessed any longer, because IF2004 does not support 38400 Baud (Baudrate of ILD1401).

H.17 Set_OutputMode

Set the output mode of sensor.

Parameter: int SP_OutputMode

SP_OutputMode

Direction: Down

Valid values:

0= continuous

1= timed

2= triggered

Description: Data output mode of the sensor.

H.18 Set_OutputTime

Set the output time of sensor.

Parameter: int SP_OutputTime

SP_OutputTime

Direction: Down

Valid values:

Minimum: 0

Maximum: 65536

Unit: ms

Description: Data output time of the sensor.

H.19 Set_SaveSettingsMode

Set the save settings mode of sensor.

Parameter: int SP_SaveSettingsMode

SP_SaveSettingsMode

Direction: Down

Valid values:

0= temporary in RAM

1= persistent in Flash

Description: The mode if parameters should be temporary or stored persistent.

H.20 Set_ExtInputMode

Set the mode of external input at sensor.

Parameter: int SP_ExtInputMode

SP_ExtInputMode

Direction: Down

Valid values:

0= used for teaching

1= used as trigger

Description: Specifies if the external input is used for teaching or as trigger input.

H.21 Set_TeachValue

Set the teaching values at sensor.

Parameter: double SP_TeachValue1

SP_TeachValue1

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 16368.0

Description: The lower teach limit.

Parameter: double SP_TeachValue2

SP_TeachValue2

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 16368.0

Description: The higher teach limit.

H.22 Reset_TeachValue

Reset the teaching values at sensor.

H.23 Set_PeakSearching

Set the mode of external input at sensor.

Parameter: int SP_PeakSearching

SP_PeakSearching

Direction: Down

Valid values:

0= global maximum

1= first peak

2= last peak

Description: Specifies how the peak searching algorithm does work.

H.24 Set_Threshold

Set the spectral threshold of sensor.

Parameter: int SP_Threshold

SP_Threshold

Direction: Down

Valid values:

0= lower

1= normal

2= higher

2= highest

Description: Specifies the spectral threshold.

H.25 Get_Video

Get recent video signal from sensor.

Parameter: String (binary, with 0x00) SA_VideoSignal

SA_VideoSignal

Direction: Up

Valid values:

256 bytes

convertable to 128 WORDS.

Description: Raw video signal

H.26 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_Speed

IP_Speed

Direction: Down

Valid values:

0= 1.5kHz

1= 1.0kHz

2= 750Hz

3= 375Hz

4= 50Hz

Description: Tells the driver the measure speed of the sensor. It is used for timeout check.

Parameter: int IP_OutputType

IP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= RS422

Description: Tells the driver the data output (only values, not answer) interface of the sensor. It is used for timeout check.

Parameter: int IP_OutputMode

IP_OutputMode

Direction: Down

Valid values:

0= continuous

1= timed

2= triggered

Description: Tells the driver the data output mode of the sensor. It is used for timeout check.

Parameter: int IP_OutputTime

IP_OutputTime

Direction: Down

Valid values:

Minimum: 1

Maximum: 65535

Unit: ms

Description: Tells the driver the data output time of the sensor. It is used for timeout check.

H.27 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_Speed

IA_Speed

Direction: Up

Valid values:

0= 1.5kHz

1= 1.0kHz

2= 750Hz

3= 375Hz

4= 50Hz

Description: The output speed of the sensor assumed by the driver for timeout check.

Parameter: int IA_OutputType

IA_OutputType

Direction: Up

Valid values:

0= current (4..20mA)

1= RS422

Description: The data output (only values, not answer) interface of the sensor assumed by the driver for timeout check.

Parameter: int IA_OutputMode

IA_OutputMode

Direction: Up

Valid values:

0= continuous

1= timed

2= triggered

Description: The data output mode of the sensor.

Parameter: int IA_OutputTime

IA_OutputTime

Direction: Up

Valid values:

Minimum: 1

Maximum: 65535

Unit: ms

Description: The data output time of the sensor.

I Commands for ILD1700

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (additional, e.g. RS422_USB-Converter and RS232 high level interface).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

I.1 Get_MeasValue

Is only useful in trigger mode (ErrorOutput is 2 or 3).

Tells the sensor to measure values without hardware trigger condition (software trigger).

Parameter: int SP_MeasValue SP_MeasValue
Direction: Down
Valid values:
 Minimum: 0
 Maximum: INT_MAX (2147483647)
Description: The number of values to measure.

I.2 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_Sensor SA_Sensor
Direction: Up
Description: Name of the sensor.

Parameter: String SA_SensorType SA_SensorType
Direction: Up
Description: Type of the sensor.

Parameter: String SA_Softwareversion SA_Softwareversion
Direction: Up
Description: Software version of the sensor.

Parameter: int SA_OutputType SA_OutputType
Direction: Up
Valid values:
 0 = current (4..20mA)
 1 = voltage (0..10V)
 2 = RS422
Description: Data output (only values, not answer) interface of the sensor.

Parameter: int SA_ErrorOutput

SA_ErrorOutput

Direction: Up

Valid values:

- 0= sync error mode
- 1= sync switch mode
- 2= trigger error mode
- 3= trigger switch mode

Description: Sync/trigger and error/switch mode respectively of the sensor.

Parameter: int SA_Speed

SA_Speed

Direction: Up

Valid values:

- 0= 2.5kHz
- 1= 1.25kHz
- 2= 625Hz
- 3= 312.5Hz

Description: The output speed of the sensor.

Parameter: double SA_Samplerate

SA_Samplerate

Direction: Up

Valid values:

- 2500
- 1250
- 625
- 312.5 (or other on specific sensor settings)

Unit: Hz

Description: The output speed of the sensor.

Parameter: int SA_AvType

SA_AvType

Direction: Up

Valid values:

- 0= recursive
- 1= moving
- 2= Median

Description: The averaging type.

Parameter: int SA_AvIndex

SA_AvIndex

Direction: Up

Valid values:

- Minimum:** 0
- Maximum:** 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

Parameter: int SA_ErrorHandler

SA_ErrorHandler

Direction: Up

Valid values:

- 0= error values
- 1= hold last value

Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

Parameter: int SA_Sync_TrgMode

SA_Sync_TrgMode

Direction: Up

Valid values:

Condition: at sync mode (ErrorOutput is 0 or 1):

0= synchronous master off

1= synchronous master on

2= slave

3= alternating synchronous master

Valid values:

Condition: at trigger mode (ErrorOutput is 2 or 3):

0= edge L/H

1= edge H/L

2= level H

3= level L

Description: The sync mode or the trigger mode of the sensor.

Parameter: int SA_ASCII

SA_ASCII

Direction: Up

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

Parameter: int SA_Keylock

SA_Keylock

Direction: Up

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Valid values:

2

10

20

50

100

200

250

500

750 (or other on further versions)

Unit: mm

Description: Range of the sensor.

Parameter: String SA_SerialNumber

SA_SerialNumber

Direction: Up

Description: Serial number of the sensor.

Parameter: String SA_Option

SA_Option

Direction: Up

Description: Option of the sensor.

Parameter: String SA_ArticleNumber Direction: Up Description: Article number of the sensor.	SA_ArticleNumber
Parameter: String SA_Date Direction: Up Description: Software release date of the sensor.	SA_Date
Parameter: String SA_BootLoaderVer Direction: Up Description: Boot loader version of the sensor.	SA_BootLoaderVer
Parameter: String SA_SWType Direction: Up Description: Software type of the sensor.	SA_SWType
Parameter: int SA_EnableFlash Direction: Up Valid values: 0= locked 1= enabled Description: The flash is locked or enabled for writing.	SA_EnableFlash

I.3 Get_Settings

Retrieve detailed information about the sensor.

Parameter: int SA_OutputType Direction: Up Valid values: 0= current (4..20mA) 1= voltage (0..10V) 2= RS422 Description: Data output (only values, not answer) interface of the sensor.	SA_OutputType
Parameter: int SA_Speed Direction: Up Valid values: 0= 2.5kHz 1= 1.25kHz 2= 625Hz 3= 312.5Hz Description: The output speed of the sensor.	SA_Speed
Parameter: int SA_AvIndex Direction: Up Valid values: Minimum: 0 Maximum: 15 Description: The averaging index. Averaging $N = 2^{AvIndex}$.	SA_AvIndex

Parameter: int SA_ErrorHandler

SA_ErrorHandler

Direction: Up

Valid values:

0= error values

1= hold last value

Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

Parameter: int SA_Sync_TriggerMode

SA_Sync_TriggerMode

Direction: Up

Valid values:

Condition: at sync mode (ErrorOutput is 0 or 1):

0= synchronous master off

1= synchronous master on

2= slave

3= alternating synchronous master

Valid values:

Condition: at trigger mode (ErrorOutput is 2 or 3):

0= edge L/H

1= edge H/L

2= level H

3= level L

Description: The sync mode or the trigger mode of the sensor.

Parameter: int SA_AvType

SA_AvType

Direction: Up

Valid values:

0= recursive

1= moving

2= Median

Description: The averaging type.

Parameter: int SA_ErrorOutput

SA_ErrorOutput

Direction: Up

Valid values:

0= sync error mode

1= sync switch mode

2= trigger error mode

3= trigger switch mode

Description: Sync/trigger and error/switch mode respectively of the sensor.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

0= 115200 Baud

1= 57600 Baud

2= 19200 Baud

3= 9600 Baud

Description: The serial connection baudrate of the sensor.

Parameter: int SA_ASCII

SA_ASCII

Direction: Up

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

Parameter: int SA_Upper_limit

SA_Upper_limit

Direction: Up

Valid values:

Minimum: 0

Maximum: 16368

Description: The upper limit of the sensor.

Parameter: int SA_Lower_limit

SA_Lower_limit

Direction: Up

Valid values:

Minimum: 0

Maximum: 16368

Description: The lower limit of the sensor.

Parameter: int SA_Upper_hysteresis

SA_Upper_hysteresis

Direction: Up

Valid values:

Minimum: 0

Maximum: 16368

Description: The upper hysteresis of the sensor.

Parameter: int SA_Lower_hysteresis

SA_Lower_hysteresis

Direction: Up

Valid values:

Minimum: 0

Maximum: 16368

Description: The lower hysteresis of the sensor.

Parameter: int SA_Master_value

SA_Master_value

Direction: Up

Valid values:

Minimum: 0

Maximum: 16368

Description: The master value of the sensor.

Parameter: int SA_Master_MidPoint_Setup

SA_Master_MidPoint_Setup

Direction: Up

Valid values:

Condition: switch mode:

0/2= Not mastered

1/3= Mastered

Valid values:

Condition: error mode:

0/1 = mid-point value not set

2/3= mid-point value set

Description: The master and midpoint of the sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Valid values:

2
10
20
50
100
200
250
500
750 (or other on further versions)

Unit: mm

Description: Range of the sensor.

Parameter: int SA_AssignLimits_ErrorOutput

SA_AssignLimits_ErrorOutput

Direction: Up

Valid values:

0= Set_LowerLimit_F1
1= Set_UpperLimit_F1

Description: The assignment of the error outputs of the sensor.

Parameter: int SA_Keylock

SA_Keylock

Direction: Up

Valid values:

0= off (keys enabled)
1= on (keys locked)

Description: The keypad state at the sensor.

Parameter: int SA_DatOut

SA_DatOut

Direction: Up

Valid values:

0= Dat_Out_Off
1= Dat_Out_On

Description: Data output from sensor.

Parameter: int SA_LaserState

SA_LaserState

Direction: Up

Valid values:

0= Laser_Off
1= Laser_On

Description: Laser state of sensor.

Parameter: int SA_EnableFlash

SA_EnableFlash

Direction: Up

Valid values:

0= locked
1= enabled

Description: The flash is locked or enabled for writing.

I.4 Set_KeyLock

Locks/Unlocks the keypad of sensor.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

I.5 WriteFlashZero

Locks/enables the flash of sensor for writing.

Parameter: int SP_EnableFlash

SP_EnableFlash

Direction: Down

Valid values:

0= locked

1= enabled

Description: The flash is locked or enabled for writing.

I.6 Set_Av0

Set averaging index AvIndex= 0, Averaging N= 1.

I.7 Set_Av1

Set averaging index AvIndex= 2, Averaging N= 4.

I.8 Set_Av2

Set averaging index AvIndex= 5, Averaging N= 32.

I.9 Set_Av3

Set averaging index AvIndex= 7, Averaging N= 128.

I.10 Set_AvX

Set averaging index of sensor.

Parameter: int SP_AvIndex

SP_AvIndex

Direction: Down

Valid values:

Minimum: 0

Maximum: 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

I.11 Dat_Out_Off

Switch off data output from sensor.

I.12 Dat_Out_On

Switch on data output from sensor.

I.13 Set_Av_T

Set averaging type of sensor.

Parameter: int SP_AvType

SP_AvType

Direction: Down

Valid values:

0= recursive

1= moving

2= Median

Description: The averaging type.

I.14 Set_Limits

Set sensor limits.

Parameter: int SP_Upper_limit

SP_Upper_limit

Direction: Down

Valid values:

Minimum: 0

Maximum: 16368

Description: Upper limit.

Parameter: int SP_Lower_limit

SP_Lower_limit

Direction: Down

Valid values:

Minimum: 0

Maximum: 16368

Description: Lower limit.

Parameter: int SP_Upper_hysteresis

SP_Upper_hysteresis

Direction: Down

Valid values:

Minimum: 0

Maximum: 16368

Description: Upper hysteresis.

Parameter: int SP_Lower_hysteresis

SP_Lower_hysteresis

Direction: Down

Valid values:

Minimum: 0

Maximum: 16368

Description: Lower hysteresis.

Parameter: int SP_Master_value
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 16368
Description: Master value.

SP_Master_value

I.15 Set_Baudrate

Set the baudrate of the serial interface of sensor.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled first (sensor command Dat_Out_Off).

Parameter: int SP_SensorBaudrate
Direction: Down
Valid values:
 0= 115200 Baud
 1= 57600 Baud
 2= 19200 Baud
 3= 9600 Baud
Description: The serial connection baudrate of the sensor.

SP_SensorBaudrate

I.16 Set_ErrorHandler

Set the behaviour on invalid values at sensor.

Parameter: int SP_ErrorHandler
Direction: Down
Valid values:
 0= error values
 1= hold last value
Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

SP_ErrorHandler

I.17 Set_Sync_TrgMode

Set the synchronisation and trigger mode respectively.

Parameter: int SP_Sync_TrgMode
Direction: Down
Valid values:
 Condition: at sync mode (ErrorOutput is 0 or 1):
 0= synchronous master off
 1= synchronous master on
 2= slave
 3= alternating synchronous master
 Valid values:
 Condition: at trigger mode (ErrorOutput is 2 or 3):
 0= edge L/H
 1= edge H/L
 2= level H
 3= level L
Description: The sync mode or the trigger mode of the sensor.

SP_Sync_TrgMode

I.18 Set_UpperLimit_F1

Assign the upper limit of sensor to error output 1 and lower limit to error output 2.

I.19 Set_LowerLimit_F1

Assign the lower limit of sensor to error output 1 and upper limit to error output 2.

I.20 Set_Speed

Set the data acquisition speed of the sensor.

Parameter: int SP_Speed

SP_Speed

Direction: Down

Valid values:

0= 2.5kHz

1= 1.25kHz

2= 625Hz

3= 312.5Hz

Description: The output speed of the sensor.

I.21 Laser_Off

Switch the laser off.

I.22 Laser_On

Switch the laser on.

I.23 ASCII_Output

Set digital data transfer (only values, no sensor answer) to ASCII or binary.

Parameter: int SP_ASCII

SP_ASCII

Direction: Down

Valid values:

0= off (binary 2 bytes/value)

1= on (ASCII 6 bytes/value)

Description: Returns the mode the sensor is sending data (only values).

I.24 Set_OutputType

Set the output type of sensor.

Parameter: int SP_OutputType

SP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= voltage (0..10V)

2= RS422

Description: Data output (only values, not answer) interface of the sensor.

I.25 Set_ErrorOutput

Set the synchronisation or trigger mode and set the error or switch mode.

Parameter: int SP_ErrorOutput

SP_ErrorOutput

Direction: Down

Valid values:

- 0= sync error mode
- 1= sync switch mode
- 2= trigger error mode
- 3= trigger switch mode

Description: Sync/trigger and error/switch mode respectively of the sensor.

I.26 Reset_Boot

Resets the sensor.

I.27 Set_Default

Resets the sensor to factory settings.

I.28 Set_VideoMode

Enter/Leave the video mode of the sensor.

Parameter: int SP_VideoMode

SP_VideoMode

Direction: Down

Valid values:

- 0= off
- 1= on

Description: Switch video mode on or off.

I.29 Get_Video

Get recent video signal from sensor.

Parameter: String (binary, with 0x00) SA_VideoSignal

SA_VideoSignal

Direction: Up

Valid values:

- 512 bytes
- each byte is an intensity value.

Description: Raw video signal

I.30 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_Speed

IP_Speed

Direction: Down

Valid values:

0= 2.5kHz

1= 1.25kHz

2= 625Hz

3= 312.5Hz

Description: Tells the driver the measure speed of the sensor. It is used for timeout check.

Parameter: int IP_ErrorOutput

IP_ErrorOutput

Direction: Down

Valid values:

0= sync error mode

1= sync switch mode

2= trigger error mode

3= trigger switch mode

Description: Tells the driver the sync/trigger and error/switch mode respectively of the sensor. It is used for timeout check.

Parameter: int IP_Sync_TriggerMode

IP_Sync_TriggerMode

Direction: Down

Valid values:

Condition: at sync mode (ErrorOutput is 0 or 1):

0= synchronous master off

1= synchronous master on

2= slave

3= alternating synchronous master

Valid values:

Condition: at trigger mode (ErrorOutput is 2 or 3):

0= edge L/H

1= edge H/L

2= level H

3= level L

Description: Tells the driver the sync mode or the trigger mode of the sensor. It is used for timeout check.

Parameter: int IP_OutputType

IP_OutputType

Direction: Down

Valid values:

0= current (4..20mA)

1= voltage (0..10V)

2= RS422

Description: Tells the driver the data output (only values, not answer) interface of the sensor. It is used for timeout check.

I.31 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_Speed

IA_Speed

Direction: Up

Valid values:

0= 2.5kHz

1= 1.25kHz

2= 625Hz

3= 312.5Hz

Description: The output speed of the sensor assumed by the driver for timeout check.

Parameter: int IA_ErrorOutput

IA_ErrorOutput

Direction: Up

Valid values:

0= sync error mode

1= sync switch mode

2= trigger error mode

3= trigger switch mode

Description: The sync/trigger and error/switch mode respectively of the sensor assumed by the driver for timeout check.

Parameter: int IA_Sync_TriggerMode

IA_Sync_TriggerMode

Direction: Up

Valid values:

Condition: at sync mode (ErrorOutput is 0 or 1):

0= synchronous master off

1= synchronous master on

2= slave

3= alternating synchronous master

Valid values:

Condition: at trigger mode (ErrorOutput is 2 or 3):

0= edge L/H

1= edge H/L

2= level H

3= level L

Description: The sync mode or the trigger mode of the sensor assumed by the driver for timeout check.

Parameter: int IA_OutputType

IA_OutputType

Direction: Up

Valid values:

0= current (4..20mA)

1= voltage (0..10V)

2= RS422

Description: The data output (only values, not answer) interface of the sensor assumed by the driver for timeout check.

J Commands for ILD1800

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native, but not with a RS422_USB-Converter and RS232 high level interface).

IF2004 (native).

TCP/IP (additional, e.g. RS232 to TCP/IP comm server and RS232 high level interface).

J.1 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_Sensor SA_Sensor

Direction: Up

Description: Name of the sensor.

Parameter: String SA_SensorType SA_SensorType

Direction: Up

Description: Type of the sensor.

Parameter: double SA_Samplerate SA_Samplerate

Direction: Up

Valid values:

5000

2500

Unit: Hz

Description: The output speed of the sensor.

Parameter: double SA_Range SA_Range

Direction: Up

Valid values:

2

10

20

50

100

200

500

750 (or other on further versions)

Unit: mm

Description: Range of the sensor.

Parameter: String SA_Option SA_Option

Direction: Up

Description: Option of the sensor.

Parameter: String SA_SerialNumber SA_SerialNumber

Direction: Up

Description: Serial number of the sensor.

Parameter: int SA_AvIndex

SA_AvIndex

Direction: Up

Valid values:

Minimum: 0

Maximum: 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

Parameter: int SA_Output_Analog

SA_Output_Analog

Direction: Up

Valid values:

0 = not avail

1 = voltage (0..10V)

Description: The analog output type.

Parameter: int SA_Output_Digital

SA_Output_Digital

Direction: Up

Valid values:

0 = not avail

1 = RS232

2 = RS422

Description: The digital output type.

J.2 Get_Version

Retrieve the sensor software version.

Parameter: String SA_Softwareversion

SA_Softwareversion

Direction: Up

Description: Software version of the sensor.

J.3 Zero

Autozero the analog output value.

J.4 Set_Av0

Set averaging index AvIndex= 0, Averaging N= 1.

J.5 Set_Av1

Set averaging index AvIndex= 2, Averaging N= 4.

J.6 Set_Av2

Set averaging index AvIndex= 5, Averaging N= 32.

J.7 Set_Av3

Set averaging index AvIndex= 7, Averaging N= 128.

J.8 Set_AvX

Set averaging index of sensor.

Parameter: int SP_AvIndex

SP_AvIndex

Direction: Down

Valid values:

Minimum: 0

Maximum: 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

J.9 Dat_Out_Off

Switch off data output from sensor.

J.10 Dat_Out_On

Switch on data output from sensor.

J.11 Displacement

With special option this sensor can do thickness measurement. This command switch the sensor back to displacement measurement.

J.12 Thickness

With special option this sensor can do thickness measurement. This command switch the sensor to thickness measurement.

J.13 Multilayer

With special option this sensor can do thickness measurement. This command switch the sensor to multilayer measurement.

J.14 Refraction

With special option this sensor can do thickness measurement. This command sets the refraction index for multilayer measurement.

Parameter: double SP_RefractIndex

SP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: The refractive index.

J.15 Set_Av_T

Set averaging type of sensor.

Parameter: int SP_AvType

SP_AvType

Direction: Down

Valid values:

0= recursive

1= moving

2= Median

Description: The averaging type.

J.16 Reset_Boot

Resets the sensor.

K Commands for ILD2000

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

IF2004 (native).

K.1 Set_AvX

Set averaging index of sensor.

Parameter: int SP_AvIndex

SP_AvIndex

Direction: Down

Valid values:

Minimum: 0

Maximum: 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

K.2 Save_EEPROM

Store sensor settings in EEPROM.

L Commands for ILD2200

See sensor manual for detailed description of sensor commands. Driver for ILD2200 also includes ILD2220 support.

This sensor supports following interfaces:

RS232 (additional, e.g. RS422_USB-Converter and RS232 high level interface).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

L.1 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_Sensor	SA_Sensor
Direction: Up	
Description: Name of the sensor.	
Parameter: String SA_SensorType	SA_SensorType
Direction: Up	
Description: Type of the sensor.	
Parameter: double SA_Samplerate	SA_Samplerate
Direction: Up	
Valid values:	
20000	
10000	
5000	
2500	
Unit: Hz	
Description: The output speed of the sensor.	
Parameter: double SA_Range	SA_Range
Direction: Up	
Valid values:	
2	
10	
20	
40	
50	
100	
200 (or other on further versions)	
Unit: mm	
Description: Range of the sensor.	
Parameter: String SA_Option	SA_Option
Direction: Up	
Description: Option of the sensor.	

Parameter: String SA_SerialNumber Direction: Up Description: Serial number of the sensor.	SA_SerialNumber
Parameter: int SA_AvIndex Direction: Up Valid values: Minimum: 0 Maximum: 15 Description: The averaging index. Averaging $N = 2^{AvIndex}$.	SA_AvIndex
Parameter: int SA_Output_Analog Direction: Up Valid values: 0 = not avail 1 = voltage (+-5V) Description: The analog output type.	SA_Output_Analog
Parameter: int SA_Output_Digital Direction: Up Valid values: 0 = not avail 2 = RS422 Description: The digital output type.	SA_Output_Digital

L.2 Get_Settings

Retrieve detailed information about the sensor.

Parameter: int SA_Speed Direction: Up Valid values: 0 = 10kHz 1 = 5kHz 2 = 2.5kHz 3 = 20kHz Description: The output speed of the sensor.	SA_Speed
Parameter: int SA_AvIndex Direction: Up Valid values: Minimum: 0 Maximum: 15 Description: The averaging index. Averaging $N = 2^{AvIndex}$.	SA_AvIndex
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0 = error values 1 = hold last value Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler

Parameter: int SA_AvType

SA_AvType

Direction: Up

Valid values:

0= recursive

1= moving

2= Median

Description: The averaging type.

Parameter: double SA_OffsetValue

SA_OffsetValue

Direction: Up

Valid values:

Minimum: -50.0

Maximum: +50.0

Unit: %

Description: The offset value of the sensor.

Parameter: int SA_ZeroPoint

SA_ZeroPoint

Direction: Up

Valid values:

0= absolute

1= relative (zero is set)

Description: Autozero off/on.

Parameter: double SA_Range

SA_Range

Direction: Up

Valid values:

2

10

20

50

100

200

500

750 (or other on further versions)

Unit: mm

Description: Range of the sensor.

Parameter: int SA_Keylock

SA_Keylock

Direction: Up

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

Parameter: int SA_DatOut

SA_DatOut

Direction: Up

Valid values:

0= Dat_Out_Off

1= Dat_Out_On

Description: Data output from sensor.

Parameter: int SA_LaserState

SA_LaserState

Direction: Up

Valid values:

0= Laser_Off

1= Laser_On

Description: Laser state of sensor.

L.3 Get_Version

Retrieve the sensor software version.

Parameter: String SA_Softwareversion

SA_Softwareversion

Direction: Up

Description: Software version of the sensor.

L.4 Set_KeyLock

Locks/Unlocks the keypad of sensor.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

L.5 Zero

Autozero the analog output value.

L.6 Set_Av0

Set averaging index AvIndex= 0, Averaging N= 1.

L.7 Set_Av1

Set averaging index AvIndex= 2, Averaging N= 4.

L.8 Set_Av2

Set averaging index AvIndex= 5, Averaging N= 32.

L.9 Set_Av3

Set averaging index AvIndex= 7, Averaging N= 128.

L.10 Set_AvX

Set averaging index of sensor.

Parameter: int SP_AvIndex

SP_AvIndex

Direction: Down

Valid values:

Minimum: 0

Maximum: 15

Description: The averaging index. Averaging $N = 2^{AvIndex}$.

L.11 Dat_Out_Off

Switch off data output from sensor.

L.12 Dat_Out_On

Switch on data output from sensor.

L.13 Set_Av_T

Set averaging type of sensor.

Parameter: int SP_AvType

SP_AvType

Direction: Down

Valid values:

0 = recursive

1 = moving

2 = Median

Description: The averaging type.

L.14 Laser_Off

Switch off the laser.

L.15 Laser_On

Switch on the laser.

L.16 Transmit_Intensity

Tells the sensor to transmit the intensity value after each distance value.

This command is only available with sensor option 204.

This setting will not be stored persistently in the sensor so it has to be set after each reset or power on.

Parameter: int SP_TransmitIntensity

SP_TransmitIntensity

Direction: Down

Valid values:

0 = no

1 = yes

Description: Transmit intensity value.

L.17 Reset_Boot

Resets the sensor.

M Commands for IFD2400

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

M.1 Set_ActiveSensor (SEN)

Set the active sensor (up to 6 sensors can be stored in controller) for measurement. After this command, the internal range is resetted because the old one is no longer valid and the new one is not known until command Get_Range or Get_Status is called.

Parameter: int SP_Sensor

SP_Sensor

Direction: Down

Valid values:

Minimum: 0

Maximum: 5

Description: Number of active sensor.

M.2 Get_ActiveSensor (SEN?)

Get the active sensor.

Parameter: int SA_Sensor

SA_Sensor

Direction: Up

Valid values:

Minimum: 0

Maximum: 5

Description: Number of active sensor.

M.3 Get_Range (SCA)

Get the range of active sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: μm

Description: Range of active sensor.

M.4 Acquire_DarkSig (DRK)

Acquire dark signal.

Parameter: int SA_MinSRIndex

SA_MinSRIndex

Direction: Up

Valid values:

3= 30 Hz

4= 100 Hz

5= 300 Hz

6= 1000 Hz

Description: Minimal samplerate index.

M.5 FastDark (FDK)

Acquire dark signal for active sensor and samplerate.

Parameter: int SP_AveragingForDark

SP_AveragingForDark

Direction: Down

Valid values:

Minimum: 1

Maximum: 99

Description: Averaging factor for dark.

Parameter: int SP_Weighting

SP_Weighting

Direction: Down

Valid values:

Minimum: 1

Maximum: 32767

Description: Weighting factor.

M.6 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

3= 30 Hz

4= 100 Hz

5= 300 Hz

6= 1000 Hz

Description: Samplerate index.

M.7 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

3= 30 Hz

4= 100 Hz

5= 300 Hz

6= 1000 Hz

Description: Samplerate index.

M.8 Set_SumFreq (SHZ)

Set the sum frequency (only for sensors with option double frequency).

Parameter: int SP_SumFreq
Direction: Down
Valid values:
 Minimum: 1
 Maximum: 490
Unit: Hz
Description: Sum Frequency.

SP_SumFreq

M.9 Get_SumFreq (SHZ?)

Get the current sum frequency (only for controller with option double frequency).

Parameter: int SA_SumFreq
Direction: Up
Valid values:
 Minimum: 1
 Maximum: 490
Unit: Hz
Description: Sum Frequency.

SA_SumFreq

M.10 Set_DutyCycle (DCY)

Set the duty cycle (only for controller with option double frequency).

Parameter: int SP_DutyCycle
Direction: Down
Valid values:
 Minimum: 1
 Maximum: 100
Unit: %
Description: Sum Frequency.

SP_DutyCycle

M.11 Get_DutyCycle (DCY?)

Get the current duty cycle (only for sensors with option double frequency).

Parameter: int SA_DutyCycle
Direction: Up
Valid values:
 Minimum: 1
 Maximum: 100
Unit: %
Description: Sum Frequency.

SA_DutyCycle

M.12 Set_MeasureMode (MOD)

Set the measure mode.

Parameter: int SP_MeasureMode

SP_MeasureMode

Direction: Down

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the controller.

M.13 Get_MeasureMode (MOD?)

Get the current measure mode.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the controller.

M.14 Set_RefractIndex (SRI)

Set the refraction index for thickness measurement.

Parameter: double SP_RefractIndex

SP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the controller for thickness measurement.

M.15 Get_RefractIndex (SRI?)

Get the current refraction index.

Parameter: double SA_RefractIndex

SA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the controller for thickness measurement.

M.16 Set_Threshold (THR)

Set the detection threshold.

Parameter: int SP_Threshold

SP_Threshold

Direction: Down

Valid values:

Minimum: 0

Maximum: 4094

Description: Threshold of the controller.

M.17 Get_Threshold (THR?)

Get the current detection threshold.

Parameter: int SA_Threshold

SA_Threshold

Direction: Up

Valid values:

Minimum: 0

Maximum: 4094

Description: Threshold of the controller.

M.18 Set_KeyLock (LOC)

Locks/Unlocks the keypad of controller.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the controller.

M.19 Get_KeyLock (LOC?)

Returns the lock state of controller keypad.

Parameter: int SA_Keylock

SA_Keylock

Direction: Up

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

M.20 Set_Averaging (AVR)

Set data averaging at controller.

Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

M.21 Get_Averaging (AVR?)

Returns current data averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

M.22 Set_SpectralAv (AVS)

Set spectral data averaging at controller.

Note! Acquisition speed will be reduced.

Parameter: int SP_SpectralAv

SP_SpectralAv

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Spectral data averaging factor.

M.23 Get_SpectralAv (AVS?)

Returns current spectral data averaging at controller.

Parameter: int SA_SpectralAv

SA_SpectralAv

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Spectral data averaging factor.

M.24 Continue (CTN)

Continues data acquisition (after a trigger command or an error).

M.25 Start_Trigger (TRG)

Starts the trigger mode at controller.

M.26 End_Trigger

Stops the trigger mode at controller.

M.27 SingleShot_Trg (TRE)

Starts the single shot trigger mode at controller (only for sensors with option single shot trigger).

M.28 Get_Status (STS)

Retrieve detailed information about the controller and sensor.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

3= 30 Hz

4= 100 Hz

5= 300 Hz

6= 1000 Hz

Description: Samplerate index.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the controller.

Parameter: int SA_Sensor

SA_Sensor

Direction: Up

Valid values:

Minimum: 0

Maximum: 5

Description: Number of active sensor.

Parameter: int SA_ASCII

SA_ASCII

Direction: Up

Description: Data Transfer mode (values) of controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

Parameter: int SA_X1..8 Direction: Up Valid values: 0= off 1= RS232 Description: Selection of data used by the sensor. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used). X5 means (not used / Intensity1). X6 means (not used / Intensity2). X7 means (Barycenter / Barycenter1). X8 means (not used / Barycenter2).	SA_X1..8
Parameter: int SA_0_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC1: 0= X1, 1= X2, ..., 7= X8.	SA_0_SOD
Parameter: int SA_0_0V Direction: Up Valid values: Minimum: 0 Maximum: 32767 Description: Analog out BNC1: Specifies the value which should output 0 V.	SA_0_0V
Parameter: int SA_0_10V Direction: Up Valid values: Minimum: 0 Maximum: 32767 Description: Analog out BNC1: Specifies the value which should output 10 V.	SA_0_10V
Parameter: int SA_1_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC2: 0= X1, 1= X2, ..., 7= X8.	SA_1_SOD
Parameter: int SA_1_0V Direction: Up Valid values: Minimum: 0 Maximum: 32767 Description: Analog out BNC2: Specifies the value which should output 0 V.	SA_1_0V

Parameter: int SA_1_10V SA_1_10V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Analog out BNC2: Specifies the value which should output 10 V.

Parameter: double SA_Range SA_Range
Direction: Up
Unit: μm
Description: Range of active sensor.

M.29 Get_Version (VER)

Get the software version of the controller.

Parameter: String SA_Version SA_Version
Direction: Up
Description: Software version of controller.

M.30 Set_OutputData (SOD)

Select the data to be output from controller.

Parameter: int SP_X1..8 SP_X1..8
Direction: Down
Valid values:
 0 = off
 1 = RS232
Description: Selection of data used by the sensor. This is necessary for data conversion and scaling.
 Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.
 X1 means (Distance / Thickness).
 X2 means (not used / Distance1).
 X3 means (not used / Distance2).
 X4 means (Intensity / not used).
 X5 means (not used / Intensity1).
 X6 means (not used / Intensity2).
 X7 means (Barycenter / Barycenter1).
 X8 means (not used / Barycenter2).

M.31 Get_OutputData (SOD?)

Get the data output from controller.

Parameter: int SA_X1..8 SA_X1..8
Direction: Up
Valid values:
 0 = off

1 = RS232

Description: Selection of data used by the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

M.32 Set_Ascii (ASC)

Set digital data transfer (only values, no sensor answer) to ASCII mode.

M.33 Set_Binary (BIN)

Set digital data transfer (only values, no sensor answer) to binary mode.

M.34 Set_AnalogOut (ANA)

Setup the analog outputs.

Parameter: int SP_OutNr

SP_OutNr

Direction: Down

Valid values:

0 = BNC1

1 = BNC2

Description: Number of analog output.

Parameter: int SP_SOD

SP_SOD

Direction: Down

Valid values:

Minimum: 0

Maximum: 7

Description: Specifies the number (X1..X8) to output at analog out.

Parameter: int SP_OV

SP_OV

Direction: Down

Valid values:

Minimum: 0

Maximum: 32767

Description: Specifies the value which should output 0 V.

Parameter: int SP_10V SP_10V
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Specifies the value which should output 10 V.

M.35 Get_AnalogOut (ANA?)

Get settings of analog outputs.

Parameter: int SA_0_SOD SA_0_SOD
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 7
Description: Analog out BNC1: 0= X1, 1= X2, ..., 7= X8.

Parameter: int SA_0_0V SA_0_0V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Analog out BNC1: Specifies the value which should output 0 V.

Parameter: int SA_0_10V SA_0_10V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Analog out BNC1: Specifies the value which should output 10 V.

Parameter: int SA_1_SOD SA_1_SOD
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 7
Description: Analog out BNC2: 0= X1, 1= X2, ..., 7= X8.

Parameter: int SA_1_0V SA_1_0V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Analog out BNC2: Specifies the value which should output 0 V.

Parameter: int SA_1_10V SA_1_10V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 32767
Description: Analog out BNC2: Specifies the value which should output 10 V.

M.36 Save_Setup (SSU)

Save the current setup of controller to flash.

M.37 Get_CCD (CCD)

Returns current CCD data.

Parameter: String (binary, with 0x00) SA_CCD SA_CCD
Direction: Up
Valid values:
4096 bytes
convertable to 2048 WORDS.
Description: Raw CCD line

M.38 End_CCD

Stop reading CCD line, continue sending normal sensor values.

M.39 Get_Memory (DEB)

Returns a memory range of controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Offset SP_Offset
Direction: Down
Valid values:
Minimum: 0
Maximum: INT_MAX (2147483647)
Description: Memory offset

Parameter: int SP_Length SP_Length
Direction: Down
Valid values:
Minimum: 1
Maximum: 2048
Description: Memory length

Parameter: String (binary, with 0x00) SA_DEB SA_DEB
Direction: Up
Valid values:
SP_Length bytes
convertable to SP_Length/2 WORDS.
Description: Memory dump.

M.40 Upload_CalibTable

Send a calibration table for selected sensor to the controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SP_CalibTable SP_CalibTable

Direction: Down

Valid values:

1024 values of datatype float (4 byte).

Description: Calibration table for selected sensor.

Parameter: double SP_FullRange SP_FullRange

Direction: Down

Unit: μm

Valid values:

Minimum: 1

Maximum: 1000000

Description: Measure range of selected sensor.

M.41 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_RefractIndex IP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Tells the driver the refractive index used by the controller for thickness measurement.

Parameter: int IP_MeasureMode IP_MeasureMode

Direction: Down

Valid values:

0= Distance

1= Thickness

Description: Tells the driver the measure mode of the controller.

Parameter: int IP_Averaging IP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Tells the driver the averaging used by the controller. It is used for timeout check.

Parameter: int IP_SpectralAv

IP_SpectralAv

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Tells the driver the spectral averaging used by the controller. It is used for timeout check.

Parameter: int IP_X1..8

IP_X1..8

Direction: Down

Valid values:

0= off

1= RS232

Description: Tells the driver the selection of data used by the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

M.42 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_RefractIndex

IA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index of sensor assumed by the driver for thickness measurement.

Parameter: int IA_MeasureMode

IA_MeasureMode

Direction: Up

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor assumed by the driver.

Parameter: int IA_Averaging Direction: Up Valid values: Minimum: 1 Maximum: 999 Description: Averaging of the sensor assumed by the driver. It is used for timeout check.	IA_Averaging
Parameter: int IA_SpectralAv Direction: Up Valid values: Minimum: 1 Maximum: 999 Description: Spectral averaging of the controller assumed by the driver. It is used for timeout check.	IA_SpectralAv
Parameter: int IA_X1..8 Direction: Up Valid values: 0= off 1= RS232 Description: Selection of data of the sensor assumed by the driver. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used). X5 means (not used / Intensity1). X6 means (not used / Intensity2). X7 means (Barycenter / Barycenter1). X8 means (not used / Barycenter2).	IA_X1..8

N Commands for IFD2401

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

DriverX (native).

IF2008 (native).

N.1 Set_ActiveSensor (SEN)

Set the active sensor (up to 20 sensors can be stored in controller) for measurement. After this command, the internal range is resetted because the old one is no longer valid and the new one is not known until command Get_Range or Get_Status is called.

Parameter: int SP_Sensor

SP_Sensor

Direction: Down

Valid values:

Minimum: 0

Maximum: 19

Description: Number of active sensor.

N.2 Get_ActiveSensor (SEN?)

Get the active sensor.

Parameter: int SA_Sensor

SA_Sensor

Direction: Up

Valid values:

Minimum: 0

Maximum: 19

Description: Number of active sensor.

N.3 Get_Range (SCA)

Get the range of active sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: μm

Description: Range of active sensor.

N.4 Get_AllRanges (LUL)

Get the ranges of all calibrated sensors.

Parameter: String SA_Ranges

SA_Ranges

Direction: Up

Unit: μm (after conversion from string to int or double)

Description: Ranges of all 20 sensors in a string, separated by commas.

N.5 Acquire_DarkSig (DRK)

Acquire dark signal.

Parameter: int SA_MinSRIndex

SA_MinSRIndex

Direction: Up

Valid values:

1 = 100 Hz

2 = 200 Hz

3 = 400 Hz

4 = 1000 Hz

5 = 2000 Hz

Description: Minimal samplerate index.

N.6 FastDark (FDK)

Acquire dark signal for active sensor and samplerate.

Parameter: int SP_AveragingForDark

SP_AveragingForDark

Direction: Down

Valid values:

Minimum: 1

Maximum: 99

Description: Averaging factor for dark.

Parameter: int SP_Weighting

SP_Weighting

Direction: Down

Valid values:

Minimum: 1

Maximum: 100

Description: Weighting factor.

N.7 Set_AutoDark (ADK)

Enables/Disables the "Automatic Fast Dark" mode.

Parameter: int SP_AutoDark

SP_AutoDark

Direction: Down

Valid values:

0 = off

1 = on

Description: Automatic Fast Dark.

N.8 Get_AutoDark (ADK?)

Get the state of the "Automatic Fast Dark" mode.

Parameter: int SA_AutoDark

SA_AutoDark

Direction: Up

Valid values:

0= off

1= on

Description: Automatic Fast Dark.

N.9 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

0= Free samplerate

1= 100 Hz

2= 200 Hz

3= 400 Hz

4= 1000 Hz

5= 2000 Hz

Description: Samplerate index.

N.10 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= Free samplerate

1= 100 Hz

2= 200 Hz

3= 400 Hz

4= 1000 Hz

5= 2000 Hz

Description: Samplerate index.

N.11 Set_FreeSR (FRQ)

Set free samplerate value for data acquisition.

Parameter: int SP_FreeSR

SP_FreeSR

Direction: Down

Valid values:

Minimum: 100

Maximum: 2000

Unit: Hz

Description: Free Frequency.

Parameter: int SA_FreeSR
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 2000
Unit: Hz
Description: Free Samplerate.

SA_FreeSR

N.12 Get_FreeSR (FRQ?)

Get free samplerate value.

Parameter: int SA_FreeSR
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 2000
Unit: Hz
Description: Free Samplerate.

SA_FreeSR

N.13 Set_Exposure (TEX)

Set free exposure time for data acquisition.

Parameter: int SP_Exposure
Direction: Down
Valid values:
 Minimum: 500
 Maximum: 10000
Unit: μs
Description: Free Exposure time.

SP_Exposure

Parameter: int SA_Exposure
Direction: Up
Valid values:
 Minimum: 500
 Maximum: 10000
Unit: μs
Description: Free Exposure time.

SA_Exposure

N.14 Get_Exposure (TEX?)

Get free exposure time.

Parameter: int SA_Exposure
Direction: Up
Valid values:
 Minimum: 500
 Maximum: 10000
Unit: μs
Description: Free Exposure time.

SA_Exposure

N.15 Get_MinSR (FRM)

Get the minimum authorized samplerate.

Parameter: int SA_MinSR

SA_MinSR

Direction: Up

Valid values:

Minimum: 100

Maximum: 2000

Unit: Hz

Description: Minimum samplerate (determined by dark signal).

N.16 Set_DoubleFreq (DFA)

Enables or disables the double frequency mode.

Parameter: int SP_DoubleFrequency

SP_DoubleFrequency

Direction: Down

Valid values:

0 = Disable

1 = Enable

Description: Double frequency mode.

N.17 Get_DoubleFreq (DFA?)

Get the double frequency mode.

Parameter: int SA_DoubleFrequency

SA_DoubleFrequency

Direction: Up

Valid values:

0 = Disable

1 = Enable

Description: Double frequency mode.

N.18 Set_Frequencies (DFF)

Set the frequencies for double frequency mode.

Parameter: int SP_LowFrequency

SP_LowFrequency

Direction: Down

Valid values:

Minimum: 100

Maximum: 1850

Unit: Hz

Description: Low frequency.

Parameter: int SP_HighFrequency
Direction: Down
Valid values:
 Minimum: 100
 Maximum: 1850
Unit: Hz
Description: High frequency.

SP_HighFrequency

N.19 Get_Frequencies (DFF?)

Get the frequencies of double frequency mode.

Parameter: int SA_LowFrequency
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 1850
Unit: Hz
Description: Low frequency.

SA_LowFrequency

Parameter: int SA_HighFrequency
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 1850
Unit: Hz
Description: High frequency.

SA_HighFrequency

N.20 Set_IntensityMode (DFI)

Set normalized or raw intensity to transmit.

Parameter: int SP_TransmitIntensity
Direction: Down
Valid values:
 0 = Normalized
 1 = Raw
Description: Intensity mode.

SP_TransmitIntensity

N.21 Get_IntensityMode (DFI?)

Get if normalized or raw intensity is transmitted.

Parameter: int SA_TransmitIntensity
Direction: Up
Valid values:
 0 = Normalized
 1 = Raw
Description: Intensity mode.

SA_TransmitIntensity

N.22 Set_MeasureMode (MOD)

Set the measure mode.

Parameter: int SP_MeasureMode

SP_MeasureMode

Direction: Down

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the sensor.

N.23 Get_MeasureMode (MOD?)

Get the current measure mode.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the sensor.

N.24 Set_RefractIndex (SRI)

Set the refraction index for thickness measurement.

Parameter: double SP_RefractIndex

SP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

N.25 Get_RefractIndex (SRI?)

Get the current refraction index.

Parameter: double SA_RefractIndex

SA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

N.26 Set_Threshold (MNP)

Set the detection threshold for distance mode.

Parameter: double SP_Threshold

SP_Threshold

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.27 Get_Threshold (MNP?)

Get the current detection threshold for distance mode.

Parameter: double SA_Threshold

SA_Threshold

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.28 Set_HoldLastValid (HLV)

Enable/Disable the "Hold Last Value" mode.

Parameter: int SP_LastValid

SP_LastValid

Direction: Down

Valid values:

Minimum: 0

Maximum: 999

Description: Max number of points to hold.

N.29 Get_HoldLastValid (HLV?)

Get the current "Hold Last Value" mode.

Parameter: int SA_LastValid

SA_LastValid

Direction: Up

Valid values:

Minimum: 0

Maximum: 999

Description: Max number of points to hold.

N.30 Set_Threshold1 (SPP)

Set the detection threshold for the strongest peak in thickness mode.

Parameter: double SP_Threshold1

SP_Threshold1

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.31 Get_Threshold1 (SPP?)

Get the current detection threshold for the strongest peak in thickness mode.

Parameter: double SA_Threshold1

SA_Threshold1

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.32 Set_Threshold2 (SDP)

Set the detection threshold for the second peak in thickness mode.

Parameter: double SP_Threshold2

SP_Threshold2

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.33 Get_Threshold2 (SDP?)

Get the current detection threshold for the second peak in thickness mode.

Parameter: double SA_Threshold2

SA_Threshold2

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

N.34 Set_FirstPeakMode (MSP)

Enable/Disable the "First peak" mode.

Parameter: int SP_FirstPeak

SP_FirstPeak

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: First peak state.

N.35 Get_FirstPeakMode (MSP?)

Get the current "First peak" mode.

Parameter: int SA_FirstPeak

SA_FirstPeak

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: First peak state.

N.36 Set_DataScale (CEE)

Set the scaling factor for X1, X2 and X3 in thickness mode. This is an internal command. It should not be used by the customer.

Parameter: double SP_DataScale

SP_DataScale

Direction: Down

Valid values:

Minimum: 1.0

Maximum: 5.0

Description: Scaling factor of the sensor.

N.37 Get_DataScale (CEE?)

Get the scaling factor for X1, X2 and X3 in thickness mode.

Parameter: double SA_DataScale

SA_DataScale

Direction: Up

Valid values:

Minimum: 1.0

Maximum: 5.0

Description: Scaling factor of the sensor.

N.38 Set_BarycenterSca (CEB)

Set the scaling factor for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SP_BarycenterScale

SP_BarycenterScale

Direction: Down

Valid values:

Minimum: 32.0

Maximum: 32.0

Description: Scaling factor for Barycenter.

N.39 Get_BarycenterSca (CEB?)

Get the scaling factor for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SA_BarycenterScale

SA_BarycenterScale

Direction: Up

Valid values:

Minimum: 32.0

Maximum: 32.0

Description: Scaling factor for Barycenter.

N.40 Set_BarycenterOff (CRB)

Set the offset values for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SP_BarycenterOffset

SP_BarycenterOffset

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1023.9

Description: Offset for Barycenter.

N.41 Get_BarycenterOff (CRB?)

Get the offset value for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SA_BarycenterOffset

SA_BarycenterOffset

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1023.9

Description: Offset for Barycenter.

N.42 Set_LampTest (SLP)

Enable/Disable the "Lamp Test" mode.

Parameter: int SP_LampTest

SP_LampTest

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Lamp Test state.

N.43 Get_LampTest (SLP?)

Get the current "Lamp Test" mode.

Parameter: int SA_LampTest

SA_LampTest

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: Lamp Test state.

N.44 Set_LampTestThr (CSL)

Set the threshold for "Lamp Test" mode.

Parameter: int SP_LampTestThr

SP_LampTestThr

Direction: Down

Valid values:

Minimum: 0

Maximum: 998

Description: Lamp Test threshold.

N.45 Get_LampTestThr (CSL?)

Get the threshold for "Lamp Test" mode.

Parameter: int SA_LampTestThr

SA_LampTestThr

Direction: Up

Valid values:

Minimum: 0

Maximum: 998

Description: Lamp Test threshold.

N.46 Set_LEDIntensity (LED)

Set the LED intensity level.

Parameter: int SP_LEDIntensity
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 100
Description: LED intensity.

SP_LEDIntensity

N.47 Get_LEDIntensity (LED?)

Get the LED intensity level.

Parameter: int SA_LEDIntensity
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 100
Description: LED intensity.

SA_LEDIntensity

N.48 Set_AutoAdaptLED (AAL)

Enable/Disable the "Auto-Adaptive LED" mode.

Parameter: int SP_AutoAdaptLED
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 1
Description: Auto-Adaptive LED state.

SP_AutoAdaptLED

N.49 Get_AutoAdaptLED (AAL?)

Get the "Auto-Adaptive LED" mode.

Parameter: int SA_AutoAdaptLED
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 1
Description: Auto-Adaptive LED state.

SA_AutoAdaptLED

N.50 Set_AdaptLEDThr (VTH)

Set the threshold value for the auto-adaptive LED mode.

Parameter: int SP_AutoAdaptLEDThr

SP_AutoAdaptLEDThr

Direction: Down

Valid values:

Minimum: 0

Maximum: 4095

Description: Auto-Adaptive LED threshold.

N.51 Get_AdaptLEDThr (VTH?)

Get the threshold value for the auto-adaptive LED mode.

Parameter: int SA_AutoAdaptLEDThr

SA_AutoAdaptLEDThr

Direction: Up

Valid values:

Minimum: 0

Maximum: 4095

Description: Auto-Adaptive LED threshold.

N.52 Set_IntLightSrc (CCL)

Set the state of the internal light source (LED).

Parameter: int SP_LED_Off

SP_LED_Off

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: LED on (0) or off (1).

N.53 Get_IntLightSrc (CCL?)

Get the state of the internal light source (LED).

Parameter: int SA_LED_Off

SA_LED_Off

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: LED on (0) or off (1).

N.54 Set_RefrIdxFile (INF)

Set the refractive index file.

Parameter: int SP_RefractIndexFileIdx	SP_RefractIndexFileIdx
Direction: Down	
Valid values:	
Minimum: 0	
Maximum: 8	
Description: Refractive index file index.	
Parameter: int SA_RefractIndexFileIdx	SA_RefractIndexFileIdx
Direction: Up	
Valid values:	
Minimum: 0	
Maximum: 8	
Description: Refractive index file index.	
Parameter: String SA_RefractIndexFileName	SA_RefractIndexFileName
Direction: Up	
Description: Refractive index file name.	
Parameter: double SA_MinRefractIndex	SA_MinRefractIndex
Direction: Up	
Valid values:	
Minimum: 0.0001	
Maximum: 9.9999	
Description: Minimum refractive index.	
Parameter: double SA_MaxRefractIndex	SA_MaxRefractIndex
Direction: Up	
Valid values:	
Minimum: 0.0001	
Maximum: 9.9999	
Description: Maximum refractive index.	
Parameter: double SA_AvgRefractIndex	SA_AvgRefractIndex
Direction: Up	
Valid values:	
Minimum: 0.0001	
Maximum: 9.9999	
Description: Averaging refractive index.	

N.55 Get_RefrIdxFile (INF?)

Request the selected refractive index file.

Parameter: int SA_RefractIndexFileIdx	SA_RefractIndexFileIdx
Direction: Up	
Valid values:	
Minimum: 0	
Maximum: 8	
Description: Refractive index file index.	

Parameter: String SA_RefractIndexFileName Direction: Up Description: Refractive index file name.	SA_RefractIndexFileName
Parameter: double SA_MinRefractIndex Direction: Up Valid values: Minimum: 0.0001 Maximum: 9.9999 Description: Minimum refractive index.	SA_MinRefractIndex
Parameter: double SA_MaxRefractIndex Direction: Up Valid values: Minimum: 0.0001 Maximum: 9.9999 Description: Maximum refractive index.	SA_MaxRefractIndex
Parameter: double SA_AvgRefractIndex Direction: Up Valid values: Minimum: 0.0001 Maximum: 9.9999 Description: Averaging refractive index.	SA_AvgRefractIndex

N.56 Upload_RefrIdxFile

Send a refractive index file to the controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_RefractIndexFileIdx Direction: Down Valid values: Minimum: 1 Maximum: 8 Description: Index on which position the file should be stored.	SP_RefractIndexFileIdx
Parameter: String SP_RefractIndexFile Direction: Down Description: Refractive index file with 2049 lines. First line is a name, the other lines are refractive index values with precision four.	SP_RefractIndexFile

N.57 RecenterEncoder (RCD)

Recenter encoder position.

Parameter: int SP_Encoder1 Direction: Down Valid values: Minimum: 0 Maximum: 1 Description: Recenter Encoder 1.	SP_Encoder1
--	-------------

Parameter: int SP_Encoder2

SP_Encoder2

Direction: Down**Valid values:****Minimum:** 0**Maximum:** 1**Description:** Recenter Encoder 2.**Parameter:** int SP_Encoder3

SP_Encoder3

Direction: Down**Valid values:****Minimum:** 0**Maximum:** 1**Description:** Recenter Encoder 3.

N.58 Set_MissingSignal (RSP)

Set behaviour for missing second peak in thickness mode.

Parameter: int SP_Option

SP_Option

Direction: Down**Valid values:****Minimum:** 0**Maximum:** 1**Description:** Mode for behaviour.

N.59 Get_MissingSignal (RSP?)

Returns behaviour for missing second peak in thickness mode.

Parameter: int SA_Option

SA_Option

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 1**Description:** Mode for behaviour.

N.60 Set_Reverse (RVS)

Reverse the distance signal.

Parameter: int SP_Reverse

SP_Reverse

Direction: Down**Valid values:**

0 = Normal direction

1 = Reverse direction

Description: In reverse mode, measure range - distance is transmitted.

N.61 Get_Reverse (RVS?)

Returns setting for reverse mode.

Parameter: int SA_Reverse

SA_Reverse

Direction: Up

Valid values:

0 = Normal direction

1 = Reverse direction

Description: Reverse mode

N.62 Set_Averaging (AVR)

Set data averaging at controller. Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 9999

Description: Data averaging factor.

N.63 Get_Averaging (AVR?)

Returns current data averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

N.64 Set_SpectralAv (AVS)

Set spectral averaging at controller.

Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 9999

Description: Data averaging factor.

N.65 Get_SpectralAv (AVS?)

Returns current spectral averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

N.66 Continue (CTN)

Continues data acquisition (after a trigger commands SingleShot_Trg, TriggerMode_Edge, TriggerMode_State or an error).

N.67 Start_Trigger (TRG)

Starts the trigger mode at controller.

N.68 End_Trigger

Stops the trigger mode at controller (after Start_Trigger).

N.69 SingleShot_Trg (TRE)

Starts the single shot trigger mode at controller.

Parameter: int SP_NumberOfPoints

SP_NumberOfPoints

Direction: Down

Valid values:

Minimum: 0

Maximum: 9999

Description: Number of values to read.

N.70 Set_TrgMode_Edge (TRS)

Enable/Disable the "Start/Stop on Edge Trigger" mode.

Parameter: int SP_TriggerMode_Edge

SP_TriggerMode_Edge

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Start/Stop on Edge Trigger state.

N.71 Set_TrgMode_State (TRN)

Enable/Disable the "Start/Stop on State Trigger" mode.

Parameter: int SP_TriggerMode_State

SP_TriggerMode_State

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Start/Stop on State Trigger state.

N.72 Set_ActiveEdge (TRF)

Set the active edge or state for Start_Trigger, TriggerMode_Edge or TriggerMode_State.

Parameter: int SP_ActiveEdge

SP_ActiveEdge

Direction: Down

Valid values:

0= rising edge or high state

1= falling edge or low state

Description: Active state or edge.

N.73 Get_ActiveEdge (TRF?)

Get the active edge or state for Start_Trigger, TriggerMode_Edge or TriggerMode_State.

Parameter: int SA_ActiveEdge

SA_ActiveEdge

Direction: Up

Valid values:

0= rising edge or high state

1= falling edge or low state

Description: Active state or edge.

N.74 Software_Trigger (STR)

Simulates an hardware trigger (SYNC IN) for trigger commands SingleShot_Trg, TriggerMode_Edge or TriggerMode_State.

N.75 Set_Watchdog (WDE)

Enable/Disable the Watchdog.

Parameter: int SP_Watchdog

SP_Watchdog

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Watchdog state.

N.76 Get_Watchdog (WDE?)

Get the Watchdog state.

Parameter: int SA_Watchdog

SA_Watchdog

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: Watchdog state.

N.77 Set_WatchdogPrd (WDP)

Set the Watchdog period.

Parameter: int SP_WatchdogPeriod

SP_WatchdogPeriod

Direction: Down

Unit: s

Valid values:

Minimum: 10

Maximum: 255

Description: Watchdog period.

N.78 Get_WatchdogPrd (WDP?)

Get the Watchdog period.

Parameter: int SA_WatchdogPeriod

SA_WatchdogPeriod

Direction: Up

Unit: s

Valid values:

Minimum: 10

Maximum: 255

Description: Watchdog period.

N.79 Get_Status (STS)

Retrieve detailed information about the controller and sensor.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= Free samplerate

1= 100 Hz

2= 200 Hz

3= 400 Hz

4= 1000 Hz

5= 2000 Hz

Description: Samplerate index.

Parameter: int SA_MeasureMode Direction: Up Valid values: 0= Distance 1= Thickness Description: Measure mode of the sensor.	SA_MeasureMode
Parameter: int SA_Sensor Direction: Up Valid values: Minimum: 0 Maximum: 19 Description: Number of active sensor.	SA_Sensor
Parameter: int SA_ASCII Direction: Up Description: Data Transfer mode (values) of sensor.	SA_ASCII
Parameter: int SA_Averaging Direction: Up Valid values: Minimum: 1 Maximum: 999 Description: Data averaging factor.	SA_Averaging
Parameter: int SA_X1..16 Direction: Up Valid values: 0= off 1= RS232/RS422 9= USB Description: Selection of data of the sensor. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used). X5 means (not used / Intensity1). X6 means (Counter / Intensity2). X7 means (Barycenter / Barycenter1). X8 means (not used / Barycenter2). X9 means (State Flags / State Flags). X10 means (not used / Counter). X11 means (Encoder 1 LSB / Encoder 1 LSB). X12 means (Encoder 1 MSB / Encoder 1 MSB). X13 means (Encoder 2 LSB / Encoder 2 LSB). X14 means (Encoder 2 MSB / Encoder 2 MSB). X15 means (Encoder 3 LSB / Encoder 3 LSB). X16 means (Encoder 3 MSB / Encoder 3 MSB).	SA_X1..16

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: μm

Description: Range of active sensor.

N.80 Get_Version (VER)

Get the software version of the controller.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Software version of sensor.

N.81 Set_OutputData (SOD)

Select the data to be output from controller.

Parameter: int SP_X1..16

SP_X1..16

Direction: Down

Valid values:

0= off

1= RS232/RS422

9= USB

Description: Selection of data of the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

N.82 Get_OutputData (SOD?)

Get the data output from controller.

Parameter: int SA_X1..16

SA_X1..16

Direction: Up

Valid values:

0= off

1= RS232/RS422

9= USB

Description: Selection of data of the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

N.83 Set_Ascii (ASC)

Set digital data transfer (only values, no sensor answer) to ASCII mode.

N.84 Set_Binary (BIN)

Set digital data transfer (only values, no sensor answer) to binary mode.

N.85 Set_AnalogOut (ANA)

Setup the analog outputs.

Parameter: int SP_OutNr

SP_OutNr

Direction: Down

Valid values:

0= BNC1

1= BNC2

Description: Number of analog output.

Parameter: int SP_SOD Direction: Down Valid values: Minimum: 0 Maximum: 7 Description: Specifies the number (X1..X16) to output at analog out.	SP_SOD
Parameter: int SP_0V Direction: Down Valid values: Minimum: 0 Maximum: 99999 Description: Specifies the value which should output 0 V.	SP_0V
Parameter: int SP_10V Direction: Down Valid values: Minimum: 0 Maximum: 99999 Description: Specifies the value which should output 10 V.	SP_10V

N.86 Get_AnalogOut (ANA?)

Get settings of analog outputs.

Parameter: int SA_0_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC1: 0= X1, 1= X2, ..., 15= X16.	SA_0_SOD
Parameter: int SA_0_0V Direction: Up Valid values: Minimum: 0 Maximum: 99999 Description: Analog out BNC1: Specifies the value which should output 0 V.	SA_0_0V
Parameter: int SA_0_10V Direction: Up Valid values: Minimum: 0 Maximum: 99999 Description: Analog out BNC1: Specifies the value which should output 10 V.	SA_0_10V
Parameter: int SA_1_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC2: 0= X1, 1= X2, ..., 15= X16.	SA_1_SOD

Parameter: int SA_1_0V SA_1_0V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 99999
Description: Analog out BNC2: Specifies the value which should output 0 V.

Parameter: int SA_1_10V SA_1_10V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 99999
Description: Analog out BNC2: Specifies the value which should output 10 V.

N.87 Set_AnalogZero (SOF)

Set analog output to zero.

Parameter: int SP_Zero SP_Zero
Direction: Down
Valid values:
 0 = normal
 1 = set
Description: Set/reset the analog output 0V value.

N.88 Set_Baudrate (BAU)

Set the baudrate of the serial interface of controller.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled after changing baudrate for a short time (sensor command Start_Trigger).

Parameter: int SP_SensorBaudrate SP_SensorBaudrate
Direction: Down
Valid values:
 9600
 19200
 38400
 57600
 115200
 230400
 460800
Description: Baudrate of controller.

N.89 Get_Baudrate (BAU?)

Get the baudrate of the serial interface of controller.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

9600
19200
38400
57600
115200
230400
460800

Description: Baudrate of controller.

N.90 Save_Setup (SSU)

Save the current setup of controller to flash.

N.91 Reset (RST)

Reset the controller and set the default parameter values.

N.92 Get_CCD (CCD)

Returns current CCD data.

Parameter: String (binary, with 0x00) SA_CCD

SA_CCD

Direction: Up

Valid values:

4096 bytes
convertable to 2048 WORDS.

Description: Raw CCD line.

Parameter: String (binary, with 0x00) SA_PreTreated

SA_PreTreated

Direction: Up

Valid values:

4096 bytes
convertable to 2048 WORDS.

Description: Raw CCD line.

N.93 Get_DarkSig (SGD)

Get the dark signal table of controller.

Parameter: String (binary, with 0x00) SA_DarkSig

SA_DarkSig

Direction: Up

Valid values:

4096 bytes
convertable to 2048 WORDS.

Description: Dark signal table.

N.94 Get_WhiteRef (SGW)

Get the white reference table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_WhiteRef

SA_WhiteRef

Direction: Up

Valid values:

4096 bytes

convertable to 2048 WORDS.

Description: White reference table.

N.95 Get_NormSig (SGN)

Get the norm signal table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_NormSig

SA_NormSig

Direction: Up

Valid values:

4096 bytes

convertable to 2048 WORDS.

Description: Norm signal table.

N.96 Get_CalibTable (SGC)

Get the calibration table of controller.

Parameter: String (binary, with 0x00) SA_CalibTable

SA_CalibTable

Direction: Up

Valid values:

4096 bytes

convertable to 2048 WORDS.

Description: Calibration table.

N.97 Start_Spectrum

Start the spectrum mode in controller. Only work via USB connection.
On firmware versions below 1.2.36 the spectrum is shifted 8 bytes to left.

Parameter: int IP_UsbReadBufSize

IP_UsbReadBufSize

Direction: Down

Unit: Bytes

Valid values:

Minimum: 512

Maximum: 65536

Default: 8192

Description: Buffer size for read operations on USB, while spectrum mode is active. The value is always ceiled to the next power of two (512, 1024, 2048, ..., 32768, 65536).

Parameter: int IP_TimerResolution

IP_TimerResolution

Direction: Down

Unit: ms

Valid values:

-1= Do not set timer resolution.

0= Use greatest possible accuracy.

1..255= Resolution in milliseconds.

Unit: ms

Default: 0

Description: Timer resolution (for Windows scheduler, set by timeBeginPeriod).

It is automatically reset at End_Spectrum.

N.98 Get_Spectrum

Read one Spectrum (PreTreated Signal). It is returned with a frequency of Samplerate divided by Spectral Averaging.

Parameter: int SP_WaitSpectrumTimeout

SP_WaitSpectrumTimeout

Direction: Down

Unit: ms

Valid values:

Minimum: 0

Maximum: 1000000000

Description: Timeout to wait for a spectrum.

Parameter: int SP_ReadMode

SP_ReadMode

Direction: Down

Valid values:

0= Each spectrum

1= Only newest spectrum

2= Automatic

Description: This mode specifies if each spectrum should be read or only the latest one. If set to automatic each spectrum is read until the buffer does not overflow. If the buffer becomes full one or more spectra are discarded.

Parameter: String (binary, with 0x00) SA_Spectrum

SA_Spectrum

Direction: Up

Valid values:

4096 bytes

convertable to 2048 WORDS.

Description: PreTreated Signal.

Parameter: double SA_Timestamp

SA_Timestamp

Direction: Up

Valid values:

Minimum: 0

Unit: ms

Description: Timestamp of the signal. It starts from 1970 Jan 01 at 01:00. It is generated when the spectrum is read from USB

Parameter: int SA_SkippedSpectra SA_SkippedSpectra
Direction: Up
Valid values:
 Minimum: 0
 Maximum: INT_MAX (2147483647)
Description: Number of skipped spectra, if SP_NewestSpectrum is set to 1.

N.99 End_Spectrum

End the spectrum mode in controller.

Parameter: int IP_UsbReadBufSize IP_UsbReadBufSize
Direction: Down
Unit: Bytes
Valid values:
 Minimum: 512
 Maximum: 65536
Default: 512
Description: Buffer size for read operations on USB, after spectrum mode was finished. The value is always ceiled to the next power of two (512, 1024, 2048, ..., 32768, 65536).

N.100 Upload_CalibTable

Send a calibration table for selected sensor to the controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_BlockSize SP_BlockSize
Direction: Down
Unit: Values
Valid values:
 Minimum: 1
 Maximum: 999
Default: 64
Description: Size of blocks in which the calibration table is separated before sending.

Parameter: String (binary, with 0x00) SP_CalibTable SP_CalibTable
Direction: Down
Valid values:
 2048 values of datatype float (4 byte).
Description: Calibration table for selected sensor.

Parameter: double SP_FullRange SP_FullRange
Direction: Down
Unit: μm
Valid values:
 Minimum: 1
 Maximum: 1000000
Description: Measure range of selected sensor.

N.101 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_RefractIndex Direction: Down Valid values: Minimum: 0.001 Maximum: 9.999 Description: Tells the driver the refractive index used by the controller for thickness measurement.	IP_RefractIndex
Parameter: int IP_MeasureMode Direction: Down Valid values: 0= Distance 1= Thickness Description: Tells the driver the measure mode of the controller.	IP_MeasureMode
Parameter: int IP_Averaging Direction: Down Valid values: Minimum: 1 Maximum: 999 Description: Tells the driver the averaging used by the controller. It is used for timeout check.	IP_Averaging
Parameter: int IP_SpectralAv Direction: Down Valid values: Minimum: 1 Maximum: 999 Description: Tells the driver the spectral averaging used by the controller. It is used for timeout check.	IP_SpectralAv
Parameter: int IP_X1..16 Direction: Down Valid values: 0= off 1= RS232 9= USB Description: Tells the driver the selection of data used by the sensor. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used).	IP_X1..16

X5 means (not used / Intensity1).
 X6 means (Counter / Intensity2).
 X7 means (Barycenter / Barycenter1).
 X8 means (not used / Barycenter2).
 X9 means (State Flags / State Flags).
 X10 means (not used / Counter).
 X11 means (Encoder 1 LSB / Encoder 1 LSB).
 X12 means (Encoder 1 MSB / Encoder 1 MSB).
 X13 means (Encoder 2 LSB / Encoder 2 LSB).
 X14 means (Encoder 2 MSB / Encoder 2 MSB).
 X15 means (Encoder 3 LSB / Encoder 3 LSB).
 X16 means (Encoder 3 MSB / Encoder 3 MSB).

N.102 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_RefractIndex

IA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index of sensor assumed by the driver for thickness measurement.

Parameter: int IA_MeasureMode

IA_MeasureMode

Direction: Up

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor assumed by the driver.

Parameter: int IA_Averaging

IA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Averaging of the sensor assumed by the driver. It is used for timeout check.

Parameter: int IA_SpectralAv

IA_SpectralAv

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Spectral averaging of the controller assumed by the driver. It is used for timeout check.

Parameter: int IA_X1..16

IA_X1..16

Direction: Up

Valid values:

0= off

1= RS232

9= USB

Description: Selection of data of the sensor assumed by the driver. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

O Commands for IFD2430

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (native).

O.1 Set_ActiveSensor (SEN)

Set the active sensor (up to 20 sensors can be stored in controller) for measurement. After this command, the internal range is resetted because the old one is no longer valid and the new one is not known until command Get_Range or Get_Status is called.

Parameter: int SP_Sensor	SP_Sensor
Direction: Down	
Valid values:	
Minimum: 0	
Maximum: 19	
Description: Number of active sensor.	

O.2 Get_ActiveSensor (SEN?)

Get the active sensor.

Parameter: int SA_Sensor	SA_Sensor
Direction: Up	
Valid values:	
Minimum: 0	
Maximum: 19	
Description: Number of active sensor.	

O.3 Get_Range (SCA)

Get the range of active sensor.

Parameter: double SA_Range	SA_Range
Direction: Up	
Unit: μm	
Description: Range of active sensor.	

O.4 Acquire_DarkSig (DRK)

Acquire dark signal.

Parameter: int SA_MinSRIndex

SA_MinSRIndex

Direction: Up

Valid values:

- 1 = 100 Hz
- 2 = 200 Hz
- 3 = 400 Hz
- 4 = 1000 Hz
- 5 = 2000 Hz
- 6 = 4000 Hz
- 7 = 6250 Hz
- 8 = 10000 Hz
- 9 = 15625 Hz
- 10 = 20000 Hz
- 11 = 25000 Hz
- 12 = 31250 Hz

Description: Minimal samplerate index.

O.5 FastDark (FDK)

Acquire dark signal for active sensor and samplerate.

Parameter: int SP_AveragingForDark

SP_AveragingForDark

Direction: Down

Valid values:

Minimum: 1

Maximum: 99

Description: Averaging factor for dark.

Parameter: int SP_Weighting

SP_Weighting

Direction: Down

Valid values:

Minimum: 1

Maximum: 32767

Description: Weighting factor.

O.6 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

- 1 = 100 Hz
- 2 = 200 Hz
- 3 = 400 Hz
- 4 = 1000 Hz
- 5 = 2000 Hz

6= 4000 Hz
7= 6250 Hz
8= 10000 Hz
9= 15625 Hz
10= 20000 Hz
11= 2500 Hz
12= 31250 Hz

Description: Samplerate index.

O.7 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

1= 100 Hz
2= 200 Hz
3= 400 Hz
4= 1000 Hz
5= 2000 Hz
6= 4000 Hz
7= 6250 Hz
8= 10000 Hz
9= 15625 Hz
10= 20000 Hz
11= 2500 Hz
12= 31250 Hz

Description: Samplerate index.

O.8 Set_FreeSR (FRQ)

Set free samplerate value for data acquisition.

Parameter: int SP_FreeSR

SP_FreeSR

Direction: Down

Valid values:

Minimum: 100

Maximum: 31250

Unit: Hz

Description: Free Frequency.

Parameter: int SA_FreeSR

SA_FreeSR

Direction: Up

Valid values:

Minimum: 100

Maximum: 31250

Unit: Hz

Description: Free Samplerate.

O.9 Get_FreeSR (FRQ?)

Get free samplerate value.

Parameter: int SA_FreeSR

SA_FreeSR

Direction: Up

Valid values:

Minimum: 100

Maximum: 31250

Unit: Hz

Description: Free Samplerate.

O.10 Set_Exposure (TEX)

Set free exposure time for data acquisition.

Parameter: int SP_Exposure

SP_Exposure

Direction: Down

Valid values:

Minimum: 32

Maximum: 10000

Unit: μs

Description: Free Exposure time.

Parameter: int SA_Exposure

SA_Exposure

Direction: Up

Valid values:

Minimum: 32

Maximum: 10000

Unit: μs

Description: Free Exposure time.

O.11 Get_Exposure (TEX?)

Get free exposure time.

Parameter: int SA_Exposure

SA_Exposure

Direction: Up

Valid values:

Minimum: 32

Maximum: 10000

Unit: μs

Description: Free Exposure time.

O.12 Set_MeasureMode (MOD)

Set the measure mode.

Parameter: int SP_MeasureMode

SP_MeasureMode

Direction: Down

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor.

O.13 Get_MeasureMode (MOD?)

Get the current measure mode.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the sensor.

O.14 Set_RefractIndex (SRI)

Set the refraction index for thickness measurement.

Parameter: double SP_RefractIndex

SP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

O.15 Get_RefractIndex (SRI?)

Get the current refraction index.

Parameter: double SA_RefractIndex

SA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

O.16 Set_Threshold (MNP)

Set the detection threshold.

Parameter: double SP_Threshold

SP_Threshold

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

O.17 Get_Threshold (MNP?)

Get the current detection threshold.

Parameter: double SA_Threshold

SA_Threshold

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

O.18 Set_KeyLock (LOC)

Locks/Unlocks the keypad of controller.

Parameter: int SP_Keylock

SP_Keylock

Direction: Down

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

O.19 Get_KeyLock (LOC?)

Returns the lock state of controller keypad.

Parameter: int SA_Keylock

SA_Keylock

Direction: Up

Valid values:

0= off (keys enabled)

1= on (keys locked)

Description: The keypad state at the sensor.

O.20 Set_Averaging (AVR)

Set data averaging at controller. Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

O.21 Get_Averaging (AVR?)

Returns current data averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

O.22 Continue (CTN)

Continues data acquisition (after a trigger command or an error).

O.23 Start_Trigger (TRG)

Starts the trigger mode at controller.

O.24 End_Trigger

Stops the trigger mode at controller.

O.25 Get_Status (STS)

Retrieve detailed information about the controller and sensor.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

1 = 100 Hz

2 = 200 Hz

3 = 400 Hz

4 = 1000 Hz

5 = 2000 Hz

6 = 4000 Hz

7 = 6250 Hz

8 = 10000 Hz

9 = 15625 Hz

10 = 20000 Hz

11 = 2500 Hz

12 = 31250 Hz

Description: Samplerate index.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the sensor.

Parameter: int SA_Sensor Direction: Up Valid values: Minimum: 0 Maximum: 19 Description: Number of active sensor.	SA_Sensor
Parameter: int SA_ASCII Direction: Up Description: Data Transfer mode (values) of sensor.	SA_ASCII
Parameter: int SA_Averaging Direction: Up Valid values: Minimum: 1 Maximum: 999 Description: Data averaging factor.	SA_Averaging
Parameter: int SA_X1..8 Direction: Up Valid values: 0= off 1= RS232 9= TCP/IP Description: Selection of data of the sensor (Distance / Thickness). This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used). X5 means (not used / Intensity1). X6 means (not used / Intensity2). X7 means (Barycenter / Barycenter1). X8 means (not used / Barycenter2).	SA_X1..8
Parameter: double SA_Range Direction: Up Unit: μm Description: Range of active sensor.	SA_Range

O.26 Get_Version (VER)

Get the software version of the controller.

Parameter: String SA_Version Direction: Up Description: Software version of sensor.	SA_Version
--	------------

O.27 Set_OutputData (SOD)

Select the data to be output from controller.

Parameter: int SP_X1..8

SP_X1..8

Direction: Down

Valid values:

0= off

1= RS232

9= TCP/IP

Description: Selection of data of the sensor (Distance / Thickness). This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

O.28 Get_OutputData (SOD?)

Get the data output from controller.

Parameter: int SA_X1..8

SA_X1..8

Direction: Up

Valid values:

0= off

1= RS232

9= TCP/IP

Description: Selection of data of the sensor (Distance / Thickness). This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

O.29 Set_Ascii (ASC)

Set digital data transfer (only values, no sensor answer) to ASCII mode.

O.30 Set_Binary (BIN)

Set digital data transfer (only values, no sensor answer) to binary mode.

O.31 Set_IPAddress (IPA)

Set the IP-Adress of the ethernet interface of controller.

Parameter: String SP_IPAddress

SP_IPAddress

Direction: Down

Description: IP-Address of controller. The format must be xx.xx.xx.xx! It has to be change in any case because the default address is not valid!

O.32 Get_IPAddress (IPA?)

Get the IP-Adress of the ethernet interface of controller.

Parameter: String SA_IPAddress

SA_IPAddress

Direction: Up

Description: IP-Address of controller.

O.33 Set_Baudrate (BAU)

Set the baudrate of the serial interface of controller.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled after changing baudrate for a short time (sensor command Start_Trigger).

Parameter: int SP_SensorBaudrate

SP_SensorBaudrate

Direction: Down

Valid values:

9600
19200
38400
57600
115200

Description: Baudrate of controller.

O.34 Get_Baudrate (BAU?)

Get the baudrate of the serial interface of controller.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

9600
19200
38400
57600
115200

Description: Baudrate of controller.

O.35 Save_Setup (SSU)

Save the current setup of controller to flash.

O.36 Get_CCD (CCD)

Returns current CCD data.

Parameter: String (binary, with 0x00) SA_CCD SA_CCD
Direction: Up
Valid values:
4096 bytes
convertable to 2048 WORDS
Description: Raw CCD line.

Parameter: String (binary, with 0x00) SA_PreTreated SA_PreTreated
Direction: Up
Valid values:
4096 bytes
convertable to 2048 WORDS.
Description: Raw CCD line.

O.37 Get_DarkSig (SGD)

Get the dark signal table of controller.

Parameter: String (binary, with 0x00) SA_DarkSig SA_DarkSig
Direction: Up
Valid values:
2048 bytes
convertable to 1024 WORDS
Description: Dark signal table.

O.38 Get_WhiteRef (SGW)

Get the white reference table table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_WhiteRef SA_WhiteRef
Direction: Up
Valid values:
2048 bytes
convertable to 1024 WORDS
Description: White reference table.

O.39 Get_NormSig (SGN)

Get the norm signal table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_NormSig SA_NormSig
Direction: Up
Valid values:
2048 bytes
convertable to 1024 WORDS
Description: Norm signal table.

O.40 Get_CalibTable (SGC)

Get the calibration table of controller.

Parameter: String (binary, with 0x00) SA_CalibTable SA_CalibTable
Direction: Up
Valid values:
2048 bytes
convertable to 1024 WORDS
Description: Calibration table.

O.41 Upload_CalibTable

Send a calibration table for selected sensor to the controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_BlockSize SP_BlockSize
Direction: Down
Unit: Values
Valid values:
Minimum: 1
Maximum: 999
Description: Size of blocks in which the calibration table is separated before sending.

Parameter: String (binary, with 0x00) SP_CalibTable SP_CalibTable
Direction: Down
Valid values:
1024 values of datatype float (4 byte).
Description: Calibration table for selected sensor.

Parameter: double SP_FullRange SP_FullRange
Direction: Down
Unit: μm
Valid values:
Minimum: 1
Maximum: 1000000
Description: Measure range of selected sensor.

O.42 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_RefractIndex

IP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Tells the driver the refractive index used by the controller for thickness measurement.

Parameter: int IP_MeasureMode

IP_MeasureMode

Direction: Down

Valid values:

0 = Distance

1 = Thickness

Description: Tells the driver the measure mode of the controller.

Parameter: int IP_Averaging

IP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 999

Description: Tells the driver the averaging used by the controller. It is used for timeout check.

Parameter: int IP_X1..8

IP_X1..8

Direction: Down

Valid values:

0 = off

1 = RS232

9 = TCP/IP

Description: Tells the driver the selection of data used by the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

O.43 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_RefractIndex

IA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index of sensor assumed by the driver for thickness measurement.

Parameter: int IA_MeasureMode

IA_MeasureMode

Direction: Up

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor assumed by the driver.

Parameter: int IA_Averaging

IA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Averaging of the sensor assumed by the driver. It is used for timeout check.

Parameter: int IA_X1..8

IA_X1..8

Direction: Up

Valid values:

0= off

1= RS232

9= TCP/IP

Description: Selection of data of the sensor assumed by the driver. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..8.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (not used / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

P Commands for IFD2431

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

DriverX (native).

IF2008 (native).

P.1 Set_ActiveSensor (SEN)

Set the active sensor (up to 20 sensors can be stored in controller) for measurement. After this command, the internal range is resetted because the old one is no longer valid and the new one is not known until command Get_Range or Get_Status is called.

Parameter: int SP_Sensor

SP_Sensor

Direction: Down

Valid values:

Minimum: 0

Maximum: 19

Description: Number of active sensor.

P.2 Get_ActiveSensor (SEN?)

Get the active sensor.

Parameter: int SA_Sensor

SA_Sensor

Direction: Up

Valid values:

Minimum: 0

Maximum: 19

Description: Number of active sensor.

P.3 Get_Range (SCA)

Get the range of active sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: μm

Description: Range of active sensor.

P.4 Get_AllRanges (LUL)

Get the ranges of all calibrated sensors.

Parameter: String SA_Ranges

SA_Ranges

Direction: Up

Unit: μm (after conversion from string to int or double)

Description: Ranges of all 20 sensors in a string, separated by commas.

P.5 Acquire_DarkSig (DRK)

Acquire dark signal.

Parameter: int SA_MinSRIndex

SA_MinSRIndex

Direction: Up

Valid values:

1 = 500 Hz

2 = 1000 Hz

3 = 2000 Hz

4 = 5000 Hz

5 = 10000 Hz

6 = 15625 Hz

7 = 20000 Hz

8 = 25000 Hz

9 = 31250 Hz

Description: Minimal samplerate index.

P.6 FastDark (FDK)

Acquire dark signal for active sensor and samplerate.

Parameter: int SP_AveragingForDark

SP_AveragingForDark

Direction: Down

Valid values:

Minimum: 1

Maximum: 99

Description: Averaging factor for dark.

Parameter: int SP_Weighting

SP_Weighting

Direction: Down

Valid values:

Minimum: 1

Maximum: 100

Description: Weighting factor.

P.7 Set_AutoDark (ADK)

Enables/Disables the "Automatic Fast Dark" mode.

Parameter: int SP_AutoDark

SP_AutoDark

Direction: Down

Valid values:

0 = off

1 = on

Description: Automatic Fast Dark.

P.8 Get_AutoDark (ADK?)

Get the state of the "Automatic Fast Dark" mode.

Parameter: int SA_AutoDark

SA_AutoDark

Direction: Up

Valid values:

0= off

1= on

Description: Automatic Fast Dark.

P.9 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

0= Free samplerate

1= 500 Hz

2= 1000 Hz

3= 2000 Hz

4= 5000 Hz

5= 10000 Hz

6= 15625 Hz

7= 20000 Hz

8= 25000 Hz

9= 31250 Hz

Description: Samplerate index.

P.10 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= Free samplerate

1= 500 Hz

2= 1000 Hz

3= 2000 Hz

4= 5000 Hz

5= 10000 Hz

6= 15625 Hz

7= 20000 Hz

8= 25000 Hz

9= 31250 Hz

Description: Samplerate index.

P.11 Set_FreeSR (FRQ)

Set free samplerate value for data acquisition.

Parameter: int SP_FreeSR SP_FreeSR
Direction: Down
Valid values:
 Minimum: 100
 Maximum: 31250
Unit: Hz
Description: Free Frequency.

Parameter: int SA_FreeSR SA_FreeSR
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 31250
Unit: Hz
Description: Free Samplerate.

P.12 Get_FreeSR (FRQ?)

Get free samplerate value.

Parameter: int SA_FreeSR SA_FreeSR
Direction: Up
Valid values:
 Minimum: 100
 Maximum: 31250
Unit: Hz
Description: Free Samplerate.

P.13 Set_Exposure (TEX)

Set free exposure time for data acquisition.

Parameter: int SP_Exposure SP_Exposure
Direction: Down
Valid values:
 Minimum: 32
 Maximum: 2000
Unit: μs
Description: Free Exposure time.

Parameter: int SA_Exposure SA_Exposure
Direction: Up
Valid values:
 Minimum: 32
 Maximum: 2000
Unit: μs
Description: Free Exposure time.

P.14 Get_Exposure (TEX?)

Get free exposure time.

Parameter: int SA_Exposure

SA_Exposure

Direction: Up

Valid values:

Minimum: 32

Maximum: 2000

Unit: μs

Description: Free Exposure time.

P.15 Get_MinSR (FRM)

Get the minimum authorized samplerate.

Parameter: int SA_MinSR

SA_MinSR

Direction: Up

Valid values:

Minimum: 100

Maximum: 31250

Unit: Hz

Description: Minimum samplerate (determined by dark signal).

P.16 Set_MeasureMode (MOD)

Set the measure mode.

Parameter: int SP_MeasureMode

SP_MeasureMode

Direction: Down

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor.

P.17 Get_MeasureMode (MOD?)

Get the current measure mode.

Parameter: int SA_MeasureMode

SA_MeasureMode

Direction: Up

Valid values:

0= Distance

1= Thickness

Description: Measure mode of the sensor.

P.18 Set_RefractIndex (SRI)

Set the refraction index for thickness measurement.

Parameter: double SP_RefractIndex

SP_RefractIndex

Direction: Down

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

P.19 Get_RefractIndex (SRI?)

Get the current refraction index.

Parameter: double SA_RefractIndex

SA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index used by the sensor for thickness measurement.

P.20 Set_Threshold (MNP)

Set the detection threshold for distance mode.

Parameter: double SP_Threshold

SP_Threshold

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

P.21 Get_Threshold (MNP?)

Get the current detection threshold for distance mode.

Parameter: double SA_Threshold

SA_Threshold

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

P.22 Set_HoldLastValid (HLV)

Enable/Disable the "Hold Last Value" mode.

Parameter: int SP_LastValid

SP_LastValid

Direction: Down

Valid values:

Minimum: 0

Maximum: 999

Description: Max number of points to hold.

P.23 Get_HoldLastValid (HLV?)

Get the current "Hold Last Value" mode.

Parameter: int SA_LastValid

SA_LastValid

Direction: Up

Valid values:

Minimum: 0

Maximum: 999

Description: Max number of points to hold.

P.24 Set_Threshold1 (SPP)

Set the detection threshold for the strongest peak in thickness mode.

Parameter: double SP_Threshold1

SP_Threshold1

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

P.25 Get_Threshold1 (SPP?)

Get the current detection threshold for the strongest peak in thickness mode.

Parameter: double SA_Threshold1

SA_Threshold1

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Threshold of the sensor.

P.26 Set_Threshold2 (SDP)

Set the detection threshold for the second peak in thickness mode.

Parameter: double SP_Threshold2 SP_Threshold2
Direction: Down
Valid values:
 Minimum: 0.0
 Maximum: 1.0
Description: Threshold of the sensor.

P.27 Get_Threshold2 (SDP?)

Get the current detection threshold for the second peak in thickness mode.

Parameter: double SA_Threshold2 SA_Threshold2
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 1.0
Description: Threshold of the sensor.

P.28 Set_FirstPeakMode (MSP)

Enable/Disable the "First peak" mode.

Parameter: int SP_FirstPeak SP_FirstPeak
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 1
Description: First peak state.

P.29 Get_FirstPeakMode (MSP?)

Get the current "First peak" mode.

Parameter: int SA_FirstPeak SA_FirstPeak
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 1
Description: First peak state.

P.30 Set_DataScale (CEE)

Set the scaling factor for X1, X2 and X3 in thickness mode. This is an internal command. It should not be used by the customer.

Parameter: double SP_DataScale SP_DataScale
Direction: Down
Valid values:
 Minimum: 1.0
 Maximum: 5.0
Description: Scaling factor of the sensor.

P.31 Get_DataScale (CEE?)

Get the scaling factor for X1, X2 and X3 in thickness mode.

Parameter: double SA_DataScale SA_DataScale
Direction: Up
Valid values:
 Minimum: 1.0
 Maximum: 5.0
Description: Scaling factor of the sensor.

P.32 Set_BarycenterSca (CEB)

Set the scaling factor for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SP_BarycenterScale SP_BarycenterScale
Direction: Down
Valid values:
 Minimum: 32.0
 Maximum: 32.0
Description: Scaling factor for Barycenter.

P.33 Get_BarycenterSca (CEB?)

Get the scaling factor for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SA_BarycenterScale SA_BarycenterScale
Direction: Up
Valid values:
 Minimum: 32.0
 Maximum: 32.0
Description: Scaling factor for Barycenter.

P.34 Set_BarycenterOff (CRB)

Set the offset values for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SP_BarycenterOffset

SP_BarycenterOffset

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1023.9

Description: Offset for Barycenter.

P.35 Get_BarycenterOff (CRB?)

Get the offset value for barycenter values. This is an internal command. It should not be used by the customer.

Parameter: double SA_BarycenterOffset

SA_BarycenterOffset

Direction: Up

Valid values:

Minimum: 0.0

Maximum: 1023.9

Description: Offset for Barycenter.

P.36 Set_RefIdxFile (INF)

Set the refractive index file.

Parameter: int SP_RefractIndexFileIdx

SP_RefractIndexFileIdx

Direction: Down

Valid values:

Minimum: 0

Maximum: 8

Description: Refractive index file index.

Parameter: int SA_RefractIndexFileIdx

SA_RefractIndexFileIdx

Direction: Up

Valid values:

Minimum: 0

Maximum: 8

Description: Refractive index file index.

Parameter: String SA_RefractIndexFileName

SA_RefractIndexFileName

Direction: Up

Description: Refractive index file name.

Parameter: double SA_MinRefractIndex

SA_MinRefractIndex

Direction: Up

Valid values:

Minimum: 0.0001

Maximum: 9.9999

Description: Minimum refractive index.

Parameter: double SA_MaxRefractIndex SA_MaxRefractIndex
Direction: Up
Valid values:
 Minimum: 0.0001
 Maximum: 9.9999
Description: Maximum refractive index.

Parameter: double SA_AvgRefractIndex SA_AvgRefractIndex
Direction: Up
Valid values:
 Minimum: 0.0001
 Maximum: 9.9999
Description: Averaging refractive index.

P.37 Get_RefIdxFile (INF?)

Request the selected refractive index file.

Parameter: int SA_RefractIndexFileIdx SA_RefractIndexFileIdx
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 8
Description: Refractive index file index.

Parameter: String SA_RefractIndexFileName SA_RefractIndexFileName
Direction: Up
Description: Refractive index file name.

Parameter: double SA_MinRefractIndex SA_MinRefractIndex
Direction: Up
Valid values:
 Minimum: 0.0001
 Maximum: 9.9999
Description: Minimum refractive index.

Parameter: double SA_MaxRefractIndex SA_MaxRefractIndex
Direction: Up
Valid values:
 Minimum: 0.0001
 Maximum: 9.9999
Description: Maximum refractive index.

Parameter: double SA_AvgRefractIndex SA_AvgRefractIndex
Direction: Up
Valid values:
 Minimum: 0.0001
 Maximum: 9.9999
Description: Averaging refractive index.

P.38 Upload_RefrIdxFile

Send a refractive index file to the controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_RefractIndexFileIdx

SP_RefractIndexFileIdx

Direction: Down

Valid values:

Minimum: 1

Maximum: 8

Description: Index on which position the file should be stored.

Parameter: String SP_RefractIndexFile

SP_RefractIndexFile

Direction: Down

Description: Refractive index file with 1025 lines. First line is a name, the other lines are refractive index values with precision four.

P.39 RecenterEncoder (RCD)

Recenter encoder position.

Parameter: int SP_Encoder1

SP_Encoder1

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Recenter Encoder 1.

Parameter: int SP_Encoder2

SP_Encoder2

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Recenter Encoder 2.

Parameter: int SP_Encoder3

SP_Encoder3

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Recenter Encoder 3.

P.40 Set_MissingSignal (RSP)

Set behaviour for missing second peak in thickness mode.

Parameter: int SP_Option

SP_Option

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Mode for behaviour.

P.41 Get_MissingSignal (RSP?)

Returns behaviour for missing second peak in thickness mode.

Parameter: int SA_Option

SA_Option

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: Mode for behaviour.

P.42 Set_Reverse (RVS)

Reverse the distance signal.

Parameter: int SP_Reverse

SP_Reverse

Direction: Down

Valid values:

0 = Normal direction

1 = Reverse direction

Description: In reverse mode, measure range - distance is transmitted.

P.43 Get_Reverse (RVS?)

Returns setting for reverse mode.

Parameter: int SA_Reverse

SA_Reverse

Direction: Up

Valid values:

0 = Normal direction

1 = Reverse direction

Description: Reverse mode

P.44 Set_Binning (FBH)

Set the binning value.

This is an internal command. It should not be used by the customer.

Parameter: int SP_Binning

SP_Binning

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Binning value

P.45 Get_Binning (FBH?)

Get the binning value.

This is an internal command. It should not be used by the customer.

Parameter: int SA_Binning

SA_Binning

Direction: Up

Valid values:

Minimum: 1

Maximum: 2

Description: Binning value

P.46 Set_Averaging (AVR)

Set data averaging at controller. Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 9999

Description: Data averaging factor.

P.47 Get_Averaging (AVR?)

Returns current data averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

P.48 Set_SpectralAv (AVS)

Set spectral averaging at controller. Note! Acquisition speed will be reduced.

Parameter: int SP_Averaging

SP_Averaging

Direction: Down

Valid values:

Minimum: 1

Maximum: 9999

Description: Data averaging factor.

P.49 Get_SpectralAv (AVS?)

Returns current spectral averaging at controller.

Parameter: int SA_Averaging

SA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Data averaging factor.

P.50 Continue (CTN)

Continues data acquisition (after trigger commands SingleShot_Trg, TriggerMode_Edge, TriggerMode_State or an error).

P.51 Start_Trigger (TRG)

Starts the trigger mode at controller.

P.52 End_Trigger

Stops the trigger mode at controller (after Start_Trigger).

P.53 SingleShot_Trg (TRE)

Starts the single shot trigger mode at controller.

Parameter: int SP_NumberOfPoints

SP_NumberOfPoints

Direction: Down

Valid values:

Minimum: 0

Maximum: 9999

Description: Number of values to read.

P.54 Set_TrgMode_Edge (TRS)

Enable/Disable the "Start/Stop on Edge Trigger" mode.

Parameter: int SP_TriggerMode_Edge

SP_TriggerMode_Edge

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Start/Stop on Edge Trigger state.

P.55 Set_TrgMode_State (TRN)

Enable/Disable the "Start/Stop on State Trigger" mode.

Parameter: int SP_TriggerMode_State

SP_TriggerMode_State

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Start/Stop on State Trigger state.

P.56 Set_ActiveEdge (TRF)

Set the active edge or state for Start_Trigger, TriggerMode_Edge or TriggerMode_State.

Parameter: int SP_ActiveEdge

SP_ActiveEdge

Direction: Down

Valid values:

0= rising edge or high state

1= falling edge or low state

Description: Active state or edge.

P.57 Get_ActiveEdge (TRF?)

Get the active edge or state for Start_Trigger, TriggerMode_Edge or TriggerMode_State.

Parameter: int SA_ActiveEdge

SA_ActiveEdge

Direction: Up

Valid values:

0= rising edge or high state

1= falling edge or low state

Description: Active state or edge.

P.58 Software_Trigger (STR)

Simulates an hardware trigger (SYNC IN) for trigger commands SingleShot_Trg, TriggerMode_Edge or TriggerMode_State.

P.59 Set_Watchdog (WDE)

Enable/Disable the Watchdog.

Parameter: int SP_Watchdog

SP_Watchdog

Direction: Down

Valid values:

Minimum: 0

Maximum: 1

Description: Watchdog state.

P.60 Get_Watchdog (WDE?)

Get the Watchdog state.

Parameter: int SA_Watchdog

SA_Watchdog

Direction: Up

Valid values:

Minimum: 0

Maximum: 1

Description: Watchdog state.

P.61 Set_WatchdogPrd (WDP)

Set the Watchdog period.

Parameter: int SP_WatchdogPeriod

SP_WatchdogPeriod

Direction: Down

Unit: s

Valid values:

Minimum: 10

Maximum: 255

Description: Watchdog period.

P.62 Get_WatchdogPrd (WDP?)

Get the Watchdog period.

Parameter: int SA_WatchdogPeriod

SA_WatchdogPeriod

Direction: Up

Unit: s

Valid values:

Minimum: 10

Maximum: 255

Description: Watchdog period.

P.63 Get_Status (STS)

Retrieve detailed information about the controller and sensor.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= Free samplerate

1= 500 Hz

2= 1000 Hz

3= 2000 Hz

4= 5000 Hz

5= 10000 Hz

6= 15625 Hz

7= 20000 Hz

8= 25000 Hz

9= 31250 Hz

Description: Samplerate index.

Parameter: int SA_MeasureMode Direction: Up Valid values: 0= Distance 1= Thickness Description: Measure mode of the sensor.	SA_MeasureMode
Parameter: int SA_Sensor Direction: Up Valid values: Minimum: 0 Maximum: 19 Description: Number of active sensor.	SA_Sensor
Parameter: int SA_ASCII Direction: Up Description: Data Transfer mode (values) of sensor.	SA_ASCII
Parameter: int SA_Averaging Direction: Up Valid values: Minimum: 1 Maximum: 999 Description: Data averaging factor.	SA_Averaging
Parameter: int SA_X1..16 Direction: Up Valid values: 0= off 1= RS232/RS422 9= USB Description: Selection of data of the sensor. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used). X5 means (not used / Intensity1). X6 means (Counter / Intensity2). X7 means (Barycenter / Barycenter1). X8 means (not used / Barycenter2). X9 means (State Flags / State Flags). X10 means (not used / Counter). X11 means (Encoder 1 LSB / Encoder 1 LSB). X12 means (Encoder 1 MSB / Encoder 1 MSB). X13 means (Encoder 2 LSB / Encoder 2 LSB). X14 means (Encoder 2 MSB / Encoder 2 MSB). X15 means (Encoder 3 LSB / Encoder 3 LSB). X16 means (Encoder 3 MSB / Encoder 3 MSB).	SA_X1..16

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: μm

Description: Range of active sensor.

P.64 Get_Version (VER)

Get the software version of the controller.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Software version of sensor.

P.65 Set_OutputData (SOD)

Select the data to be output from controller.

Parameter: int SP_X1..16

SP_X1..16

Direction: Down

Valid values:

0= off

1= RS232/RS422

9= USB

Description: Selection of data of the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

P.66 Get_OutputData (SOD?)

Get the data output from controller.

Parameter: int SA_X1..16

SA_X1..16

Direction: Up

Valid values:

0= off

1= RS232/RS422

9= USB

Description: Selection of data of the sensor. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

P.67 Set_Ascii (ASC)

Set digital data transfer (only values, no sensor answer) to ASCII mode.

P.68 Set_Binary (BIN)

Set digital data transfer (only values, no sensor answer) to binary mode.

P.69 Set_AnalogOut (ANA)

Setup the analog outputs.

Parameter: int SP_OutNr

SP_OutNr

Direction: Down

Valid values:

0= BNC1

1= BNC2

Description: Number of analog output.

Parameter: int SP_SOD Direction: Down Valid values: Minimum: 0 Maximum: 7 Description: Specifies the number (X1..X16) to output at analog out.	SP_SOD
Parameter: int SP_0V Direction: Down Valid values: Minimum: 0 Maximum: 99999 Description: Specifies the value which should output 0 V.	SP_0V
Parameter: int SP_10V Direction: Down Valid values: Minimum: 0 Maximum: 99999 Description: Specifies the value which should output 10 V.	SP_10V

P.70 Get_AnalogOut (ANA?)

Get settings of analog outputs.

Parameter: int SA_0_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC1: 0= X1, 1= X2, ..., 15= X16.	SA_0_SOD
Parameter: int SA_0_0V Direction: Up Valid values: Minimum: 0 Maximum: 99999 Description: Analog out BNC1: Specifies the value which should output 0 V.	SA_0_0V
Parameter: int SA_0_10V Direction: Up Valid values: Minimum: 0 Maximum: 99999 Description: Analog out BNC1: Specifies the value which should output 10 V.	SA_0_10V
Parameter: int SA_1_SOD Direction: Up Valid values: Minimum: 0 Maximum: 7 Description: Analog out BNC2: 0= X1, 1= X2, ..., 15= X16.	SA_1_SOD

Parameter: int SA_1_0V SA_1_0V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 99999
Description: Analog out BNC2: Specifies the value which should output 0 V.

Parameter: int SA_1_10V SA_1_10V
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 99999
Description: Analog out BNC2: Specifies the value which should output 10 V.

P.71 Set_AnalogZero (SOF)

Set analog output to zero.

Parameter: int SP_Zero SP_Zero
Direction: Down
Valid values:
 0 = normal
 1 = set
Description: Set/reset the analog output 0V value.

P.72 Set_Baudrate (BAU)

Set the baudrate of the serial interface of controller.

Attention! To avoid synchronisation problems, sending data by sensor should be disabled after changing baudrate for a short time (sensor command Start_Trigger).

Parameter: int SP_SensorBaudrate SP_SensorBaudrate
Direction: Down
Valid values:
 9600
 19200
 38400
 57600
 115200
 230400
 460800
Description: Baudrate of controller.

P.73 Get_Baudrate (BAU?)

Get the baudrate of the serial interface of controller.

Parameter: int SA_SensorBaudrate

SA_SensorBaudrate

Direction: Up

Valid values:

9600
19200
38400
57600
115200
230400
460800

Description: Baudrate of controller.

P.74 Save_Setup (SSU)

Save the current setup of controller to flash.

P.75 Reset (RST)

Reset the controller and set the default parameter values.

P.76 Get_CCD (CCD)

Returns current CCD data.

Parameter: String (binary, with 0x00) SA_CCD

SA_CCD

Direction: Up

Valid values:

2048 bytes
convertible to 1024 WORDS.

Description: Raw CCD line.

Parameter: String (binary, with 0x00) SA_PreTreated

SA_PreTreated

Direction: Up

Valid values:

2048 bytes
convertible to 1024 WORDS.

Description: Raw CCD line.

P.77 Get_DarkSig (SGD)

Get the dark signal table of controller.

Parameter: String (binary, with 0x00) SA_DarkSig

SA_DarkSig

Direction: Up

Valid values:

2048 bytes
convertible to 1024 WORDS.

Description: Dark signal table.

P.78 Get_WhiteRef (SGW)

Get the white reference table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_WhiteRef

SA_WhiteRef

Direction: Up

Valid values:

2048 bytes

convertible to 1024 WORDS.

Description: White reference table.

P.79 Get_NormSig (SGN)

Get the norm signal table of controller. This is an internal command. It should not be used by the customer.

Parameter: String (binary, with 0x00) SA_NormSig

SA_NormSig

Direction: Up

Valid values:

2048 bytes

convertible to 1024 WORDS.

Description: Norm signal table.

P.80 Get_CalibTable (SGC)

Get the calibration table of controller.

Parameter: String (binary, with 0x00) SA_CalibTable

SA_CalibTable

Direction: Up

Valid values:

2048 bytes

convertible to 1024 WORDS.

Description: Calibration table.

P.81 Start_Spectrum

Start the spectrum mode in controller. Only work via USB connection.

On firmware versions below 1.2.36 the spectrum is shifted 8 bytes to left.

This mode does work up to 4.7 kHz only. For higher samplerates data will be lost and the spectrums are corrupted. To avoid this, increase spectral averaging.

Parameter: int IP_UsbReadBufSize

IP_UsbReadBufSize

Direction: Down

Unit: Bytes

Valid values:

Minimum: 512

Maximum: 65536

Default: 8192

Description: Buffer size for read operations on USB, while spectrum mode is active. The value is always ceiled to the next power of two (512, 1024, 2048, ..., 32768, 65536).

Parameter: int IP_TimerResolution

IP_TimerResolution

Direction: Down

Unit: ms

Valid values:

-1= Do not set timer resolution.

0= Use greatest possible accuracy.

1..255= Resolution in milliseconds.

Unit: ms

Default: 0

Description: Timer resolution (for Windows scheduler, set by timeBeginPeriod).

It is automatically reset at End_Spectrum.

P.82 Get_Spectrum

Read one Spectrum (PreTreated Signal). It is returned with a frequency of Samplerate divided by Spectral Averaging.

Parameter: int SP_WaitSpectrumTimeout

SP_WaitSpectrumTimeout

Direction: Down

Unit: ms

Valid values:

Minimum: 0

Maximum: 1000000000

Description: Timeout to wait for a spectrum.

Parameter: int SP_ReadMode

SP_ReadMode

Direction: Down

Valid values:

0= Each spectrum

1= Only newest spectrum

2= Automatic

Description: This mode specifies if each spectrum should be read or only the latest one. If set to automatic each spectrum is read until the buffer does not overflow. If the buffer becomes full one or more spectra are discarded.

Parameter: String (binary, with 0x00) SA_Spectrum

SA_Spectrum

Direction: Up

Valid values:

2048 bytes

convertible to 1024 WORDS.

Description: PreTreated Signal.

Parameter: double SA_Timestamp

SA_Timestamp

Direction: Up

Valid values:

Minimum: 0

Unit: ms

Description: Timestamp of the signal. It starts from 1970 Jan 01 at 01:00. It is generated when the spectrum is read from USB

Parameter: int SA_SkippedSpectra SA_SkippedSpectra
Direction: Up
Valid values:
 Minimum: 0
 Maximum: INT_MAX (2147483647)
Description: Number of skipped spectra, if SP_NewestSpectrum is set to 1.

P.83 End_Spectrum

End the spectrum mode in controller.

Parameter: int IP_UsbReadBufSize IP_UsbReadBufSize
Direction: Down
Unit: Bytes
Valid values:
 Minimum: 512
 Maximum: 65536
Default: 512
Description: Buffer size for read operations on USB, after spectrum mode was finished. The value is always ceiled to the next power of two (512, 1024, 2048, ..., 32768, 65536).

P.84 Upload_CalibTable

Send a calibration table for selected sensor to the controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_BlockSize SP_BlockSize
Direction: Down
Unit: Values
Valid values:
 Minimum: 1
 Maximum: 999
Default: 64
Description: Size of blocks in which the calibration table is separated before sending.

Parameter: String (binary, with 0x00) SP_CalibTable SP_CalibTable
Direction: Down
Valid values:
 1024 values of datatype float (4 byte).
Description: Calibration table for selected sensor.

Parameter: double SP_FullRange SP_FullRange
Direction: Down
Unit: μm
Valid values:
 Minimum: 1
 Maximum: 1000000
Description: Measure range of selected sensor.

P.85 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_RefractIndex Direction: Down Valid values: Minimum: 0.001 Maximum: 9.999 Description: Tells the driver the refractive index used by the controller for thickness measurement.	IP_RefractIndex
Parameter: int IP_MeasureMode Direction: Down Valid values: 0 = Distance 1 = Thickness Description: Tells the driver the measure mode of the controller.	IP_MeasureMode
Parameter: int IP_Averaging Direction: Down Valid values: Minimum: 1 Maximum: 999 Description: Tells the driver the averaging used by the controller. It is used for timeout check.	IP_Averaging
Parameter: int IP_SpectralAv Direction: Down Valid values: Minimum: 1 Maximum: 999 Description: Tells the driver the spectral averaging used by the controller. It is used for timeout check.	IP_SpectralAv
Parameter: int IP_X1..16 Direction: Down Valid values: 0 = off 1 = RS232 9 = USB Description: Tells the driver the selection of data used by the sensor. This is necessary for data conversion and scaling. Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16. X1 means (Distance / Thickness). X2 means (not used / Distance1). X3 means (not used / Distance2). X4 means (Intensity / not used).	IP_X1..16

X5 means (not used / Intensity1).
 X6 means (Counter / Intensity2).
 X7 means (Barycenter / Barycenter1).
 X8 means (not used / Barycenter2).
 X9 means (State Flags / State Flags).
 X10 means (not used / Counter).
 X11 means (Encoder 1 LSB / Encoder 1 LSB).
 X12 means (Encoder 1 MSB / Encoder 1 MSB).
 X13 means (Encoder 2 LSB / Encoder 2 LSB).
 X14 means (Encoder 2 MSB / Encoder 2 MSB).
 X15 means (Encoder 3 LSB / Encoder 3 LSB).
 X16 means (Encoder 3 MSB / Encoder 3 MSB).

Parameter: int IP_Binning

IP_Binning

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Tells the driver the binning value used by the controller.

This is an internal parameter. It should not be used by the customer.

P.86 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_RefractIndex

IA_RefractIndex

Direction: Up

Valid values:

Minimum: 0.001

Maximum: 9.999

Description: Refractive index of sensor assumed by the driver for thickness measurement.

Parameter: int IA_MeasureMode

IA_MeasureMode

Direction: Up

Valid values:

0 = Distance

1 = Thickness

Description: Measure mode of the sensor assumed by the driver.

Parameter: int IA_Averaging

IA_Averaging

Direction: Up

Valid values:

Minimum: 1

Maximum: 999

Description: Averaging of the sensor assumed by the driver. It is used for timeout check.

Parameter: int IA_SpectralAv

IA_SpectralAv

Direction: Up**Valid values:****Minimum:** 1**Maximum:** 999**Description:** Spectral averaging of the controller assumed by the driver. It is used for timeout check.**Parameter:** int IA_X1..16

IA_X1..16

Direction: Up**Valid values:**

0= off

1= RS232

9= USB

Description: Selection of data of the sensor assumed by the driver. This is necessary for data conversion and scaling.

Depending on measure mode (distance / thickness) the selected value have different meaning. The following list shows the meaning for X1..16.

X1 means (Distance / Thickness).

X2 means (not used / Distance1).

X3 means (not used / Distance2).

X4 means (Intensity / not used).

X5 means (not used / Intensity1).

X6 means (Counter / Intensity2).

X7 means (Barycenter / Barycenter1).

X8 means (not used / Barycenter2).

X9 means (State Flags / State Flags).

X10 means (not used / Counter).

X11 means (Encoder 1 LSB / Encoder 1 LSB).

X12 means (Encoder 1 MSB / Encoder 1 MSB).

X13 means (Encoder 2 LSB / Encoder 2 LSB).

X14 means (Encoder 2 MSB / Encoder 2 MSB).

X15 means (Encoder 3 LSB / Encoder 3 LSB).

X16 means (Encoder 3 MSB / Encoder 3 MSB).

Parameter: int IA_Binning

IA_Binning

Direction: Up**Valid values:****Minimum:** 1**Maximum:** 2**Description:** Binning value of the controller assumed by the driver.

This is an internal parameter. It should not be used by the customer.

Q Commands for ODC2500

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

Q.1 Reset_Boot

Resets the sensor. This command has no parameters.

Q.2 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_SerialNumber	SA_SerialNumber
Direction: Up	
Description: Serial number of the sensor.	
Parameter: String SA_Option	SA_Option
Direction: Up	
Description: Option of the sensor.	
Parameter: double SA_Range	SA_Range
Direction: Up	
Unit: mm	
Description: Range of the sensor.	
Parameter: int SA_Reserve_1	SA_Reserve_1
Direction: Up	
Description: Reserved.	
Parameter: String SA_SoftArtBoot	SA_SoftArtBoot
Direction: Up	
Description: Sensor software versions.	
Parameter: String SA_SoftArtArm	SA_SoftArtArm
Direction: Up	
Description: Sensor software versions.	
Parameter: String SA_SoftArtDSP	SA_SoftArtDSP
Direction: Up	
Description: Sensor software versions.	
Parameter: String SA_SoftVerBoot	SA_SoftVerBoot
Direction: Up	
Description: Sensor software versions.	

Parameter: String SA_SoftVerArm SA_SoftVerArm
Direction: Up
Description: Sensor software versions.

Parameter: String SA_SoftVerDSP SA_SoftVerDSP
Direction: Up
Description: Sensor software versions.

Q.3 Dat_Out_Off

Switch off data output from sensor.

Q.4 Dat_Out_On

Switch on data output from sensor.

Q.5 Choose_MeasProg

Select the measurement program of sensor.

Parameter: int SP_MeasProgNumber SP_MeasProgNumber
Direction: Down
Valid values:
 0= EdgeHL
 1= EdgeLH
 2= DIA
 3= GAP
 4= SEG_2_4
 5= 2-SEG
 6= USER1
 7= USER2
 8= USER3
 9= USER4
Description: Measure program of sensor.

Q.6 Switch_Edge

Select the edges to measure.

Parameter: int SP_FrontEdge_Seg1 SP_FrontEdge_Seg1
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 32
Description: Front edge of segment 1.

Parameter: int SP_FrontEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Front edge of segment 2.	SP_FrontEdge_Seg2
Parameter: int SP_RearEdge_Seg1 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Rear edge of segment 1.	SP_RearEdge_Seg1
Parameter: int SP_RearEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Rear edge of segment 2.	SP_RearEdge_Seg2

Q.7 Read_OptionData

Read the option data from sensor.

Parameter: int SA_MeasProgNumber Direction: Up Valid values: 0 to 5= Standard 6 to 9= User defined Description: Measure program of sensor.	SA_MeasProgNumber
Parameter: int SA_Language Direction: Up Valid values: 0= German 1= English Description: Language of sensor.	SA_Language
Parameter: int SA_DispMeasUnit Direction: Up Valid values: 0= mm 1= inch Description: Display measurement unit of sensor.	SA_DispMeasUnit
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0= error output 1= retain last value Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler

Parameter: int SA_Reserve_2 Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SA_Reserve_2
Parameter: int SA_Ext_LaserSwitch Direction: Up Valid values: 0 = not active 1 = active Description: Enable/Disable external laser switch.	SA_Ext_LaserSwitch
Parameter: int SA_LaserIntensity Direction: Up Valid values: Minimum: 0 Maximum: 100 Unit: % Description: No effect.	SA_LaserIntensity
Parameter: int SA_Contrast Direction: Up Valid values: Minimum: 0 Maximum: 100 Unit: % Description: Display contrast.	SA_Contrast
Parameter: int SA_Reserve_3 Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SA_Reserve_3
Parameter: int SA_ActiveSerialIf Direction: Up Valid values: 0 = RS422 1 = RS232 Description: Active serial interface of sensor.	SA_ActiveSerialIf
Parameter: int SA_RS232_Baudrate Direction: Up Valid values: 9600 19200 38400 115200 Unit: Baud Description: Baudrate of RS232 interface of sensor.	SA_RS232_Baudrate

Parameter: int SA_RS232_Parity Direction: Up Valid values: 0= none 1= even 2= odd Description: Parity of RS232 interface of sensor.	SA_RS232_Parity
Parameter: int SA_RS232_StopBits Direction: Up Valid values: 1 2 Description: Stop bits of RS232 interface of sensor.	SA_RS232_StopBits
Parameter: int SA_RS232_TimeoutSend Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS232_TimeoutSend
Parameter: int SA_RS232_TimeoutRecv Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS232_TimeoutRecv
Parameter: int SA_RS422_Baudrate Direction: Up Valid values: 9600 19200 38400 115200 691200 Unit: Baud Description: Baudrate of RS422 interface of sensor.	SA_RS422_Baudrate
Parameter: int SA_RS422_Parity Direction: Up Valid values: 0= none 1= even 2= odd Description: Parity of RS422 interface of sensor.	SA_RS422_Parity
Parameter: int SA_RS422_StopBits Direction: Up Valid values: 1 2 Description: Stop bits of RS422 interface of sensor.	SA_RS422_StopBits

Parameter: int SA_RS422_TimeoutSend SA_RS422_TimeoutSend
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: No effect.

Parameter: int SA_RS422_TimeoutRecv SA_RS422_TimeoutRecv
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: No effect.

Q.8 Read_MeasProgData

Read the measurement program data from sensor.

Parameter: int SA_UserMeasProgNumber SA_UserMeasProgNumber
Direction: Up
Valid values:
 Minimum: 6
 Maximum: 9= User1 to User4
Description: User measure program of sensor.

Parameter: String SA_MeasProgName SA_MeasProgName
Direction: Up
Description: Measure program name of sensor.

Parameter: double SA_AnalogOffset SA_AnalogOffset
Direction: Up
Valid values:
 Minimum: -10.0
 Maximum: 10.0
Unit: V
Description: Analog output offset of sensor.

Parameter: double SA_AnalogGain SA_AnalogGain
Direction: Up
Valid values:
 Minimum: -3.4
 Maximum: 3.4
Description: Analog output gain of sensor.

Parameter: double SA_DisplayOffset SA_DisplayOffset
Direction: Up
Valid values:
 Minimum: -99.99
 Maximum: 99.99
Unit: mm
Description: Display offset of sensor.

Parameter: double SA_DisplayGain Direction: Up Valid values: Minimum: -2.0 Maximum: 2.0 Description: Display gain of sensor.	SA_DisplayGain
Parameter: double SA_UpperLimit Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper limit of sensor.	SA_UpperLimit
Parameter: double SA_LowerLimit Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower limit of sensor.	SA_LowerLimit
Parameter: double SA_UpperWarning Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper warning of sensor.	SA_UpperWarning
Parameter: double SA_LowerWarning Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower warning of sensor.	SA_LowerWarning
Parameter: int SA_Target_Distance Direction: Up Valid values: 0= 20 mm 1= 50 mm 2= 100 mm 3= 150 mm Description: Target distance of sensor.	SA_Target_Distance
Parameter: int SA_Average_for_reading Direction: Up Valid values: 1 to 128 sliding 129 to 4096 recursive Description: Averaging mode and number of sensor.	SA_Average_for_reading

Parameter: int SA_Reserve_4

SA_Reserve_4

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 65535**Description:** Reserved.**Parameter:** int SA_MeasObject

SA_MeasObject

Direction: Up**Valid values:**

1 = EdgeHL

2 = EdgeLH

3 = DIA

4 = GAP

5 = SEG_2_4

6 = 2-SEG

Description: Measurement program.**Parameter:** int SA_FrontEdge_Seg1

SA_FrontEdge_Seg1

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 32**Description:** Front edge of segment 1.**Parameter:** int SA_FrontEdge_Seg2

SA_FrontEdge_Seg2

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 32**Description:** Front edge of segment 2.**Parameter:** int SA_RearEdge_Seg1

SA_RearEdge_Seg1

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 32**Description:** Rear edge of segment 1.**Parameter:** int SA_RearEdge_Seg2

SA_RearEdge_Seg2

Direction: Up**Valid values:****Minimum:** 0**Maximum:** 32**Description:** Rear edge of segment 2.**Parameter:** double SA_MasterValue

SA_MasterValue

Direction: Up**Valid values:****Minimum:** -34.0**Maximum:** 34.0**Unit:** mm**Description:** Master value of sensor.

Q.9 Write_OptionData

Write the option data to sensor.

Parameter: int SP_MeasProgNumber Direction: Down Valid values: 0..5= Standard 6..9= User defined Description: Measure program of sensor.	SP_MeasProgNumber
Parameter: int SP_Language Direction: Down Valid values: 0= German 1= English Description: Language of sensor.	SP_Language
Parameter: int SP_DispMeasUnit Direction: Down Valid values: 0= mm 1= inch Description: Display measurement unit of sensor.	SP_DispMeasUnit
Parameter: int SP_ErrorHandler Direction: Down Valid values: 0= error output 1= retain last value Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SP_ErrorHandler
Parameter: int SP_Reserve_2 Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SP_Reserve_2
Parameter: int SP_Ext_LaserSwitch Direction: Down Valid values: 0= not active 1= active Description: Enable/Disable external laser switch.	SP_Ext_LaserSwitch
Parameter: int SP_LaserIntensity Direction: Down Valid values: Minimum: 0 Maximum: 100 Unit: % Description: No effect.	SP_LaserIntensity

Parameter: int SP_Contrast Direction: Down Valid values: Minimum: 0 Maximum: 100 Unit: % Description: Display contrast.	SP_Contrast
Parameter: int SP_Reserve_3 Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SP_Reserve_3
Parameter: int SP_ActiveSerialIf Direction: Down Valid values: 0 = RS422 1 = RS232 Description: Active serial interface of sensor.	SP_ActiveSerialIf
Parameter: int SP_RS232_Baudrate Direction: Down Valid values: 9600 19200 38400 115200 Unit: Baud Description: Baudrate of RS232 interface of sensor.	SP_RS232_Baudrate
Parameter: int SP_RS232_Parity Direction: Down Valid values: 0 = none 1 = even 2 = odd Description: Parity of RS232 interface of sensor.	SP_RS232_Parity
Parameter: int SP_RS232_StopBits Direction: Down Valid values: 1 2 Description: Stop bits of RS232 interface of sensor.	SP_RS232_StopBits
Parameter: int SP_RS232_TimeoutSend Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS232_TimeoutSend

Parameter: int SP_RS232_TimeoutRecv Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS232_TimeoutRecv
Parameter: int SP_RS422_Baudrate Direction: Down Valid values: 9600 19200 38400 115200 691200 Unit: Baud Description: Baudrate of RS422 interface of sensor.	SP_RS422_Baudrate
Parameter: int SP_RS422_Parity Direction: Down Valid values: 0= none 1= even 2= odd Description: Parity of RS422 interface of sensor.	SP_RS422_Parity
Parameter: int SP_RS422_StopBits Direction: Down Valid values: 1 2 Description: Stop bits of RS422 interface of sensor.	SP_RS422_StopBits
Parameter: int SP_RS422_TimeoutSend Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS422_TimeoutSend
Parameter: int SP_RS422_TimeoutRecv Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS422_TimeoutRecv

Q.10 Write_MeasProgData

Write the measurement program data to sensor.

Parameter: int SP_UserMeasProgNumber Direction: Down Valid values: Minimum: 6 Maximum: 9= User1 to User4 Description: User measure program of sensor.	SP_UserMeasProgNumber
--	-----------------------

Parameter: String SP_MeasProgName Direction: Down Description: Measure program name of sensor.	SP_MeasProgName
Parameter: double SP_AnalogOffset Direction: Down Valid values: Minimum: -10.0 Maximum: 10.0 Unit: V Description: Analog output offset of sensor.	SP_AnalogOffset
Parameter: double SP_AnalogGain Direction: Down Valid values: Minimum: -3.4 Maximum: 3.4 Description: Analog output gain of sensor.	SP_AnalogGain
Parameter: double SP_DisplayOffset Direction: Down Valid values: Minimum: -99.99 Maximum: 99.99 Unit: mm Description: Display offset of sensor.	SP_DisplayOffset
Parameter: double SP_DisplayGain Direction: Down Valid values: Minimum: -2.0 Maximum: 2.0 Description: Display gain of sensor.	SP_DisplayGain
Parameter: double SP_UpperLimit Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper limit of sensor.	SP_UpperLimit
Parameter: double SP_LowerLimit Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower limit of sensor.	SP_LowerLimit
Parameter: double SP_UpperWarning Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper warning of sensor.	SP_UpperWarning

Parameter: double SP_LowerWarning Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower warning of sensor.	SP_LowerWarning
Parameter: int SP_Target_Distance Direction: Down Valid values: 0= 20 mm 1= 50 mm 2= 100 mm 3= 150 mm Description: Target distance of sensor.	SP_Target_Distance
Parameter: int SP_Average_for_reading Direction: Down Valid values: 1 to 128 sliding 129 to 4096 recursive Description: Averaging mode and number of sensor.	SP_Average_for_reading
Parameter: int SP_Reserve_4 Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SP_Reserve_4
Parameter: int SP_MeasObject Direction: Down Valid values: 1= EdgeHL 2= EdgeLH 3= DIA 4= GAP 5= SEG_2_4 6= 2-SEG Description: Measurement program.	SP_MeasObject
Parameter: int SP_FrontEdge_Seg1 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Front edge of segment 1.	SP_FrontEdge_Seg1
Parameter: int SP_FrontEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Front edge of segment 2.	SP_FrontEdge_Seg2

Parameter: int SP_RearEdge_Seg1 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Rear edge of segment 1.	SP_RearEdge_Seg1
Parameter: int SP_RearEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 32 Description: Rear edge of segment 2.	SP_RearEdge_Seg2
Parameter: double SP_MasterValue Direction: Down Valid values: Minimum: -34.0 Maximum: 34.0 Unit: mm Description: Master value of sensor.	SP_MasterValue

Q.11 Save_OptionData

Save the option data to flash.

Q.12 Save_MeasProgData

Save the measure program data to flash.

Q.13 Read_MinMax

Read the minimum and maximum values from sensor.

Parameter: int SA_MinRaw Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Minimum raw value.	SA_MinRaw
Parameter: int SA_MaxRaw Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Maximum raw value.	SA_MaxRaw

Parameter: int SA_MinScaled SA_MinScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 34.0
Unit: mm
Description: Minimum scaled value.

Parameter: int SA_MaxScaled SA_MaxScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 34.0
Unit: mm
Description: Maximum scaled value.

Q.14 Read_MinMaxReset

Read the minimum and maximum values from sensor and reset the values.

Parameter: int SA_MinRaw SA_MinRaw
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Minimum raw value.

Parameter: int SA_MaxRaw SA_MaxRaw
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Maximum raw value.

Parameter: int SA_MinScaled SA_MinScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 34.0
Unit: mm
Description: Minimum scaled value.

Parameter: int SA_MaxScaled SA_MaxScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 34.0
Unit: mm
Description: Maximum scaled value.

Q.15 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_NumberOfSegments

IP_NumberOfSegments

Direction: Down

Valid values:

1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4

2 for 2-SEG

Description: Tells the driver the number of segments measured by the sensor.

Q.16 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_NumberOfSegments

IA_NumberOfSegments

Direction: Up

Valid values:

1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4

2 for 2-SEG

Description: Number of segments used by driver.

R Commands for ODC2600

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

RS232 (native).

IF2004 (native).

TCP/IP (additional, e.g. RS232/RS422 to TCP/IP comm server and RS232 high level interface).

IF2008 (native).

R.1 Reset_Boot

Resets the sensor. This command has no parameters.

R.2 Get_Info

Retrieve some information about the sensor.

Parameter: String SA_ArticleNumber

SA_ArticleNumber

Direction: Up

Description: Article number of the sensor.

Parameter: String SA_SerialNumber

SA_SerialNumber

Direction: Up

Description: Serial number of the sensor.

Parameter: String SA_Option

SA_Option

Direction: Up

Description: Option of the sensor.

Parameter: double SA_Range

SA_Range

Direction: Up

Unit: mm

Description: Range of the sensor.

Parameter: int SA_Reserve_1

SA_Reserve_1

Direction: Up

Description: Reserved.

Parameter: String SA_SoftArtBoot

SA_SoftArtBoot

Direction: Up

Description: Sensor software versions.

Parameter: String SA_SoftArtArm

SA_SoftArtArm

Direction: Up

Description: Sensor software versions.

Parameter: String SA_SoftArtDSP

SA_SoftArtDSP

Direction: Up

Description: Sensor software versions.

Parameter: String SA_SoftVerBoot	SA_SoftVerBoot
Direction: Up	
Description: Sensor software versions.	
Parameter: String SA_SoftVerArm	SA_SoftVerArm
Direction: Up	
Description: Sensor software versions.	
Parameter: String SA_SoftVerDSP	SA_SoftVerDSP
Direction: Up	
Description: Sensor software versions.	

R.3 Dat_Out_Off

Switch off data output from sensor.

R.4 Dat_Out_On

Switch on data output from sensor.

R.5 Choose_MeasProg

Select the measurement program of sensor.

Parameter: int SP_MeasProgNumber	SP_MeasProgNumber
Direction: Down	
Valid values:	
0= EdgeHL	
1= EdgeLH	
2= DIA	
3= GAP	
4= SEG_2_4	
5= MULTISEG	
6= USER1	
7= USER2	
8= USER3	
9= USER4	
Description: Measure program of sensor.	

R.6 Switch_Edge

Select the edges to measure.

Parameter: int SP_FrontEdge_Seg1	SP_FrontEdge_Seg1
Direction: Down	
Valid values:	
Minimum: 0	
Maximum: 80	
Description: Front edge of segment 1.	

Parameter: int SP_FrontEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 2.	SP_FrontEdge_Seg2
Parameter: int SP_FrontEdge_Seg3 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 3.	SP_FrontEdge_Seg3
Parameter: int SP_FrontEdge_Seg4 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 4.	SP_FrontEdge_Seg4
Parameter: int SP_RearEdge_Seg1 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 1.	SP_RearEdge_Seg1
Parameter: int SP_RearEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 2.	SP_RearEdge_Seg2
Parameter: int SP_RearEdge_Seg3 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 3.	SP_RearEdge_Seg3
Parameter: int SP_RearEdge_Seg4 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 4.	SP_RearEdge_Seg4

R.7 Read_OptionData

Read the option data from sensor.

Parameter: int SA_MeasProgNumber Direction: Up Valid values: 0 to 5= Standard 6 to 9= User defined Description: Measure program of sensor.	SA_MeasProgNumber
Parameter: int SA_Language Direction: Up Valid values: 0= German 1= English Description: Language of sensor.	SA_Language
Parameter: int SA_DispmasUnit Direction: Up Valid values: 0= mm 1= inch Description: Display measurement unit of sensor.	SA_DispmasUnit
Parameter: int SA_ErrorHandler Direction: Up Valid values: 0= error output 1= retain last value Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.	SA_ErrorHandler
Parameter: int SA_SerialOutFormat Direction: Up Valid values: 0= binary 1= ASCII Description: Serial output format of sensor.	SA_SerialOutFormat
Parameter: int SA_Ext_LaserSwitch Direction: Up Valid values: 0= not active 1= active Description: Enable/Disable external laser switch.	SA_Ext_LaserSwitch
Parameter: int SA_LaserIntensity Direction: Up Valid values: Minimum: 0 Maximum: 100 Unit: % Description: No effect.	SA_LaserIntensity

Parameter: int SA_Contrast Direction: Up Valid values: Minimum: 0 Maximum: 100 Unit: % Description: Display contrast.	SA_Contrast
Parameter: int SA_EdgeDetectThreshold Direction: Up Valid values: Minimum: 20 Maximum: 90 Unit: % Description: Contrast for edge detection.	SA_EdgeDetectThreshold
Parameter: int SA_Reserve_2 Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SA_Reserve_2
Parameter: int SA_ActiveSerialIf Direction: Up Valid values: 0= RS422 1= RS232 Description: Active serial interface of sensor.	SA_ActiveSerialIf
Parameter: int SA_RS232_Baudrate Direction: Up Valid values: 9600 19200 38400 115200 Unit: Baud Description: Baudrate of RS232 interface of sensor.	SA_RS232_Baudrate
Parameter: int SA_RS232_Parity Direction: Up Valid values: 0= none 1= even 2= odd Description: Parity of RS232 interface of sensor.	SA_RS232_Parity
Parameter: int SA_RS232_StopBits Direction: Up Valid values: 1 2 Description: Stop bits of RS232 interface of sensor.	SA_RS232_StopBits

Parameter: int SA_RS232_TimeoutSend Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS232_TimeoutSend
Parameter: int SA_RS232_TimeoutRecv Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS232_TimeoutRecv
Parameter: int SA_RS422_Baudrate Direction: Up Valid values: 9600 19200 38400 115200 691200 Unit: Baud Description: Baudrate of RS422 interface of sensor.	SA_RS422_Baudrate
Parameter: int SA_RS422_Parity Direction: Up Valid values: 0= none 1= even 2= odd Description: Parity of RS422 interface of sensor.	SA_RS422_Parity
Parameter: int SA_RS422_StopBits Direction: Up Valid values: 1 2 Description: Stop bits of RS422 interface of sensor.	SA_RS422_StopBits
Parameter: int SA_RS422_TimeoutSend Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS422_TimeoutSend
Parameter: int SA_RS422_TimeoutRecv Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SA_RS422_TimeoutRecv

R.8 Read_MeasProgData

Read measurement program data from sensor

Parameter: int SA_UserMeasProgNumber	SA_UserMeasProgNumber
Direction: Up	
Valid values:	
Minimum: 6	
Maximum: 9= User1 to User4	
Description: User measure program of sensor.	
Parameter: String SA_MeasProgName	SA_MeasProgName
Direction: Up	
Description: Measure program name of sensor.	
Parameter: double SA_AnalogOffset	SA_AnalogOffset
Direction: Up	
Valid values:	
Minimum: -10.0	
Maximum: 10.0	
Unit: V	
Description: Analog output offset of sensor.	
Parameter: double SA_AnalogGain	SA_AnalogGain
Direction: Up	
Valid values:	
Minimum: -4.0	
Maximum: 4.0	
Description: Analog output gain of sensor.	
Parameter: double SA_DisplayOffset	SA_DisplayOffset
Direction: Up	
Valid values:	
Minimum: -99.99	
Maximum: 99.99	
Unit: mm	
Description: Display offset of sensor.	
Parameter: double SA_DisplayGain	SA_DisplayGain
Direction: Up	
Valid values:	
Minimum: -2.0	
Maximum: 2.0	
Description: Display gain of sensor.	
Parameter: double SA_UpperLimit	SA_UpperLimit
Direction: Up	
Valid values:	
Minimum: -168.876	
Maximum: 168.876	
Unit: mm	
Description: Upper limit of sensor.	

Parameter: double SA_LowerLimit Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower limit of sensor.	SA_LowerLimit
Parameter: double SA_UpperWarning Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper warning of sensor.	SA_UpperWarning
Parameter: double SA_LowerWarning Direction: Up Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower warning of sensor.	SA_LowerWarning
Parameter: int SA_Reserve_3 Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SA_Reserve_3
Parameter: int SA_MeasMode Direction: Up Valid values: 0= Normal 1= Max_Cont 2= Min_Cont 3= P-P_Cont 4= Max_Trg 5= Min_Trg 6= P-P_Trg 7= SC1_Trg Description: Measure Mode.	SA_MeasMode
Parameter: int SA_Median Direction: Up Valid values: 0= no Median 3 5 7 9 Description: Median over n values.	SA_Median

Parameter: int SA_Average_for_reading Direction: Up Valid values: 1 to 128 sliding 129 to 4096 recursive Description: Averaging mode and number of sensor.	SA_Average_for_reading
Parameter: int SA_Reserve_4 Direction: Up Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SA_Reserve_4
Parameter: int SA_MeasObject Direction: Up Valid values: 1 = EdgeHL 2 = EdgeLH 3 = DIA 4 = GAP 5 = SEG_2_4 6 = MULTISEG Description: Measurement program.	SA_MeasObject
Parameter: int SA_NumberOfSegments Direction: Up Valid values: 1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4 2..4 for MULTISEG Description: Number of segments.	SA_NumberOfSegments
Parameter: int SA_FrontEdge_Seg1 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 1.	SA_FrontEdge_Seg1
Parameter: int SA_FrontEdge_Seg2 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 2.	SA_FrontEdge_Seg2
Parameter: int SA_FrontEdge_Seg3 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 3.	SA_FrontEdge_Seg3

Parameter: int SA_FrontEdge_Seg4 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 4.	SA_FrontEdge_Seg4
Parameter: int SA_Reserve_6 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SA_Reserve_6
Parameter: int SA_Reserve_7 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SA_Reserve_7
Parameter: int SA_RearEdge_Seg1 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 1.	SA_RearEdge_Seg1
Parameter: int SA_RearEdge_Seg2 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 2.	SA_RearEdge_Seg2
Parameter: int SA_Reserve_8 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SA_Reserve_8
Parameter: int SA_Reserve_9 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SA_Reserve_9
Parameter: int SA_Reserve_10 Direction: Up Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SA_Reserve_10

Parameter: double SA_MasterValue SA_MasterValue
Direction: Up
Valid values:
 Minimum: -40.0
 Maximum: 40.0
Unit: mm
Description: Master value of sensor.

R.9 Write_OptionData

Write option data to sensor.

Parameter: int SP_MeasProgNumber SP_MeasProgNumber
Direction: Down
Valid values:
 0 to 5= Standard
 6 to 9= User defined
Description: Measure program of sensor.

Parameter: int SP_Language SP_Language
Direction: Down
Valid values:
 0= German
 1= English
Description: Language of sensor.

Parameter: int SP_DispMeasUnit SP_DispMeasUnit
Direction: Down
Valid values:
 0= mm
 1= inch
Description: Display measurement unit of sensor.

Parameter: int SP_ErrorHandler SP_ErrorHandler
Direction: Down
Valid values:
 0= error output
 1= retain last value
Description: If the sensor cannot measure values, it can output the last valid value or it can output an error values.

Parameter: int SP_SerialOutFormat SP_SerialOutFormat
Direction: Down
Valid values:
 0= binary
 1= ASCII
Description: Serial output format of sensor.

Parameter: int SP_Ext_LaserSwitch SP_Ext_LaserSwitch
Direction: Down
Valid values:
 0= not active
 1= active
Description: Enable/Disable external laser switch.

Parameter: int SP_LaserIntensity Direction: Down Valid values: Minimum: 0 Maximum: 100 Unit: % Description: No effect.	SP_LaserIntensity
Parameter: int SP_Contrast Direction: Down Valid values: Minimum: 0 Maximum: 100 Unit: % Description: Display contrast.	SP_Contrast
Parameter: int SP_EdgeDetectThreshold Direction: Down Valid values: Minimum: 20 Maximum: 90 Unit: % Description: Contrast for edge detection.	SP_EdgeDetectThreshold
Parameter: int SP_Reserve_2 Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SP_Reserve_2
Parameter: int SP_ActiveSerialIf Direction: Down Valid values: 0= RS422 1= RS232 Description: Active serial interface of sensor.	SP_ActiveSerialIf
Parameter: int SP_RS232_Baudrate Direction: Down Valid values: 9600 19200 38400 115200 Unit: Baud Description: Baudrate of RS232 interface of sensor.	SP_RS232_Baudrate
Parameter: int SP_RS232_Parity Direction: Down Valid values: 0= none 1= even 2= odd Description: Parity of RS232 interface of sensor.	SP_RS232_Parity

Parameter: int SP_RS232_StopBits Direction: Down Valid values: 1 2 Description: Stop bits of RS232 interface of sensor.	SP_RS232_StopBits
Parameter: int SP_RS232_TimeoutSend Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS232_TimeoutSend
Parameter: int SP_RS232_TimeoutRecv Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS232_TimeoutRecv
Parameter: int SP_RS422_Baudrate Direction: Down Valid values: 9600 19200 38400 115200 691200 Unit: Baud Description: Baudrate of RS422 interface of sensor.	SP_RS422_Baudrate
Parameter: int SP_RS422_Parity Direction: Down Valid values: 0= none 1= even 2= odd Description: Parity of RS422 interface of sensor.	SP_RS422_Parity
Parameter: int SP_RS422_StopBits Direction: Down Valid values: 1 2 Description: Stop bits of RS422 interface of sensor.	SP_RS422_StopBits
Parameter: int SP_RS422_TimeoutSend Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: No effect.	SP_RS422_TimeoutSend

Parameter: int SP_RS422_TimeoutRecv

SP_RS422_TimeoutRecv

Direction: Down

Valid values:

Minimum: 0

Maximum: 65535

Description: No effect.

R.10 Write_MeasProgData

Write measurement program data to sensor.

Parameter: int SP_UserMeasProgNumber

SP_UserMeasProgNumber

Direction: Down

Valid values:

Minimum: 6

Maximum: 9= User1 to User4

Description: User measure program of sensor.

Parameter: String SP_MeasProgName

SP_MeasProgName

Direction: Down

Description: Measure program name of sensor.

Parameter: double SP_AnalogOffset

SP_AnalogOffset

Direction: Down

Valid values:

Minimum: -10.0

Maximum: 10.0

Unit: V

Description: Analog output offset of sensor.

Parameter: double SP_AnalogGain

SP_AnalogGain

Direction: Down

Valid values:

Minimum: -4.0

Maximum: 4.0

Description: Analog output gain of sensor.

Parameter: double SP_DisplayOffset

SP_DisplayOffset

Direction: Down

Valid values:

Minimum: -99.99

Maximum: 99.99

Unit: mm

Description: Display offset of sensor.

Parameter: double SP_DisplayGain

SP_DisplayGain

Direction: Down

Valid values:

Minimum: -2.0

Maximum: 2.0

Description: Display gain of sensor.

Parameter: double SP_UpperLimit Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper limit of sensor.	SP_UpperLimit
Parameter: double SP_LowerLimit Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower limit of sensor.	SP_LowerLimit
Parameter: double SP_UpperWarning Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Upper warning of sensor.	SP_UpperWarning
Parameter: double SP_LowerWarning Direction: Down Valid values: Minimum: -168.876 Maximum: 168.876 Unit: mm Description: Lower warning of sensor.	SP_LowerWarning
Parameter: int SP_Reserve_3 Direction: Down Valid values: Minimum: 0 Maximum: 65535 Description: Reserved.	SP_Reserve_3
Parameter: int SP_MeasMode Direction: Down Valid values: 0= Normal 1= Max_Cont 2= Min_Cont 3= P-P_Cont 4= Max_Trg 5= Min_Trg 6= P-P_Trg 7= SC1_Trg Description: Measure Mode.	SP_MeasMode

Parameter: int SP_Median

Direction: Down

Valid values:

0= no Median
3
5
7
9

Description: Median over n values.

SP_Median

Parameter: int SP_Average_for_reading

Direction: Down

Valid values:

1 to 128 sliding
129 to 4096 recursive

Description: Averaging mode and number of sensor.

SP_Average_for_reading

Parameter: int SP_Reserve_4

Direction: Down

Valid values:

Minimum: 0
Maximum: 65535

Description: Reserved.

SP_Reserve_4

Parameter: int SP_MeasObject

Direction: Down

Valid values:

1= EdgeHL
2= EdgeLH
3= DIA
4= GAP
5= SEG_2_4
6= MULTISEG

Description: Measurement program.

SP_MeasObject

Parameter: int SP_NumberOfSegments

Direction: Down

Valid values:

1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4
2..4 for MULTISEG

Description: Number of segments.

SP_NumberOfSegments

Parameter: int SP_FrontEdge_Seg1

Direction: Down

Valid values:

Minimum: 0
Maximum: 80

Description: Front edge of segment 1.

SP_FrontEdge_Seg1

Parameter: int SP_FrontEdge_Seg2

Direction: Down

Valid values:

Minimum: 0
Maximum: 80

Description: Front edge of segment 2.

SP_FrontEdge_Seg2

Parameter: int SP_FrontEdge_Seg3 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 3.	SP_FrontEdge_Seg3
Parameter: int SP_FrontEdge_Seg4 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Front edge of segment 4.	SP_FrontEdge_Seg4
Parameter: int SP_Reserve_6 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SP_Reserve_6
Parameter: int SP_Reserve_7 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Reserved.	SP_Reserve_7
Parameter: int SP_RearEdge_Seg1 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 1.	SP_RearEdge_Seg1
Parameter: int SP_RearEdge_Seg2 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 2.	SP_RearEdge_Seg2
Parameter: int SP_RearEdge_Seg3 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 3.	SP_RearEdge_Seg3
Parameter: int SP_RearEdge_Seg4 Direction: Down Valid values: Minimum: 0 Maximum: 80 Description: Rear edge of segment 4.	SP_RearEdge_Seg4

Parameter: int SP_Reserve_9

SP_Reserve_9

Direction: Down

Valid values:

Minimum: 0

Maximum: 80

Description: Reserved.

Parameter: int SP_Reserve_10

SP_Reserve_10

Direction: Down

Valid values:

Minimum: 0

Maximum: 80

Description: Reserved.

Parameter: double SP_MasterValue

SP_MasterValue

Direction: Down

Valid values:

Minimum: -40.0

Maximum: 40.0

Unit: mm

Description: Master value of sensor.

R.11 Save_OptionData

Save the option data to flash.

R.12 Save_MeasProgData

Save the measure program data to flash.

R.13 Triggermode_Reset

Resets the values at trigger mode.

R.14 Triggermode_Trigger

Activate output in trigger mode.

R.15 Set_LightRef

Set the light reference.

R.16 Reset_LightRef

Reset the light reference.

R.17 Read_MinMax

Read the minimum and maximum values from sensor.

Parameter: `int SA_MinRaw` `SA_MinRaw`
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Minimum raw value.

Parameter: `int SA_MaxRaw` `SA_MaxRaw`
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Maximum raw value.

Parameter: `int SA_MinScaled` `SA_MinScaled`
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 40.0
Unit: mm
Description: Minimum scaled value.

Parameter: `int SA_MaxScaled` `SA_MaxScaled`
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 40.0
Unit: mm
Description: Maximum scaled value.

R.18 Read_MinMaxReset

Read the minimum and maximum values from sensor and reset the values.

Parameter: `int SA_MinRaw` `SA_MinRaw`
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Minimum raw value.

Parameter: `int SA_MaxRaw` `SA_MaxRaw`
Direction: Up
Valid values:
 Minimum: 0
 Maximum: 65535
Description: Maximum raw value.

Parameter: int SA_MinScaled SA_MinScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 40.0
Unit: mm
Description: Minimum scaled value.

Parameter: int SA_MaxScaled SA_MaxScaled
Direction: Up
Valid values:
 Minimum: 0.0
 Maximum: 40.0
Unit: mm
Description: Maximum scaled value.

R.19 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_NumberOfSegments IP_NumberOfSegments
Direction: Down
Valid values:
 1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4
 2..4 for MULTISEG
Description: Tells the driver the number of segments measured by the sensor.

R.20 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_NumberOfSegments IA_NumberOfSegments
Direction: Up
Valid values:
 1 for EdgeHL, EdgeLH, DIA, GAP and SEG_2_4
 2..4 for MULTISEG
Description: Number of segments used by driver.

S Commands for ESC4912

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

TCP/IP (native).

S.1 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz

Description: Samplerate index

S.2 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz

Description: Samplerate index.

S.3 Set_Trigger (TRG)

Activate/disable the trigger at controller.

Parameter: int SP_TrgMode

SP_TrgMode

Direction: Down

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

S.4 Get_Trigger (TRG?)

Retrieve the trigger mode at controller.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

S.5 Get_Measure (GMD)

Retrieve one value from controller, even if trigger is active.

S.6 Set_AvrType (AVT)

Set the averaging type at controller.

Parameter: int SP_AvrType

SP_AvrType

Direction: Down

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Averaging type.

S.7 Get_AvrType (AVT?)

Retrieve the averaging type at controller.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Averaging type at controller.

S.8 Set_AvrNbr (AVN)

Set the averaging number at controller.

Parameter: int SP_AvrNbr

SP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number.

S.9 Get_AvrNbr (AVN?)

Retrieve the averaging number at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number at controller.

S.10 ChannelStatus (CHS)

Retrieve the available channels at controller.

Parameter: int SA_ChExist1..16

SA_ChExist1..16

Direction: Up

Valid values:

0= FALSE

1= TRUE

Description: Channel 1 to 16 is available at controller.

S.11 Set_Channel (CHT)

Set the channels to transmit from controller.

Parameter: int SP_ChTransmit1..16

SP_ChTransmit1..16

Direction: Down

Valid values:

0= no

1= yes

Description: Channel 1 to 16 is transmitted.

S.12 Get_Channel (CHT?)

Get the channels transmitted from controller.

Parameter: int SA_ChTransmit1..16

SA_ChTransmit1..16

Direction: Up

Valid values:

0= no

1= yes

Description: Channel 1 to 16 is transmitted.

S.13 Set_LinMode (LIN)

Set the linearisation mode for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 16

Description: Channels to be linearized.

Parameter: int SP_LinMode

SP_LinMode

Direction: Down

Valid values:

0= off

1= offset correction

2= 2 point linearization

3= 3 point linearization

4= 5 point linearization

Description: Linearisation mode.

S.14 Get_LinMode (LIN?)

Retrieve the linearisation mode for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SA_LinMode1..16

SA_LinMode1..16

Direction: Down

Valid values:

0= off

1= offset correction

2= 2 point linearization

3= 3 point linearization

4= 5 point linearization

Description: Linearisation mode for channel 1 to 16.

S.15 Set_LinPoint (SLP)

Set a linearisation point for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chan SP_Chan
Direction: Down
Valid values:
 Minimum: 1
 Maximum: 16
Description: Channel to be set for.

Parameter: int SP_LinPos SP_LinPos
Direction: Down
Valid values:
 1 = at 10%
 3 = at 30%
 5 = at 50%
 7 = at 70%
 9 = at 90%
Description: Linearisation position to be set for.

S.16 Get_LinPoint (GLP)

Get a linearisation point for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chan SP_Chan
Direction: Down
Valid values:
 Minimum: 1
 Maximum: 16
Description: Channel to be get for.

Parameter: int SP_LinPos SP_LinPos
Direction: Down
Valid values:
 1 = at 10%
 3 = at 30%
 5 = at 50%
 7 = at 70%
 9 = at 90%
Description: Linearisation position to be get for.

Parameter: double SA_LinPoint SA_LinPoint
Direction: Up
Description: Linearisation point.

S.17 Get_Status (STS)

Retrieve detailed information about the controller.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz

Description: Samplerate index.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

- 0= off
- 1= Moving average
- 2= Mean (arithmetic)
- 3= Median

Description: Averaging type at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number at controller.

Parameter: int SA_ChExist1..16

SA_ChExist1..16

Direction: Up

Valid values:

- 0= FALSE
- 1= TRUE

Description: Channel 1 to 16 is available at controller.

Parameter: int SA_ChTransmit1..16

SA_ChTransmit1..16

Direction: Up

Valid values:

- 0= no
- 1= yes

Description: Channel 1 to 16 is transmitted.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

- 0= disabled
- 1= active

Description: Trigger active/disabled.

Parameter: int SA_LinMode1..16

SA_LinMode1..16

Direction: Up

Valid values:

- 0= off
- 1= offset correction
- 2= 2 point linearization
- 3= 3 point linearization
- 4= 5 point linearization

Description: Linearisation mode for channel 1 to 16.

S.18 Get_Version (VER)

Retrieve the sensor software version.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Software version of the controller.

S.19 Save_Setup (SSU)

Save the current setup of controller to flash.

S.20 Read_Setup (RSU)

Read the setup from controller flash.

S.21 Factory_Defaults (FDE)

Restore the controller to factory defaults.

To save the default values call Save_Setup too. The new parameters are returned.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz

Description: Samplerate index.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Averaging type at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number at controller.

Parameter: int SA_ChExist1..16

SA_ChExist1..16

Direction: Up

Valid values:

0= FALSE

1= TRUE

Description: Channel 1 to 16 is available at controller.

Parameter: int SA_ChTransmit1..16

SA_ChTransmit1..16

Direction: Up

Valid values:

0= no

1= yes

Description: Channel 1 to 16 is transmitted.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

Parameter: int SA_LinMode1..16

SA_LinMode1..16

Direction: Up

Valid values:

0= off

1= offset correction

2= 2 point linearization

3= 3 point linearization

4= 5 point linearization

Description: Linearisation mode for channel 1 to 16.

S.22 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_Range1..16

IP_Range1..16

Direction: Down

Valid values:

Minimum: 0

Unit: μm or mm

Description: Tells the driver the range of sensor for channel 1 to 16. It is used to scale the raw sensor values into μm or mm. If it is zero, no scaling is done.

Parameter: int IP_ChExist1..16

IP_ChExist1..16

Direction: Down

Valid values:

0= FALSE

1= TRUE

Description: Tells the driver if channel 1 to 16 is available.

Parameter: int IP_ChTransmit1..16

IP_ChTransmit1..16

Direction: Down

Valid values:

0= no

1= yes

Description: Tells the driver if channel 1 to 16 is transmitted.

Parameter: int IP_AvrType

IP_AvrType

Direction: Down

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Tells the driver the averaging type of the sensor/controller.

Parameter: int IP_AvrNbr

IP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Tells the driver the averaging number of the sensor/controller.

S.23 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_Range1..16

IA_Range1..16

Direction: Up

Valid values:

Minimum: 0

Unit: μm or mm

Description: The range of sensor for channel 1 to 16, used by driver to scale values into μm or mm.

Parameter: int IA_ChExist1..16

IA_ChExist1..16

Direction: Up

Valid values:

0= FALSE

1= TRUE

Description: Setting used by driver if channel 1 to 16 is available.

Parameter: int IA_ChTransmit1..16

IA_ChTransmit1..16

Direction: Up

Valid values:

0= no

1= yes

Description: Setting used by driver if channel 1 to 16 is transmitted.

Parameter: int IA_AvrType

IA_AvrType

Direction: Up

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Setting used by driver of the averaging type.

Parameter: int IA_AvrNbr

IA_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Setting used by driver for averaging number.

T Commands for DT6100

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

TCP/IP (native).

T.1 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index

T.2 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index.

T.3 Set_Trigger (TRG)

Activate/disable the trigger at sensor.

Parameter: int SP_TrgMode

SP_TrgMode

Direction: Down

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

T.4 Get_Trigger (TRG?)

Retrieve the trigger mode at sensor.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

T.5 Get_Measure (GMD)

Retrieve one value from sensor, even if trigger is active.

T.6 Set_Coefficient (SCO)

Set coefficient at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_CoeffParam

SP_CoeffParam

Direction: Down

Valid values:

Minimum: 0

Maximum: 16

Description: Coefficient parameter.

Parameter: double SP_Coeffizient

SP_Coeffizient

Direction: Down

Valid values:

Minimum: 0

Maximum: 0xfffff

Description: Coefficient.

T.7 Get_Coefficient (GCO)

Get coefficient from controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_CoeffParam

SP_CoeffParam

Direction: Down

Valid values:

Minimum: 0

Maximum: 16

Description: Coefficient parameter.

Parameter: double SA_Coeffizient

SA_Coeffizient

Direction: Up

Description: Coefficient.

T.8 Set_AvrType (AVT)

Set the averaging type at controller.

Parameter: int SP_AvrType

SP_AvrType

Direction: Down

Valid values:

0 = off

1 = Moving average

2 = Mean (arithmetic)

3 = Median

Description: Averaging type.

T.9 Get_AvrType (AVT?)

Retrieve the averaging type at controller.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

0 = off

1 = Moving average

2 = Mean (arithmetic)

3 = Median

Description: Averaging type at controller.

T.10 Set_AvrNbr (AVN)

Set the averaging number at controller.

Parameter: int SP_AvrNbr

SP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number.

T.11 Get_AvrNbr (AVN?)

Retrieve the averaging number at controller.

Parameter: int SA_AvrNbr SA_AvrNbr
Direction: Up
Valid values:
 Minimum: 2
 Maximum: 8
Description: Averaging number at controller.

T.12 Set_LinMode (LIN)

Set the linearisation mode for sensor.

Parameter: int SP_LinMode SP_LinMode
Direction: Down
Valid values:
 0= off
 1= offset correction
 2= 2 point linearization
 3= 3 point linearization
 4= 5 point linearization
 5= 10 point linearization
Description: Linearisation mode.

T.13 Get_LinMode (LIN?)

Retrieve the linearisation mode for sensor.

Parameter: int SA_LinMode SA_LinMode
Direction: Up
Valid values:
 0= off
 1= offset correction
 2= 2 point linearization
 3= 3 point linearization
 4= 5 point linearization
 5= 10 point linearization
Description: Linearisation mode.

T.14 Set_LinPoint (SLP)

Set a linearisation point at sensor.

Parameter: int SP_LinPos SP_LinPos
Direction: Down
Valid values:
 1= at 10%
 2= at 20%

3= at 30%
 4= at 40%
 5= at 50%
 6= at 60%
 7= at 70%
 8= at 80%
 9= at 90%
 10= at 100%

Description: Linearisation position to be set for.

T.15 Get_LinPoint (GLP)

Get a linearisation point at sensor.

Parameter: int SP_LinPos

SP_LinPos

Direction: Down

Valid values:

1= at 10%
 2= at 20%
 3= at 30%
 4= at 40%
 5= at 50%
 6= at 60%
 7= at 70%
 8= at 80%
 9= at 90%
 10= at 100%

Description: Linearisation position to be get for.

Parameter: double SA_LinPoint

SA_LinPoint

Direction: Up

Description: Linearisation point.

T.16 Get_Status (STS)

Retrieve detailed information about the sensor.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= 2.60 Hz
 1= 5.21 Hz
 2= 10.42 Hz
 3= 15.63 Hz
 4= 26.04 Hz
 5= 31.25 Hz
 6= 52.08 Hz
 7= 62.50 Hz
 8= 104.17 Hz
 9= 520.83 Hz
 10= 1041.67 Hz
 11= 2083.33 Hz
 12= 3906.25 Hz
 13= 7812.50 Hz

Description: Samplerate index.

Parameter: int SA_AvrType Direction: Up Valid values: 0= off 1= Moving average 2= Mean (arithmetic) 3= Median Description: Averaging type at controller.	SA_AvrType
Parameter: int SA_AvrNbr Direction: Up Valid values: Minimum: 2 Maximum: 8 Description: Averaging number at controller.	SA_AvrNbr
Parameter: int SA_TrgMode Direction: Up Valid values: 0= disabled 1= active Description: Trigger active/disabled.	SA_TrgMode
Parameter: int SA_LinMode Direction: Up Valid values: 0= off 1= offset correction 2= 2 point linearization 3= 3 point linearization 4= 5 point linearization 5= 10 point linearization Description: Linearisation mode.	SA_LinMode

T.17 Get_Version (VER)

Retrieve the sensor software version.

Parameter: String SA_Version Direction: Up Description: Software version of the controller.	SA_Version
--	------------

T.18 Save_Setup (SSU)

Save the current setup of controller to flash.

T.19 Read_Setup (RSU)

Read the setup from controller flash.

T.20 Factory_Defaults (FDE)

Restore the sensor to factory defaults.

To save the default values call Save_Setup too. The new parameters are returned.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

- 0= off
- 1= Moving average
- 2= Mean (arithmetic)
- 3= Median

Description: Averaging type at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number at controller.

Parameter: int SA_TriggerMode

SA_TriggerMode

Direction: Up

Valid values:

- 0= disabled
- 1= active

Description: Trigger active/disabled.

Parameter: int SA_LinMode

SA_LinMode

Direction: Up

Valid values:

- 0= off
- 1= offset correction
- 2= 2 point linearization
- 3= 3 point linearization
- 4= 5 point linearization
- 5= 10 point linearization

Description: Linearisation mode.

T.21 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: int IP_AvrType	IP_AvrType
Direction: Down	
Valid values:	
0= off	
1= Moving average	
2= Mean (arithmetic)	
3= Median	
Description: Tells the driver the averaging type of the sensor/controller.	
Parameter: int IP_AvrNbr	IP_AvrNbr
Direction: Down	
Valid values:	
Minimum: 2	
Maximum: 8	
Description: Tells the driver the averaging number of the sensor/controller.	

T.22 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: int IA_AvrType	IA_AvrType
Direction: Up	
Valid values:	
0= off	
1= Moving average	
2= Mean (arithmetic)	
3= Median	
Description: Setting used by driver of the averaging type.	
Parameter: int IA_AvrNbr	IA_AvrNbr
Direction: Down	
Valid values:	
Minimum: 2	
Maximum: 8	
Description: Setting used by driver for averaging number.	

U Commands for KSS6380

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

TCP/IP (native).

U.1 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

0= 26 Hz

1= 104 Hz

2= 520 Hz

3= 1040 Hz

Description: Samplerate index

U.2 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

0= 26 Hz

1= 104 Hz

2= 520 Hz

3= 1040 Hz

Description: Samplerate index.

U.3 Set_Trigger (TRG)

Activate/disable the trigger at controller.

Parameter: int SP_TrgMode

SP_TrgMode

Direction: Down

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

U.4 Get_Trigger (TRG?)

Retrieve the trigger mode at controller.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

0= disabled

1= active

Description: Trigger active/disabled.

U.5 Get_Measure (GMD)

Retrieve one value from controller, even if trigger is active.

U.6 Set_TempCoeff (STC)

Set temperature coefficient for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chان

SP_Chان

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Channel to be set for.

Parameter: int SP_TempCoeffParam

SP_TempCoeffParam

Direction: Down

Valid values:

Minimum: 0

Maximum: 16

Description: Temperature coefficient parameter.

Parameter: double SP_TemperatureCoefficient

SP_TemperatureCoefficient

Direction: Down

Valid values:

Minimum: 0

Maximum: 0xfffff

Description: Temperature coefficient.

U.7 Get_TempCoeff (GTC)

Get temperature coefficient for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chان

SP_Chان

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Channel to be set for.

Parameter: int SP_TempCoeffParam SP_TempCoeffParam
Direction: Down
Valid values:
 Minimum: 0
 Maximum: 16
Description: Temperature coefficient parameter.

Parameter: double SA_TemperatureCoeffizient SA_TemperatureCoeffizient
Direction: Up
Description: Temperature coefficient.

U.8 Get_UnlinEddyVal (GUE)

Get unlinearized eddy value from controller. This is an internal command. It should not be used by the customer.

Parameter: double SA_UnlinearizedEddyValue SA_UnlinearizedEddyValue
Direction: Up
Description: Unlinearized eddy value .

U.9 Set_AvrType (AVT)

Set the averaging type at controller.

Parameter: int SP_AvrType SP_AvrType
Direction: Down
Valid values:
 0 = off
 1 = Moving average
Description: Averaging type.

U.10 Get_AvrType (AVT?)

Retrieve the averaging type at controller.

Parameter: int SA_AvrType SA_AvrType
Direction: Up
Valid values:
 0 = off
 1 = Moving average
Description: Averaging type at controller.

U.11 Set_AvrNbr (AVN)

Set the averaging number at controller.

Parameter: int SP_AvrNbr SP_AvrNbr
Direction: Down
Valid values:
 Minimum: 2
 Maximum: 4
Description: Averaging number.

U.12 Get_AvrNbr (AVN?)

Retrieve the averaging number at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 4

Description: Averaging number at controller.

U.13 ChannelStatus (CHS)

Retrieve the available channels at controller.

Parameter: int SA_ChExist1..4

SA_ChExist1..4

Direction: Up

Valid values:

0 = Channel not available

1 = Measured channel

Description: Channel 1 to 4 is available at controller.

U.14 Set_Channel (CHT)

Set the channels to transmit from controller.

Parameter: int SP_ChTransmit1..4

SP_ChTransmit1..4

Direction: Down

Valid values:

0 = no

1 = yes

Description: Channel 1 to 4 is transmitted.

U.15 Get_Channel (CHT?)

Get the channels transmitted from controller.

Parameter: int SA_ChTransmit1..4

SA_ChTransmit1..4

Direction: Up

Valid values:

0 = no

1 = yes

Description: Channel 1 to 4 is transmitted.

U.16 Set_LinPoint (SLP)

Set a linearisation point for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Channel to be set for.

Parameter: int SP_LinPos

SP_LinPos

Direction: Down

Valid values:

0= at 0%

1= at 5%

2= at 10%

3= at 15%

4= at 20%

5= at 25%

6= at 30%

7= at 35%

8= at 40%

9= at 45%

10= at 50%

11= at 55%

12= at 60%

13= at 65%

14= at 70%

15= at 75%

16= at 80%

17= at 85%

18= at 90%

19= at 95%

20= at 100%

Description: Linearisation position to be set for.

U.17 Get_LinPoint (GLP)

Get a linearisation point for a channel at controller. This is an internal command. It should not be used by the customer.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 2

Description: Channel to be get for.

Parameter: int SP_LinPos

SP_LinPos

Direction: Down**Valid values:**

0= at 0%
1= at 5%
2= at 10%
3= at 15%
4= at 20%
5= at 25%
6= at 30%
7= at 35%
8= at 40%
9= at 45%
10= at 50%
11= at 55%
12= at 60%
13= at 65%
14= at 70%
15= at 75%
16= at 80%
17= at 85%
18= at 90%
19= at 95%
20= at 100%

Description: Linearisation position to be get for.**Parameter:** double SA_LinPoint

SA_LinPoint

Direction: Up**Description:** Linearisation point.

U.18 Get_Status (STS)

Retrieve detailed information about the controller.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up**Valid values:**

0= 26 Hz
1= 104 Hz
2= 520 Hz
3= 1040 Hz

Description: Samplerate index.**Parameter:** int SA_AvrType

SA_AvrType

Direction: Up**Valid values:**

0= off
1= Moving average

Description: Averaging type at controller.

Parameter: int SA_AvrNbr SA_AvrNbr
Direction: Up
Valid values:
 Minimum: 2
 Maximum: 4
Description: Averaging number at controller.

Parameter: int SA_ChExist1..4 SA_ChExist1..4
Direction: Up
Valid values:
 0 = Channel not available
 1 = Measured channel
Description: Channel 1 to 4 is available at controller.

Parameter: int SA_ChTransmit1..4 SA_ChTransmit1..4
Direction: Up
Valid values:
 0 = no
 1 = yes
Description: Channel 1 to 4 is transmitted.

Parameter: int SA_TrgMode SA_TrgMode
Direction: Up
Valid values:
 0 = disabled
 1 = active
Description: Trigger active/disabled.

U.19 Get_Version (VER)

Retrieve the sensor software version.

Parameter: String SA_Version SA_Version
Direction: Up
Description: Software version of the controller.

U.20 Save_Setup (SSU)

Save the current setup of controller to flash.

U.21 Read_Setup (RSU)

Read the setup from controller flash.

U.22 Factory_Defaults (FDE)

Restore the controller to factory defaults.

To save the default values call Save_Setup too. The new parameters are returned.

Parameter: int SA_SRIndex SA_SRIndex
Direction: Up
Valid values:
0= 26 Hz
1= 104 Hz
2= 520 Hz
3= 1040 Hz
Description: Samplerate index.

Parameter: int SA_AvrType SA_AvrType
Direction: Up
Valid values:
0= off
1= Moving average
Description: Averaging type at controller.

Parameter: int SA_AvrNbr SA_AvrNbr
Direction: Up
Valid values:
Minimum: 2
Maximum: 4
Description: Averaging number at controller.

Parameter: int SA_ChExist1..4 SA_ChExist1..4
Direction: Up
Valid values:
0= Channel not available
1= Measured channel
Description: Channel 1 to 4 is available at controller.

Parameter: int SA_ChTransmit1..4 SA_ChTransmit1..4
Direction: Up
Valid values:
0= no
1= yes
Description: Channel 1 to 4 is transmitted.

Parameter: int SA_TrgMode SA_TrgMode
Direction: Up
Valid values:
0= disabled
1= active
Description: Trigger active/disabled.

U.23 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_Range1..4

IP_Range1..4

Direction: Down

Valid values:

Minimum: 0

Unit: μm or mm

Description: Tells the driver the range of sensor for channel 1 to 4. It is used to scale the raw sensor values into μm or mm. If it is zero, no scaling is done.

Parameter: int IP_ChTransmit1..4

IP_ChTransmit1..4

Direction: Down

Valid values:

0 = no

1 = yes

Description: Tells the driver if channel 1 to 4 is transmitted.

Parameter: int IP_AvrType

IP_AvrType

Direction: Down

Valid values:

0 = off

1 = Moving average

Description: Tells the driver the averaging type of the sensor/controller.

Parameter: int IP_AvrNbr

IP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 4

Description: Tells the driver the averaging number of the sensor/controller.

U.24 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_Range1..4

IA_Range1..4

Direction: Up

Valid values:

Minimum: 0

Unit: μm or mm

Description: The range of sensor for channel 1 to 4, used by driver to scale values into μm or mm.

<p>Parameter: int IA_ChTransmit1..4</p> <p>Direction: Up</p> <p>Valid values:</p> <p>0= no</p> <p>1= yes</p> <p>Description: Setting used by driver if channel 1 to 4 is transmitted.</p>	IA_ChTransmit1..4
<p>Parameter: int IA_AvrType</p> <p>Direction: Up</p> <p>Valid values:</p> <p>0= off</p> <p>1= Moving average</p> <p>Description: Setting used by driver of the averaging type.</p>	IA_AvrType
<p>Parameter: int IA_AvrNbr</p> <p>Direction: Down</p> <p>Valid values:</p> <p>Minimum: 2</p> <p>Maximum: 4</p> <p>Description: Setting used by driver for averaging number.</p>	IA_AvrNbr

V Commands for DT6500

See sensor manual for detailed description of sensor commands.

This sensor supports following interfaces:

TCP/IP (native).

V.1 Set_SRIndex (SRA)

Set the samplerate for data acquisition.

Parameter: int SP_SRIndex

SP_SRIndex

Direction: Down

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index

V.2 Get_SRIndex (SRA?)

Get the current samplerate.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index.

V.3 Set_Trigger (TRG)

Activate/disable the trigger at controller.

Parameter: int SP_TrgMode

SP_TrgMode

Direction: Down

Valid values:

0= Off

1= Rising edge

2= High level

3= Gate at rising edge

Description: Trigger active/disabled.

V.4 Get_Trigger (TRG?)

Retrieve the trigger mode at controller.

Parameter: int SA_TrgMode

SA_TrgMode

Direction: Up

Valid values:

0= Off

1= Rising edge

2= High level

3= Gate at rising edge

Description: Trigger active/disabled.

V.5 Get_Measure (GMD)

Retrieve one value from controller, even if trigger is active. To get the values from MEDAQLib, use the functions Poll or TransferData.

V.6 Set_AvrType (AVT)

Set the averaging type at controller.

Parameter: int SP_AvrType

SP_AvrType

Direction: Down

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Averaging type.

V.7 Get_AvrType (AVT?)

Retrieve the averaging type at controller.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Averaging type at controller.

V.8 Set_AvrNbr (AVN)

Set the averaging number at controller.

Parameter: int SP_AvrNbr

SP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number.

V.9 Get_AvrNbr (AVN?)

Retrieve the averaging number at controller.

Parameter: int SA_AvrNbr

SA_AvrNbr

Direction: Up

Valid values:

Minimum: 2

Maximum: 8

Description: Averaging number at controller.

V.10 ChannelStatus (CHS)

Retrieve the available channels at controller.

Parameter: int SA_ChExist1..8

SA_ChExist1..8

Direction: Up

Valid values:

0= Channel not available

1= Measured channel

2= Mathematic channel

Description: Channel 1 to 8 is available at controller.

V.11 Set_Channel (CHT)

Set the channels to transmit from controller.

Parameter: int SP_ChTransmit1..8

SP_ChTransmit1..8

Direction: Down

Valid values:

0= no

1= yes

Description: Channel 1 to 8 is transmitted.

V.12 Get_Channel (CHT?)

Get the channels transmitted from controller.

Parameter: int SA_ChTransmit1..8

SA_ChTransmit1..8

Direction: Up

Valid values:

0= no

1= yes

Description: Channel 1 to 8 is transmitted.

V.13 Set_Display (DIS)

Set the display settings at controller.

Parameter: int SP_ShowChannels

SP_ShowChannels

Direction: Down

Valid values:

0= none

1= all

2= selected

Description: Channels to be displayed.

Parameter: int SP_ShowLinearized

SP_ShowLinearized

Direction: Down

Valid values:

0= off

1= on

Description: Channels should be displayed linearized.

V.14 Get_Display (DIS?)

Retrieve the display settings at controller.

Parameter: int SA_ShowChannels

SA_ShowChannels

Direction: Up

Valid values:

0= none

1= all

2= selected

Description: Channels to be displayed.

Parameter: int SA_ShowLinearized SA_ShowLinearized
Direction: Up
Valid values:
0= off
1= on
Description: Channels should be displayed linearized.

V.15 Set_MathFunction (SMF)

Set mathematic function at controller.

The result of the math function is transmitted like normal sensor data at a specific channel.

Because of the mathematic operations (multiplication, addition) the values can exceed the range of 24 bit. To avoid this, the controller does automatically divide the results by eight (Result= Result/8).

Parameter: int SP_Chan SP_Chan
Direction: Down
Valid values:
Minimum: 1
Maximum: 8
Description: Channels which returns the result.

Parameter: double SP_Offset SP_Offset
Direction: Down
Valid values:
Minimum: -0xfffff
Maximum: 0xfffff
Description: Offset to be added to result.

Parameter: double SP_FactorCh1..8 SP_FactorCh1..8
Direction: Down
Valid values:
Minimum: -9.9
Maximum: 9.9
Description: Multiplication factor for channel 1 to 8.

V.16 Get_MathFunction (GMF)

Get mathematic function from controller.

Parameter: int SP_Chan SP_Chan
Direction: Down
Valid values:
Minimum: 1
Maximum: 8
Description: Channels for which the function should be read.

Parameter: double SA_Offset

SA_Offset

Direction: Up

Valid values:

Minimum: -0xfffff

Maximum: 0xfffff

Description: Offset to be added to result.

Parameter: double SA_FactorCh1..8

SA_FactorCh1..8

Direction: Up

Valid values:

Minimum: -9.9

Maximum: 9.9

Description: Multiplication factor for channel 1 to 8.

V.17 Clr_MathFunction (CMF)

Clears mathimatic function at controller.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 8

Description: Channels for which the function should be cleared.

V.18 Set_LinMode (LIN)

Set the linearisation mode for a channel at controller.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 8

Description: Channels to be linearized.

Parameter: int SP_LinMode

SP_LinMode

Direction: Down

Valid values:

0= off

1= offset correction

2= 2 point linearization

3= 3 point linearization

4= 5 point linearization

5= 10 point linearization

Description: Linearisation mode.

V.19 Get_LinMode (LIN?)

Retrieve the linearisation mode for a channel at controller.

Parameter: int SA_LinMode1..8

SA_LinMode1..8

Direction: Up

Valid values:

- 0= off
- 1= offset correction
- 2= 2 point linearization
- 3= 3 point linearization
- 4= 5 point linearization
- 5= 10 point linearization

Description: Linearisation mode for channel 1 to 8.

V.20 Set_LinPoint (SLP)

Set a linearisation point for a channel at controller.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 8

Description: Channel to be set for.

Parameter: int SP_LinPos

SP_LinPos

Direction: Down

Valid values:

- 1= at 10%
- 2= at 20%
- 3= at 30%
- 4= at 40%
- 5= at 50%
- 6= at 60%
- 7= at 70%
- 8= at 80%
- 9= at 90%
- 10= at 100%

Description: Linearisation position to be set for.

V.21 Get_LinPoint (GLP)

Get a linearisation point for a channel at controller.

Parameter: int SP_Chan

SP_Chan

Direction: Down

Valid values:

Minimum: 1

Maximum: 8

Description: Channel to be get for.

Parameter: int SP_LinPos

SP_LinPos

Direction: Down

Valid values:

- 1 = at 10%
- 2 = at 20%
- 3 = at 30%
- 4 = at 40%
- 5 = at 50%
- 6 = at 60%
- 7 = at 70%
- 8 = at 80%
- 9 = at 90%
- 10 = at 100%

Description: Linearisation position to be get for.

Parameter: double SA_LinPoint

SA_LinPoint

Direction: Up

Description: Linearisation point.

V.22 Get_Status (STS)

Retrieve detailed information about the controller.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0 = 2.60 Hz
- 1 = 5.21 Hz
- 2 = 10.42 Hz
- 3 = 15.63 Hz
- 4 = 26.04 Hz
- 5 = 31.25 Hz
- 6 = 52.08 Hz
- 7 = 62.50 Hz
- 8 = 104.17 Hz
- 9 = 520.83 Hz
- 10 = 1041.67 Hz
- 11 = 2083.33 Hz
- 12 = 3906.25 Hz
- 13 = 7812.50 Hz

Description: Samplerate index.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

- 0 = off
- 1 = Moving average
- 2 = Mean (arithmetic)
- 3 = Median

Description: Averaging type at controller.

Parameter: int SA_AvrNbr Direction: Up Valid values: Minimum: 2 Maximum: 8 Description: Averaging number at controller.	SA_AvrNbr
Parameter: int SA_ChExist1..8 Direction: Up Valid values: 0= Channel not available 1= Measured channel 2= Mathematic channel Description: Channel 1 to 8 is available at controller.	SA_ChExist1..8
Parameter: int SA_ChTransmit1..8 Direction: Up Valid values: 0= no 1= yes Description: Channel 1 to 8 is transmitted.	SA_ChTransmit1..8
Parameter: int SA_TrgMode Direction: Up Valid values: 0= Off 1= Rising edge 2= High level 3= Gate at rising edge Description: Trigger active/disabled.	SA_TrgMode
Parameter: int SA_LinMode1..8 Direction: Up Valid values: 0= off 1= offset correction 2= 2 point linearization 3= 3 point linearization 4= 5 point linearization Description: Linearisation mode for channel 1 to 8.	SA_LinMode1..8
Parameter: int SA_ShowChannels Direction: Up Valid values: 0= none 1= all 2= selected Description: Channels to be displayed.	SA_ShowChannels
Parameter: int SA_ShowLinearized Direction: Up Valid values: 0= off 1= on Description: Channels should be displayed linearized.	SA_ShowLinearized

V.23 Get_Version (VER)

Retrieve the sensor software version.

Parameter: String SA_Version

SA_Version

Direction: Up

Description: Software version of the controller.

V.24 Save_Setup (SSU)

Save the current setup of controller to flash.

V.25 Read_Setup (RSU)

Read the setup from controller flash.

V.26 Factory_Defaults (FDE)

Restore the controller to factory defaults.

To save the default values call Save_Setup too. The new parameters are returned.

Parameter: int SA_SRIndex

SA_SRIndex

Direction: Up

Valid values:

- 0= 2.60 Hz
- 1= 5.21 Hz
- 2= 10.42 Hz
- 3= 15.63 Hz
- 4= 26.04 Hz
- 5= 31.25 Hz
- 6= 52.08 Hz
- 7= 62.50 Hz
- 8= 104.17 Hz
- 9= 520.83 Hz
- 10= 1041.67 Hz
- 11= 2083.33 Hz
- 12= 3906.25 Hz
- 13= 7812.50 Hz

Description: Samplerate index.

Parameter: int SA_AvrType

SA_AvrType

Direction: Up

Valid values:

- 0= off
- 1= Moving average
- 2= Mean (arithmetic)
- 3= Median

Description: Averaging type at controller.

Parameter: int SA_AvrNbr Direction: Up Valid values: Minimum: 2 Maximum: 8 Description: Averaging number at controller.	SA_AvrNbr
Parameter: int SA_ChExist1..8 Direction: Up Valid values: 0= Channel not available 1= Measured channel 2= Mathematic channel Description: Channel 1 to 8 is available at controller.	SA_ChExist1..8
Parameter: int SA_ChTransmit1..8 Direction: Up Valid values: 0= no 1= yes Description: Channel 1 to 8 is transmitted.	SA_ChTransmit1..8
Parameter: int SA_TrgMode Direction: Up Valid values: 0= Off 1= Rising edge 2= High level 3= Gate at rising edge Description: Trigger active/disabled.	SA_TrgMode
Parameter: int SA_LinMode1..8 Direction: Up Valid values: 0= off 1= offset correction 2= 2 point linearization 3= 3 point linearization 4= 5 point linearization Description: Linearisation mode for channel 1 to 8.	SA_LinMode1..8
Parameter: int SA_ShowChannels Direction: Up Valid values: 0= none 1= all 2= selected Description: Channels to be displayed.	SA_ShowChannels
Parameter: int SA_ShowLinearized Direction: Up Valid values: 0= off 1= on Description: Channels should be displayed linearized.	SA_ShowLinearized

V.27 Use_Defaults

This command tells the driver to use default values to operate with sensor data. If some parameters are not specified they are not changed. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_Range1..8

IP_Range1..8

Direction: Down

Valid values:

Minimum: 0

Unit: μm or mm

Description: Tells the driver the range of sensor for channel 1 to 8. It is used to scale the raw sensor values into μm or mm. If it is zero, no scaling is done.

Parameter: int IP_ChExist1..8

IP_ChExist1..8

Direction: Down

Valid values:

0 = Channel not available

1 = Measured channel

2 = Mathematic channel

Description: Tells the driver if channel 1 to 8 is available.

Parameter: int IP_ChTransmit1..8

IP_ChTransmit1..8

Direction: Down

Valid values:

0 = no

1 = yes

Description: Tells the driver if channel 1 to 8 is transmitted.

Parameter: int IP_AvrType

IP_AvrType

Direction: Down

Valid values:

0 = off

1 = Moving average

2 = Mean (arithmetic)

3 = Median

Description: Tells the driver the averaging type of the sensor/controller.

Parameter: int IP_AvrNbr

IP_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Tells the driver the averaging number of the sensor/controller.

V.28 Get_DrvSetting

Returns the current settings of the driver used for operating with sensor data. It is the opposite of Use_Defaults. The sensor is not affected by this command.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_Range1..8

IA_Range1..8

Direction: Up

Valid values:

Minimum: 0

Unit: μm or mm

Description: The range of sensor for channel 1 to 8, used by driver to scale values into μm or mm.

Parameter: int IA_ChExist1..8

IA_ChExist1..8

Direction: Up

Valid values:

0= Channel not available

1= Measured channel

2= Mathematic channel

Description: Setting used by driver if channel 1 to 8 is available.

Parameter: int IA_ChTransmit1..8

IA_ChTransmit1..8

Direction: Up

Valid values:

0= no

1= yes

Description: Setting used by driver if channel 1 to 8 is transmitted.

Parameter: int IA_AvrType

IA_AvrType

Direction: Up

Valid values:

0= off

1= Moving average

2= Mean (arithmetic)

3= Median

Description: Setting used by driver of the averaging type.

Parameter: int IA_AvrNbr

IA_AvrNbr

Direction: Down

Valid values:

Minimum: 2

Maximum: 8

Description: Setting used by driver for averaging number.

W Commands for EncoderIF2004

See IF2004 manual for detailed description of sensor commands.

Attention! The command `Use_Defaults` must be called for correct functionality of encoder. Please see the example at command `Use_Defaults`.

This sensor supports following interfaces:

IF2004 (native).

W.1 Use_Defaults

This command parametrizes the IF2004 PCI card. If same parameters are not specified they are not changed.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double `IP_Range`

`IP_Range`

Direction: Down

Valid values:

Minimum: 0

Description: It is the distance per count of the encoder used by the driver for scaling data.
If it is zero, no scaling is done.

Parameter: int `IP_EncCountMode`

`IP_EncCountMode`

Valid values:

0= Counter without phase discriminator (Trace A is direction, Trace B is pulse, Trace C is load or latch signal).

1= Counter with phase discriminator and 1-fold discriminator.

2= Counter with phase discriminator and 2-fold discriminator.

3= Counter with phase discriminator and 4-fold discriminator.

Direction: Down

Description: The counting mode of the encoder

Parameter: int `IP_EncSwapTraceA_B`

`IP_EncSwapTraceA_B`

Direction: Down

Valid values:

0= no swap

1= swap Trace A and B

Description: Trace A and B are swopped which negates the count direction

Parameter: int `IP_EncInvertTraceA_B`

`IP_EncInvertTraceA_B`

Direction: Down

Valid values:

0= no invert

1= invert Trace A and B

Description: Trace A and B are inverted, for Encoders where Trace C is 1 when Trace A and B are 0

Parameter: int IP_EncLatchSrc

IP_EncLatchSrc

Direction: Down

Valid values:

- 0= Never
- 1..4= Latch with start bit on channel 1 to 4.
- 5= Latch with next reference on Trace C (if unlocked).
- 6= Latch with second reference on Trace C (if unlocked).
- 7= Latch with every reference on Trace C (always).

Description: Specifies when the encoder value should be stored to FIFO (only useful when IP_ChannelNumber is 3)

Parameter: int IP_EncLoadOnRef

IP_EncLoadOnRef

Direction: Down

Valid values:

- 0= never
- 1= Load with next reference on Trace C (if unlocked).
- 2= Load with every reference on Trace C (always).
- 3= Clear with every reference on Trace C (always) and load when count is -1.

Description: Specifies when the encoder value should be changed by IF2004 card.

Example how to parametrize the encoder:

```
DWORD instance= CreateSensorInstance(ENCODER_IF2004);
SetParameterString (instance, "IP_Interface", "IF2004");
SetParameterInt (instance, "IP_CardInstance", 0);
SetParameterInt (instance, "IP_ChannelNumber", 3);
DWORD err= OpenSensor (instance);
/* error handling, if err!=ERR_SUCCESS */

SetParameterString (instance, "S_Command", "Use_Defaults");
/* Set encoder to increment or decrement each full period */
SetParameterInt (instance, "IP_EncCountMode", 1);
/* Each time when a value arrives at first sensor channel, put
the encoder value into FIFO */
SetParameterInt (instance, "IP_EncLatchSrc", 1);
err= SensorCommand (instance);
/* error handling, if err!=ERR_SUCCESS*/
```

W.2 Enc_ClearCounter

Clears the counter (set to 0).

W.3 Enc_LoadCounter

Load the counter from the load register.

W.4 Enc_LatchCounter

Get the recent counter value to the latch register.

W.5 Enc_UnlockTraceC

Unlocks Trace C.

W.6 Enc_IsFirstRef

Checks if Ref (Trace C) was reached since last Unlock.

Parameter: int SA_IsFirstRef

SA_IsFirstRef

Direction: Up

Valid values:

0 = FALSE

1 = TRUE

Description: True, if it is first reference.

W.7 Enc_IsSecondRef

Checks if Ref (Trace C) was reached twice since last Unlock.

Parameter: int SA_IsSecondRef

SA_IsSecondRef

Direction: Up

Valid values:

0 = FALSE

1 = TRUE

Description: True, if it is second reference.

W.8 Enc_GetLatchReg

Get the counter value from the latch register.

Parameter: int SA_LatchReg

SA_LatchReg

Direction: Up

Valid values:

Minimum: 0

Maximum: 65535

Description: Counter value.

W.9 Enc_SetLoadReg

Set the load register for counter.

Parameter: int SP_LoadReg

SP_LoadReg

Direction: Down

Valid values:

Minimum: 0

Maximum: 65535

Description: Counter value.

W.10 IF2004_SystemReset

Make a system reset: Clear FIFO and send a pulse (100 μs) reset line at both sensor connectors.

W.11 IF2004_CheckGate

Checks the actual state of gate.

Parameter: int SA_Gate

SA_Gate

Direction: Up

Valid values:

0= FALSE

1= TRUE

Description: True if gate is set.

W.12 Enc_CheckRef

Checks the actual state of trace C.

Parameter: int SA_Ref

SA_Ref

Direction: Up

Valid values:

0= FALSE

1= TRUE

Description: True if reference (trace C) is set.

W.13 IF2004_SensorReset12

Set the reset line of sensor connector 0+1 to specified value.

Parameter: int SP_SensorReset12

SP_SensorReset12

Direction: Down

Valid values:

0= FALSE

1= TRUE

Description: Value for reset line.

W.14 IF2004_SensorReset34

Set the reset line of sensor connector 2+3 to specified value.

Parameter: int SP_SensorReset34

SP_SensorReset34

Direction: Down

Valid values:

0= FALSE

1= TRUE

Description: Value for reset line.

W.15 Get_FPGAVersion

Get the version of the FPGA on the card.

Parameter: int SA_FPGAVersion

SA_FPGAVersion

Direction: Up

Valid values:

0= V1.1

1= V1.2 (with ILD1700 support)

2= V1.3 (with 1.25 MBaud support)

Description: Version of the FPGA on the card.

X Commands for PCICard_IF2008

See PCICard_IF2008 manual for detailed description of sensor commands.

Attention! The command Use_Defaults must be called if encoder or ADC values should be scaled.

This sensor supports following interfaces:

IF2008 (native).

X.1 Use_Defaults

This command parametrizes the IF2008 PCI card. If same parameters are not specified they are not changed.

Parameters valid for several sensors are not described here but at chapter [A.3.1](#).

The other parameters are:

Parameter: double IP_Encoder1CountFactor Direction: Down Valid for sensor: PCI_CARD_IF2008 Valid values: Minimum: -DBL_MAX (-1.8e+308) Maximum: DBL_MAX (1.8e+308) Default: 1.0 Description: It is the distance per count of the encoder 1 used by the driver for scaling data. If it is zero, no scaling is done.	IP_Encoder1CountFactor
Parameter: double IP_Encoder2CountFactor Direction: Down Valid for sensor: PCI_CARD_IF2008 Valid values: Minimum: -DBL_MAX (-1.8e+308) Maximum: DBL_MAX (1.8e+308) Default: 1.0 Description: It is the distance per count of the encoder 2 used by the driver for scaling data. If it is zero, no scaling is done.	IP_Encoder2CountFactor
Parameter: int IP_ADC1Range Direction: Down Valid for sensor: PCI_CARD_IF2008 Valid values: 0= 0..5V 1= 0..10V 2= +5V 3= +-10V Default: 3 Description: It is range of ADC 1 used by the driver for scaling data.	IP_ADC1Range

Parameter: int IP_ADC2Range

IP_ADC2Range

Direction: Down

Valid for sensor:

PCI_CARD_IF2008

Valid values:

0= 0..5V

1= 0..10V

2= +-5V

3= +-10V

Default: 3

Description: It is range of ADC 2 used by the driver for scaling data.

X.2 Get_DrvSetting

Returns the current settings of the driver used for operating. It is the opposite of Use_Defaults.

Parameters valid for several sensors are not described here but at chapter [A.3.3](#).

The other parameters are:

Parameter: double IA_Encoder1CountFactor

IA_Encoder1CountFactor

Direction: Up

Valid for sensor:

PCI_CARD_IF2008

Description: It is the distance per count of the encoder 1 used by the driver for scaling data. If it is zero, no scaling is done.

Parameter: double IA_Encoder2CountFactor

IA_Encoder2CountFactor

Direction: Up

Valid for sensor:

PCI_CARD_IF2008

Description: It is the distance per count of the encoder 2 used by the driver for scaling data. If it is zero, no scaling is done.

Parameter: int IA_ADC1Range

IA_ADC1Range

Direction: Up

Valid for sensor:

PCI_CARD_IF2008

Description: It is range of ADC 1 used by the driver for scaling data.

Parameter: int IA_ADC2Range

IA_ADC2Range

Direction: Up

Valid for sensor:

PCI_CARD_IF2008

Description: It is range of ADC 2 used by the driver for scaling data.

X.3 Use_Gate

The digital inputs In 1 to In 4 of the card (5V TTL signal) can be used to lock or free the FIFO for data from specific channels.

Parameter: int SP_GateChannel

SP_GateChannel

Direction: Down

Valid values:

- 0= Digital In 1 for Sensor channel 1+2 (Base Board, Connector X1)
- 1= Digital In 1 for Sensor channel 1+2 (Base Board, Connector X1)
- 2= Digital In 2 for Sensor channel 3-6 (Base Board, Connector X2 + Extension Board, Connector X1)
- 3= Digital In 2 for Sensor channel 3-6 (Base Board, Connector X2 + Extension Board, Connector X1)
- 4= Digital In 2 for Sensor channel 3-6 (Base Board, Connector X2 + Extension Board, Connector X1)
- 5= Digital In 2 for Sensor channel 3-6 (Base Board, Connector X2 + Extension Board, Connector X1)
- 6= Digital In 3 for Encoder 1+2
- 7= Digital In 3 for Encoder 1+2
- 8= Digital In 4 for Digital In+RxD, ADC 1+2
- 9= Digital In 4 for Digital In+RxD, ADC 1+2
- 10= Digital In 4 for Digital In+RxD, ADC 1+2
- 11= Digital In 4 for Digital In+RxD, ADC 1+2

Description: Number of the channel to lock by digital in. Because multiple channels are affected by one digital input, the numbers 0..1, 2..5, 6..7 or 8..11 always affects the same digital input.

Parameter: int SP_ActivateGate

SP_ActivateGate

Direction: Down

Valid values:

- 0= off
- 1= on

Description: Specifies if the gate function is enabled or disabled for the specific digital input.

X.4 Set_TimerFrequency

Set the frequency and pulse width for the timers on the card.

Parameter: int SP_TimerNumber

SP_TimerNumber

Direction: Down

Valid values:

- 1= Timer 1
- 2= Timer 2
- 3= Timer 3

Description: Number of the timer to parametrize.

Parameter: double SP_TimerFrequency

SP_TimerFrequency

Direction: Down

Valid values:

- Minimum:** 0.01
- Maximum:** 20000000.0

Unit: Hz

Description: Frequency of the timer.
0.0 means the timer is off.

Parameter: double SP_TimerRatio

SP_TimerRatio

Direction: Down

Valid values:

Minimum: 0.0

Maximum: 1.0

Description: Ratio of the high and low contingent of one period.

If you are unsure, specify a value of 0.5.

If SP_TimerFrequency is 0.0, a value of 0.0 means the timer is low, otherwise the timer is high.

Parameter: double SA_RealFrequency

SA_RealFrequency

Direction: Up

Unit: Hz

Description: The real frequency applied to the timer. Because of internal limitations not each value can be set.

Parameter: double SA_RealRatio

SA_RealRatio

Direction: Up

Description: The real ratio applied to the timer. Because of internal limitations not each value can be set.

X.5 Set_EncoderInterpolation

Set the interpolation of the encoder.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SP_EncoderInterpolation

SP_EncoderInterpolation

Direction: Down

Valid values:

0= 1 fold (TTL & 1Vss)

1= 2 fold (TTL & 1Vss)

2= 3 fold (only 1Vss)

3= 4 fold (TTL & 1Vss)

4= 5 fold (only 1Vss)

5= 6 fold (only 1Vss)

6= 8 fold (only 1Vss)

7= 10 fold (only 1Vss)

8= 12 fold (only 1Vss)

9= 16 fold (only 1Vss)

10= 20 fold (only 1Vss)

11= 24 fold (only 1Vss)

12= 32 fold (only 1Vss)

13= 40 fold (only 1Vss)

14= 48 fold (only 1Vss)

15= 64 fold (only 1Vss)

Description: Interpolation mode of the encoder.

Attention! If a encoder with 5V TTL signal is used, only mode 0, 1 and 3 does work. If another mode is selected, the result is unpredictable.

X.6 Get_EncoderInterpolation

Get the interpolation of the encoder.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to retrieve information.

Parameter: int SA_EncoderInterpolation

SA_EncoderInterpolation

Direction: Up

Valid values:

0= 1 fold (TTL & 1Vss)

1= 2 fold (TTL & 1Vss)

2= 3 fold (only 1Vss)

3= 4 fold (TTL & 1Vss)

4= 5 fold (only 1Vss)

5= 6 fold (only 1Vss)

6= 8 fold (only 1Vss)

7= 10 fold (only 1Vss)

8= 12 fold (only 1Vss)

9= 16 fold (only 1Vss)

10= 20 fold (only 1Vss)

11= 24 fold (only 1Vss)

12= 32 fold (only 1Vss)

13= 40 fold (only 1Vss)

14= 48 fold (only 1Vss)

15= 64 fold (only 1Vss)

Description: Interpolation mode of the encoder.

X.7 Set_EncoderDirection

Set the direction of the encoder.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SP_EncoderDirection

SP_EncoderDirection

Direction: Down

Valid values:

0= normal

1= reverse

Description: Count direction of the encoder.

X.8 Get_EncoderDirection

Get the direction of the encoder.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SA_EncoderDirection

SA_EncoderDirection

Direction: Up

Valid values:

0= normal

1= reverse

Description: Count direction of the encoder.

X.9 Set_EncoderMode

Set the behaviour of the encoder when a reference is reached.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SP_EncoderMode

SP_EncoderMode

Direction: Down

Valid values:

0= no function

1= load counter with next reference mark

2= load counter with all reference marks

3= clear counter with all reference marks, load when -1 is reached

4= counter without phase discriminator (pulse counter)

Description: Behaviour when a reference is reached.

X.10 Get_EncoderMode

Get the behaviour of the encoder when a reference is reached.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SA_EncoderMode

SA_EncoderMode

Direction: Up

Valid values:

- 0= no function
- 1= load counter with next reference mark
- 2= load counter with all reference marks
- 3= clear counter with all reference marks, load when -1 is reached
- 4= counter without phase discriminator (pulse counter)

Description: Behaviour when a reference is reached.

X.11 Set_EncoderLatchSource

Set the latch source which triggers acquiring one value.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

- 6= Encoder 1
- 7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SP_EncoderLatchSource

SP_EncoderLatchSource

Direction: Down

Valid values:

- 0= never (locked)
- 1= Timer 1
- 2= Timer 2
- 3= Timer 3
- 4= Sensor channel 1
- 5= Sensor channel 2
- 6= Sensor channel 3
- 7= Sensor channel 4
- 8= Sensor channel 5
- 9= Sensor channel 6
- 10= Digital In 1
- 11= Digital In 2
- 12= Digital In 3
- 13= Digital In 4
- 14= second reference mark
- 15= all reference marks

Description: Latch source which triggers acquiring one value.

X.12 Get_EncoderLatchSource

Get the latch source which triggers acquiring one value.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

- 6= Encoder 1
- 7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SA_EncoderLatchSource

SA_EncoderLatchSource

Direction: Up

Valid values:

- 0= never (locked)
- 1= Timer 1
- 2= Timer 2
- 3= Timer 3
- 4= Sensor channel 1
- 5= Sensor channel 2
- 6= Sensor channel 3
- 7= Sensor channel 4
- 8= Sensor channel 5
- 9= Sensor channel 6
- 10= Digital In 1
- 11= Digital In 2
- 12= Digital In 3
- 13= Digital In 4
- 14= second reference mark
- 15= all reference marks

Description: Latch source which triggers acquiring one value.

X.13 Set_EncoderPreload

Set the preload value which is used when loading the encoder.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

- 6= Encoder 1
- 7= Encoder 2

Description: Number of the encoder to parametrize.

Parameter: int SP_EncoderPreloadValue

SP_EncoderPreloadValue

Direction: Down

Valid values:

Minimum: INT_MIN (-2147483648)

Maximum: INT_MAX (2147483647)

Description: Preload value which is used when loading the encoder.

X.14 Set_DigitalOutSource

Set the source which is used to output one value.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SP_DigitalOut1..4

SP_DigitalOut1..4

Direction: Down

Valid values:

- 1= leave unchanged
- 0= user (output function)

- 1 = Timer 1 pulse width
- 2 = Timer 2 pulse width
- 3 = Timer 3 pulse width

Description: Source which is used to output one value.

-1 means the value is not set. 0 means the value is set by function Set_DigitalOutValue, Timer 1..3 means the value is equal to the timer value.

X.15 Get_DigitalOutSource

Get the source which is used to output one value.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SA_DigitalOut1..4

SA_DigitalOut1..4

Direction: Up

Valid values:

- 1 = leave unchanged
- 0 = user (output function)
- 1 = Timer 1 pulse width
- 2 = Timer 2 pulse width
- 3 = Timer 3 pulse width

Description: Source which is used to output one value.

X.16 Set_TxDSource

Set the source which is used to output one value.

Parameter: int SP_TxDChannel1..6

SP_TxDChannel1..6

Direction: Down

Valid values:

- 1 = leave unchanged
- 0 = sensor output (transmitter)
- 1 = user (output function)

Description: Source which is used to output one value.

-1 means the channel will not be changed. 0 means the value is set by sensor output (if a sensor command is send to the sensor), 1 means the value is set by function Set_TxDValue.

X.17 Get_TxDSource

Get the source which is used to output one value.

Parameter: int SA_TxDChannel1..6

SA_TxDChannel1..6

Direction: Up

Valid values:

- 1 = leave unchanged
- 0 = sensor output (transmitter)
- 1 = user (output function)

Description: Source which is used to output one value.

X.18 Set_TrgSource

Set the source which is used to output one value.

Parameter: int SP_TrgChannel1..6

SP_TrgChannel1..6

Direction: Down

Valid values:

- 1 = leave unchanged
- 0 = user (output function)
- 1 = Timer 1 pulse width
- 2 = Timer 2 pulse width
- 3 = Timer 3 pulse width

Description: Source which is used to output one value.

-1 means the value is not set. 0 means the value is set by function Set_TrgValue, Timer 1..3 means the value is equal to the timer value.

X.19 Get_TrgSource

Get the source which is used to output one value.

Parameter: int SA_TrgChannel1..6

SA_TrgChannel1..6

Direction: Up

Valid values:

- 1 = leave unchanged
- 0 = user (output function)
- 1 = Timer 1 pulse width
- 2 = Timer 2 pulse width
- 3 = Timer 3 pulse width

Description: Source which is used to output one value.

X.20 Set_DigitalInLatchSource

Set the latch source which triggers acquiring one value.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SP_DigitalInLatchSource

SP_DigitalInLatchSource

Direction: Down

Valid values:

- 0 = never (locked)
- 1 = Timer 1 pulse width
- 2 = Timer 2 pulse width
- 3 = Timer 3 pulse width
- 4 = Sensor channel 1
- 5 = Sensor channel 2
- 6 = Sensor channel 3
- 7 = Sensor channel 4

Description: Latch source which triggers acquiring one value.

X.21 Get_DigitalInLatchSource

Get the latch source which triggers acquiring one value.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SA_DigitalInLatchSource

SA_DigitalInLatchSource

Direction: Up

Valid values:

- 0= never (locked)
- 1= Timer 1 pulse width
- 2= Timer 2 pulse width
- 3= Timer 3 pulse width
- 4= Sensor channel 1
- 5= Sensor channel 2
- 6= Sensor channel 3
- 7= Sensor channel 4

Description: Latch source which triggers acquiring one value.

X.22 Set_ADCLatchSource

Set the latch source which triggers acquiring one value.

Attention! This command is only allowed if IF2008E extension card is installed

Parameter: int SP_ADCNumber

SP_ADCNumber

Direction: Down

Valid values:

- 10= ADC 1
- 11= ADC 2

Description: Number of the ADC (Analog/Digital converter) to parametrize.

Parameter: int SP_ADCLatchSource

SP_ADCLatchSource

Direction: Down

Valid values:

- 0= never (locked)
- 1= Timer 1
- 2= Timer 2
- 3= Timer 3
- 4= Sensor channel 1
- 5= Sensor channel 2
- 6= Sensor channel 3
- 7= Sensor channel 4
- 8= Sensor channel 5
- 9= Sensor channel 6
- 10= Digital In 1
- 11= Digital In 2
- 12= Digital In 3
- 13= Digital In 4

Description: Latch source which triggers acquiring one value.

X.23 Get_ADCLatchSource

Get the latch source which triggers acquiring one value.

Attention! This command is only allowed if IF2008E extension card is installed

Parameter: int SP_ADCNumber

SP_ADCNumber

Direction: Down

Valid values:

6= ADC 1

7= ADC 2

Description: Number of the ADC (Analog/Digital converter) to parametrize.

Parameter: int SA_ADCLatchSource

SA_ADCLatchSource

Direction: Up

Valid values:

0= never (locked)

1= Timer 1

2= Timer 2

3= Timer 3

4= Sensor channel 1

5= Sensor channel 2

6= Sensor channel 3

7= Sensor channel 4

8= Sensor channel 5

9= Sensor channel 6

10= Digital In 1

11= Digital In 2

12= Digital In 3

13= Digital In 4

Description: Latch source which triggers acquiring one value.

X.24 Get_FPGAVersion

Get the version of the FPGA on the card.

Parameter: int SA_FPGAVersion

SA_FPGAVersion

Direction: Up

Valid values:

Minimum: 0

Maximum: 255

Description: Version of the FPGA on the card.

X.25 Is_Channel56Available

Check if additional sensor channels at IF2008E extension card are available.

Parameter: int SA_Channel56Available

SA_Channel56Available

Direction: Up

Valid values:

Minimum: 0= no

Maximum: 1= yes

Description: Availability of sensor channels at IF2008E extension card.

X.26 Is_ADCAvailable

Check if analog digital converter (ADC) at IF2008E extension card are available.

Parameter: int SA_ADCAvailable

SA_ADCAvailable

Direction: Up

Valid values:

Minimum: 0= no

Maximum: 1= yes

Description: Availability of analog digital converter (ADC) at IF2008E extension card.

X.27 Is_DigitalIOAvailable

Check if digital IO at IF2008E extension card or IF2008IO extension slot is available.

Parameter: int SA_DigitalIOAvailable

SA_DigitalIOAvailable

Direction: Up

Valid values:

Minimum: 0= no

Maximum: 1= yes

Description: Availability of digital IO at IF2008E extension card or IF2008IO extension slot.

X.28 Clear_Encoder

Clears the encoder value (set it to zero).

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to clear.

X.29 Load_Encoder

Loads the encoder value with the value set by function Set_EncoderPreload.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to load.

X.30 Latch_Encoder

Latch the encoder (get latest value into encoder register).
The value can be retrieved using function Get_EncoderVal.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to latch.

X.31 EnableRef_Encoder

Enable loading encoder value at next reference (set by function Set_EncoderMode with parameter next reference).

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder for enabling reference.

X.32 Get_EncoderValue

Get the value of encoder register.
This function is usefull in combination with function Get_EncoderVal.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to latch.

Parameter: int SA_EncoderRawValue

SA_EncoderRawValue

Direction: Up

Valid values:

Minimum: INT_MIN (-2147483648)

Maximum: INT_MAX (2147483647)

Description: Raw value of encoder register.

Parameter: double SA_EncoderScaledValue

SA_EncoderScaledValue

Direction: Up

Description: Scaled value of encoder register.

If IP_Encoder1CountFactor or IP_Encoder2CountFactor (at command Use_Defaults) is not set, the scaled value is equal to the raw value.

X.33 Get_EncoderReference

Get status if reference was reached first or second times.

Parameter: int SP_EncoderNumber

SP_EncoderNumber

Direction: Down

Valid values:

6= Encoder 1

7= Encoder 2

Description: Number of the encoder to latch.

Parameter: int SA_FirstReference

SA_FirstReference

Direction: Up

Valid values:

Minimum: 0= no

Maximum: 1= yes

Description: Reference was reached first time.

Parameter: int SA_SecondReference

SA_SecondReference

Direction: Up

Description: Reference was reached second time.

X.34 Get_ADCValue

Get the value of the ADC.

Attention! This command is only allowed if IF2008E extension card is installed

Parameter: int SP_ADCNumber

SP_ADCNumber

Direction: Down

Valid values:

10= ADC 1

11= ADC 2

Description: Number of the ADC to return value.

Parameter: int SA_ADCRawValue

SA_ADCRawValue

Direction: Up

Valid values:

Minimum: 0

Maximum: 65535

Description: Raw value of ADC.

Parameter: double SA_ADCScaledValue

SA_ADCScaledValue

Direction: Up

Valid values:

0-5

0-10

+ -5

+ -10

Unit: V

Description: Scaled value of ADC.

The range depends on IP_ADC1Range or IP_ADC2Range (at command Use_Defaults).

X.35 Get_DigitalInValue

Get the digital In values.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SA_DigitalInValue1..4

SA_DigitalInValue1..4

Direction: Up

Valid values:

0

1

Description: Digital In value.

X.36 Get_RxDValue

Get the RxD values.

Parameter: int SA_RxDValue1..6

SA_RxDValue1..6

Direction: Up

Valid values:

0

1

Description: Value of the RxD line.

X.37 Set_DigitalOutValue

Set the digital Out values.

This function does only work when Set_DigitalOutSource is set to 0 (output function)

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SP_DigitalOutValue1..4

SP_DigitalOutValue1..4

Direction: Down

Valid values:

-1 = leave unchanged

0

1

Description: Digital Out value. -1 means leave the value unchanged.

X.38 Get_DigitalOutValue

Get the digital Out values.

Attention! This command is only allowed if IF2008E extension card or IF2008IO extension slot is installed

Parameter: int SA_DigitalOutValue1..4

SA_DigitalOutValue1..4

Direction: Up

Valid values:

0

1

Description: Digital Out value.

X.39 Set_TxDValue

Set the values of TxD lines.

This function does only work when Set_TxDSource is set to 1 (output function)

Parameter: int SP_TxDValue1..6

SP_TxDValue1..6

Direction: Down

Valid values:

-1 = leave unchanged

0

1

Description: TxD value.

-1 means leave the value unchanged.

X.40 Get_TxDValue

Get the values of TxD lines.

Parameter: int SA_TxDValue1..6

SA_TxDValue1..6

Direction: Up

Valid values:

0

1

Description: TxD value.

X.41 Set_TrgValue

Set the values of Trg lines.

This function does only work when Set_TrgSource is set to 0 (output function)

Parameter: int SP_TrgValue1..6

SP_TrgValue1..6

Direction: Down

Valid values:

-1 = leave unchanged

0

1

Description: TxD value.

-1 means leave the value unchanged.

X.42 Get_TrgValue

Get the values of Trg lines.

Parameter: int SA_TrgValue1..6

SA_TrgValue1..6

Direction: Up

Valid values:

0

1

Description: Trg value.

Index

- avail, [39](#)
- ClearAllParameters, [36](#)
- CloseSensor, [37](#)
- CP_SensorAnswerTimeout, [54](#), [62](#)
- CreateSensorInstance, [25](#)
- DataAvail, [39](#)
- errText, [41](#)
- GetDLLVersion, [42](#)
- GetDLLVersionU, [42](#)
- GetError, [41](#)
- GetErrorU, [41](#)
- GetParameterDouble, [33](#)
- GetParameterDoubleU, [34](#)
- GetParameterDWORD, [32](#)
- GetParameterDWORDU, [33](#)
- GetParameterInt, [31](#)
- GetParameterIntU, [32](#)
- GetParameterString, [34](#)
- GetParameterStringU, [35](#)
- HysteresisQ1, [65](#)
- HysteresisQ2, [66](#)
- IA_ADC1Range, [346](#)
- IA_ADC2Range, [346](#)
- IA_ASCII, [62](#)
- IA_Averaging, [186](#), [218](#), [233](#), [261](#)
- IA_AvrNbr, [308](#), [316](#), [326](#), [339](#)
- IA_AvrType, [308](#), [316](#), [326](#), [339](#)
- IA_Binning, [262](#)
- IA_CheckRingBufferOverflow, [60](#)
- IA_ChExist1..16, [308](#)
- IA_ChExist1..8, [339](#)
- IA_ChTransmit1..16, [308](#)
- IA_ChTransmit1..4, [326](#)
- IA_ChTransmit1..8, [339](#)
- IA_DataAvailEvent, [63](#)
- IA_DataOn, [61](#)
- IA_Datarate, [61](#)
- IA_DistanceTracking, [114](#)
- IA_Encoder1CountFactor, [346](#)
- IA_Encoder2CountFactor, [346](#)
- IA_ErrorOutput, [160](#)
- IA_MeasureMode, [185](#), [218](#), [233](#), [261](#)
- IA_NumberOfSegments, [278](#), [298](#)
- IA_Option, [129](#)
- IA_OutputContent, [114](#)
- IA_OutputFormat, [114](#)
- IA_OutputMode, [126](#), [146](#)
- IA_OutputTime, [126](#), [146](#)
- IA_OutputType, [126](#), [146](#), [160](#)
- IA_Range, [61](#)
- IA_Range1..16, [308](#)
- IA_Range1..4, [325](#)
- IA_Range1..8, [339](#)
- IA_RefractIndex, [185](#), [218](#), [233](#), [261](#)
- IA_Samplerate, [61](#)
- IA_SettingsChanged, [60](#)
- IA_SpectralAv, [186](#), [218](#), [262](#)
- IA_Speed, [146](#), [160](#)
- IA_SpeedMeasure, [114](#)
- IA_SpeedTracking, [115](#)
- IA_Sync_TrgMode, [160](#)
- IA_TerminationChar, [115](#)
- IA_X1..16, [219](#), [262](#)
- IA_X1..8, [186](#), [233](#)
- instanceHandle, [26--41](#)
- IP_ADC1Range, [345](#)
- IP_ADC2Range, [346](#)
- IP_ASCII, [60](#)
- IP_Averaging, [184](#), [217](#), [232](#), [260](#)
- IP_AvrNbr, [307](#), [316](#), [325](#), [338](#)
- IP_AvrType, [307](#), [316](#), [325](#), [338](#)
- IP_Baudrate, [49--51](#)
- IP_Binning, [261](#)
- IP_ByteSize, [49](#)
- IP_CardInstance, [49](#), [50](#)
- IP_ChannelNumber, [50](#), [51](#)
- IP_CheckRingBufferOverflow, [59](#)
- IP_ChExist1..16, [307](#)
- IP_ChExist1..8, [338](#)
- IP_ChTransmit1..16, [307](#)
- IP_ChTransmit1..4, [325](#)
- IP_ChTransmit1..8, [338](#)
- IP_ClearReceiveBuffer, [58](#)
- IP_ClearRingBuffer, [57](#)
- IP_ClearSendBuffer, [57](#)
- IP_DataOn, [59](#)
- IP_Datarate, [59](#)
- IP_DeviceInstance, [52](#), [53](#)
- IP_DistanceTracking, [113](#)
- IP_EncCountMode, [340](#)
- IP_EncInvertTraceA_B, [340](#)
- IP_EncLatchSrc, [341](#)
- IP_EncLoadOnRef, [341](#)
- IP_Encoder1CountFactor, [345](#)

- IP_Encoder2CountFactor, 345
- IP_EncSwapTraceA_B, 340
- IP_ErrorOutput, 159
- IP_EventOnAvailableValues, 63
- IP_EventOnBufferFillsize, 63
- IP_Interface, 44
- IP_LogAppend, 47
- IP_LogFile, 47
- IP_LogFlush, 48
- IP_LogLevel, 47
- IP_LogType, 47
- IP_MaxPacketSize, 48
- IP_MeasureMode, 184, 217, 232, 260
- IP_NumberOfSegments, 278, 298
- IP_Option, 129
- IP_OutputContent, 113
- IP_OutputFormat, 113
- IP_OutputMode, 126, 145
- IP_OutputTime, 126, 145
- IP_OutputType, 125, 145, 159
- IP_PacketDelay, 48
- IP_Parity, 49, 52
- IP_Port, 48
- IP_Range, 59, 340
- IP_Range1..16, 307
- IP_Range1..4, 325
- IP_Range1..8, 338
- IP_RefractIndex, 184, 217, 232, 260
- IP_RemoteAddr, 52
- IP_RemotePort, 52
- IP_RingBufferSize, 46
- IP_Samplerate, 59
- IP_ScaleErrorValues, 44
- IP_SpectralAv, 185, 217, 260
- IP_Speed, 145, 159
- IP_SpeedMeasure, 113
- IP_SpeedTracking, 113
- IP_Stopbits, 49
- IP_Sync_TrgMode, 159
- IP_TerminationChar, 114
- IP_TimerResolution, 215, 258
- IP_UsbReadBufCnt, 53
- IP_UsbReadBufSize, 53, 54, 214, 216, 257, 259
- IP_UseGate, 50
- IP_X1..16, 217, 260
- IP_X1..8, 185, 232
- len, 30
- maxLen, 35, 41--43
- maxValues, 39, 40
- OpenSensor, 36
- paramName, 26--35
- paramValue, 26--35
- Poll, 38
- rawData, 38, 40
- read, 40
- ReleaseSensorInstance, 25
- S_Command, 54
- SA_0_0V, 179, 182, 211, 254
- SA_0_10V, 179, 182, 211, 254
- SA_0_SOD, 179, 182, 211, 254
- SA_1_0V, 179, 182, 212, 255
- SA_1_10V, 180, 182, 212, 255
- SA_1_SOD, 179, 182, 211, 254
- SA_ActiveEdge, 206, 249
- SA_ActiveSerialIf, 266, 283
- SA_ADCAvailable, 357
- SA_ADCLatchSource, 356
- SA_ADCRawValue, 359
- SA_ADCScaledValue, 359
- SA_AlarmHysteresis, 79
- SA_AlarmStart, 78, 85, 88
- SA_AlarmWidth, 79, 85, 88
- SA_AnalogGain, 268, 285
- SA_AnalogOffset, 268, 285
- SA_AnalogValue, 66
- SA_ArticleNumber, 116, 127, 134, 150, 279
- SA_ASCII, 117, 120, 136, 138, 149, 151, 178, 208, 227, 251
- SA_AssignLimits_ErrorOutput, 153
- SA_AutoAdaptLED, 199
- SA_AutoAdaptLEDThr, 200
- SA_AutoDark, 189, 236
- SA_AutostartCommand, 84, 87, 90, 102, 107, 112
- SA_AutostartEdge, 83, 86, 89
- SA_AutostartTrigger, 82, 86, 89
- SA_Average, 81, 84, 87, 99, 104, 109
- SA_Average_for_reading, 269, 287
- SA_Averaging, 177, 178, 204, 205, 208, 226, 227, 247, 248, 251
- SA_AvgRefractIndex, 201, 202, 244
- SA_AvIndex, 148, 150, 162, 167
- SA_AvrNbr, 301, 304, 306, 312, 314, 315, 320, 323, 324, 329, 335, 337
- SA_AvrType, 300, 304, 306, 311, 314, 315, 319, 322, 324, 329, 334, 336
- SA_AvType, 117, 119, 135, 137, 148, 151, 168
- SA_BarycenterOffset, 197, 243

- SA_BarycenterScale, 197, 242
- SA_BatteryCharge, 131
- SA_Binning, 247
- SA_BootLoaderVer, 116, 135, 150
- SA_CalibTable, 214, 231, 257
- SA_CCD, 183, 213, 230, 256
- SA_Channel56Available, 356
- SA_ChExist1..16, 301, 304, 306
- SA_ChExist1..4, 320, 323, 324
- SA_ChExist1..8, 329, 335, 337
- SA_ChTransmit1..16, 302, 304, 306
- SA_ChTransmit1..4, 320, 323, 324
- SA_ChTransmit1..8, 330, 335, 337
- SA_Coeffizient, 311
- SA_CompleteAnswer, 54, 62
- SA_ContinuousMode, 65
- SA_Contrast, 266, 283
- SA_DarkSig, 213, 230, 256
- SA_DataScale, 196, 242
- SA_Date, 117, 128, 135, 150
- SA_DatOut, 153, 168
- SA_DEB, 183
- SA_DigitalInLatchSource, 355
- SA_DigitalInValue1..4, 360
- SA_DigitalIOAvailable, 357
- SA_DigitalOut1..4, 353
- SA_DigitalOutValue1..4, 360
- SA_Dignosis, 108
- SA_Direction, 131
- SA_DisplayGain, 269, 285
- SA_DisplayOffset, 268, 285
- SA_DispmEasUnit, 265, 282
- SA_DoubleFrequency, 191
- SA_DutyCycle, 174
- SA_EdgeDetectThreshold, 283
- SA_EnableFlash, 150, 153
- SA_EncoderDirection, 350
- SA_EncoderInterpolation, 349
- SA_EncoderLatchSource, 352
- SA_EncoderMode, 351
- SA_EncoderRawValue, 358
- SA_EncoderScaledValue, 358
- SA_Energy, 67
- SA_ErrorHandler, 117, 119, 128, 135, 137, 148, 151, 167, 265, 282
- SA_ErrorMode, 78, 85, 88, 95, 105, 109
- SA_ErrorNumber, 54
- SA_ErrorOutput, 148, 151
- SA_ErrorStatus, 67, 68
- SA_ErrorText, 55
- SA_Exposure, 190, 223, 237, 238
- SA_Ext_LaserSwitch, 266, 282
- SA_ExtInputMode, 118, 120, 136, 139
- SA_FactorCh1..8, 332
- SA_FirstPeak, 196, 241
- SA_FirstReference, 359
- SA_FPGAVersion, 344, 356
- SA_FreeSR, 190, 222, 223, 237
- SA_FrontEdge_Seg1, 270, 287
- SA_FrontEdge_Seg2, 270, 287
- SA_FrontEdge_Seg3, 287
- SA_FrontEdge_Seg4, 288
- SA_Gate, 343
- SA_HighFrequency, 192
- SA_InvalidValues, 77, 86, 89
- SA_IPAddress, 229
- SA_IsFirstRef, 342
- SA_IsSecondRef, 342
- SA_Keylock, 118, 120, 136, 139, 149, 153, 168, 176, 225
- SA_LampTest, 198
- SA_LampTestThr, 198
- SA_Language, 265, 282
- SA_LaserIntensity, 266, 282
- SA_LaserState, 153, 169
- SA_LastValid, 194, 240
- SA_LatchReg, 342
- SA_LED_Off, 200
- SA_LEDIntensity, 199
- SA_LimitQ1-1, 65
- SA_LimitQ1-2, 65
- SA_LimitQ2-1, 66
- SA_LimitQ2-2, 66
- SA_LimitQA-1, 66
- SA_LimitQA-2, 66
- SA_LinMode, 312, 314, 315
- SA_LinMode1..16, 302, 305, 306
- SA_LinMode1..8, 333, 335, 337
- SA_LinPoint, 303, 313, 322, 334
- SA_Lower_hysteresis, 152
- SA_Lower_limit, 152
- SA_LowerLimit, 269, 286
- SA_LowerWarning, 269, 286
- SA_LowFrequency, 192
- SA_Major, 130
- SA_Master_MidPoint_Setup, 152
- SA_Master_value, 152
- SA_MasterValue, 270, 289
- SA_MaxRaw, 276, 277, 297
- SA_MaxRefractIndex, 201, 202, 244
- SA_MaxScaled, 277, 297, 298
- SA_MeasFrequency, 103, 104, 108
- SA_MeasMode, 286
- SA_MeasObject, 270, 287
- SA_MeasProgName, 268, 285
- SA_MeasProgNumber, 265, 282

- SA_MeasureMode, 175, 178, 193, 208, 224, 226, 238, 251
- SA_MeasureTime, 81, 84, 88
- SA_Median, 286
- SA_Median_OnOff, 128
- SA_MedianIndex, 117, 119, 135, 138
- SA_Minor, 130
- SA_MinRaw, 276, 277, 297
- SA_MinRefractIndex, 201, 202, 243, 244
- SA_MinScaled, 277, 297, 298
- SA_MinSR, 191, 238
- SA_MinSRIndex, 173, 188, 221, 235
- SA_ModeQ1, 65
- SA_ModeQ2, 66
- SA_MovingCount, 117, 119, 135, 138
- SA_NormQ1, 65
- SA_NormQ2, 66
- SA_NormQA, 67
- SA_NormSig, 214, 231, 257
- SA_NumberOfSegments, 287
- SA_Offset, 67, 76, 87, 90, 94, 95, 105, 109, 332
- SA_OffsetValue, 168
- SA_Option, 116, 127, 134, 149, 161, 166, 203, 246, 263, 279
- SA_OutOfRange, 131
- SA_Output_Analog, 162, 167
- SA_Output_Digital, 162, 167
- SA_OutputContent, 92, 107, 111
- SA_OutputFormat, 67, 75, 84, 87, 92, 106, 111
- SA_OutputMode, 117, 120, 136, 138
- SA_OutputTime, 120, 139
- SA_OutputType, 117, 119, 128, 131, 135, 137, 147, 150
- SA_Password, 67
- SA_PeakSearching, 118, 121, 136, 139
- SA_PilotLaser, 64, 103, 107, 112
- SA_PrecedingValues, 77, 85, 89
- SA_PreTreated, 213, 230, 256
- SA_Q1Hysteresis, 98, 105, 110
- SA_Q1Negation, 98, 105, 110
- SA_Q1Start, 97, 105, 110
- SA_Q1Value, 65
- SA_Q1Width, 98, 105, 110
- SA_Q2Hysteresis, 99
- SA_Q2Negation, 99
- SA_Q2Start, 99
- SA_Q2Value, 65
- SA_Q2Width, 99
- SA_Range, 116, 127, 132, 134, 149, 153, 161, 166, 168, 172, 180, 187, 209, 220, 227, 234, 252, 263, 279
- SA_RangeBegin, 80, 85, 88, 96, 106, 111
- SA_RangeEnd, 80, 85, 88, 97, 106, 111
- SA_Ranges, 188, 235
- SA_RealFrequency, 348
- SA_RealRatio, 348
- SA_RearEdge_Seg1, 270, 288
- SA_RearEdge_Seg2, 270, 288
- SA_Ref, 343
- SA_RefractIndex, 175, 193, 224, 239
- SA_RefractIndexFileIdx, 201, 243, 244
- SA_RefractIndexFileName, 201, 202, 243, 244
- SA_Reserve_1, 263, 279
- SA_Reserve_10, 288
- SA_Reserve_2, 266, 283
- SA_Reserve_3, 266, 286
- SA_Reserve_4, 270, 287
- SA_Reserve_6, 288
- SA_Reserve_7, 288
- SA_Reserve_8, 288
- SA_Reserve_9, 288
- SA_Reserved1, 121, 139
- SA_Reserved2, 121, 139
- SA_Reverse, 204, 246
- SA_Revision, 130
- SA_RS232_Baudrate, 266, 283
- SA_RS232_Parity, 267, 283
- SA_RS232_StopBits, 267, 283
- SA_RS232_TimeoutRecv, 267, 284
- SA_RS232_TimeoutSend, 267, 284
- SA_RS422_Baudrate, 267, 284
- SA_RS422_Parity, 267, 284
- SA_RS422_StopBits, 267, 284
- SA_RS422_TimeoutRecv, 268, 284
- SA_RS422_TimeoutSend, 268, 284
- SA_RxDValue1..6, 360
- SA_Samplerate, 148, 161, 166
- SA_SaveSettingsMode, 118, 120, 136, 139
- SA_ScaleFactor, 75, 85, 88, 94, 104, 109
- SA_SecondReference, 359
- SA_Sensor, 116, 134, 147, 161, 166, 172, 178, 187, 208, 220, 227, 234, 251
- SA_SensorBaudrate, 64, 83, 86, 90, 101, 106, 111, 120, 138, 151, 213, 229, 256
- SA_SensorDatabits, 64
- SA_SensorStopbits, 64
- SA_SensorType, 116, 134, 147, 161, 166

- SA_SerialNumber, 68, 116, 127, 134, 149, 161, 167, 263, 279
- SA_SerialOutFormat, 282
- SA_ShowChannels, 330, 335, 337
- SA_ShowLinearized, 331, 335, 337
- SA_SkippedSpectra, 216, 259
- SA_SoftArtArm, 263, 279
- SA_SoftArtBoot, 263, 279
- SA_SoftArtDSP, 263, 279
- SA_SoftVerArm, 264, 280
- SA_SoftVerBoot, 263, 280
- SA_SoftVerDSP, 264, 280
- SA_Softwareversion, 116, 127, 128, 132, 134, 147, 162, 169
- SA_SpectralAv, 177
- SA_Spectrum, 215, 258
- SA_Speed, 135, 138, 148, 150, 167
- SA_SRIndex, 173, 178, 189, 207, 222, 226, 236, 250, 299, 304, 305, 309, 313, 315, 317, 322, 324, 327, 334, 336
- SA_SSIFormat, 93, 107, 112
- SA_SumFreq, 174
- SA_SWType, 150
- SA_Sync_TrgMode, 149, 151
- SA_Target_Distance, 269
- SA_TeachValue1, 118, 119, 137
- SA_TeachValue2, 118, 119, 137
- SA_Temperature, 68, 87, 108
- SA_TemperatureCoefficient, 319
- SA_TerminationChar, 93, 107, 111
- SA_Threshold, 118, 121, 136, 139, 176, 194, 225, 239
- SA_Threshold1, 195, 240
- SA_Threshold2, 195, 241
- SA_Timestamp, 215, 258
- SA_TransmitIntensity, 192
- SA_TrgChannel1..6, 354
- SA_TrgMode, 300, 304, 306, 310, 314, 315, 318, 323, 324, 328, 335, 337
- SA_TrgValue1..6, 361
- SA_TriggerDelay, 82, 86, 89, 100, 104, 109
- SA_TriggerEdge, 82, 86, 89, 100, 104, 109
- SA_TxDChannel1..6, 353
- SA_TxDValue1..6, 361
- SA_Unit, 131
- SA_UnlinearizedEddyValue, 319
- SA_Upper_hysteresis, 152
- SA_Upper_limit, 152
- SA_UpperLimit, 269, 285
- SA_UpperWarning, 269, 286
- SA_UserMeasProgNumber, 268, 285
- SA_ValidRange, 77, 86, 89
- SA_ValueActual, 132
- SA_ValueMedian, 132
- SA_Version, 64, 68, 84, 104, 130, 180, 209, 227, 252, 305, 314, 323, 336
- SA_VideoSignal, 125, 145, 158
- SA_Watchdog, 207, 250
- SA_WatchdogPeriod, 207, 250
- SA_WhiteRef, 214, 230, 257
- SA_WindowMax, 96, 105, 109
- SA_WindowMin, 96, 104, 109
- SA_X1..16, 208, 210, 251, 253
- SA_X1..8, 179, 180, 227, 228
- SA_ZeroPoint, 168
- SA_ZeroSet, 131
- scaledData, 38, 40
- sensor, 25
- SensorCommand, 37
- SetParameterDouble, 28
- SetParameterDoubleU, 29
- SetParameterDWORD, 27
- SetParameterDWORDU, 28
- SetParameterInt, 26
- SetParameterIntU, 27
- SetParameterString, 29
- SetParameterStringL, 30
- SetParameterStringU, 30
- SP_0V, 181, 211, 254
- SP_10V, 182, 211, 254
- SP_ActivateGate, 347
- SP_ActiveEdge, 206, 249
- SP_ActiveSerialIf, 272, 290
- SP_ADCLatchSource, 355
- SP_ADCNumber, 355, 356, 359
- SP_AlarmHysteresis, 78, 85, 88
- SP_AlarmStart, 78
- SP_AlarmWidth, 79
- SP_AllDevices, 62
- SP_AnalogGain, 274, 292
- SP_AnalogOffset, 274, 292
- SP_ASCII, 123, 142, 157
- SP_AutoAdaptLED, 199
- SP_AutoAdaptLEDThr, 200
- SP_AutoDark, 188, 235
- SP_AutostartCommand, 83, 101
- SP_AutostartEdge, 82
- SP_AutostartTrigger, 82
- SP_Average, 80, 99
- SP_Average_for_reading, 275, 294
- SP_Averaging, 177, 204, 225, 247

- SP_AveragingForDark, 173, 188, 221, 235
- SP_AvIndex, 154, 163, 165, 170
- SP_AvrNbr, 301, 311, 319, 329
- SP_AvrType, 300, 311, 319, 328
- SP_AvType, 122, 140, 155, 164, 170
- SP_BarycenterOffset, 197, 243
- SP_BarycenterScale, 197, 242
- SP_Binning, 246
- SP_BlockSize, 216, 231, 259
- SP_CalibTable, 184, 216, 231, 259
- SP_Chan, 302, 303, 318, 321, 331--333
- SP_ChTransmit1..16, 301
- SP_ChTransmit1..4, 320
- SP_ChTransmit1..8, 330
- SP_CmdStr, 62
- SP_Coeffizient, 310
- SP_CoeffParam, 310, 311
- SP_ContinuousMode, 72
- SP_Contrast, 272, 290
- SP_DataScale, 196, 242
- SP_DigitalInLatchSource, 354
- SP_DigitalOut1..4, 352
- SP_DigitalOutValue1..4, 360
- SP_Direction, 133
- SP_DisplayGain, 274, 292
- SP_DisplayOffset, 274, 292
- SP_DispMeasUnit, 271, 289
- SP_DoubleFrequency, 191
- SP_DutyCycle, 174
- SP_EdgeDetectThreshold, 290
- SP_EnableFlash, 154
- SP_Encoder1, 202, 245
- SP_Encoder2, 203, 245
- SP_Encoder3, 203, 245
- SP_EncoderDirection, 349
- SP_EncoderInterpolation, 348
- SP_EncoderLatchSource, 351
- SP_EncoderMode, 350
- SP_EncoderNumber, 348--352, 357--359
- SP_EncoderPreloadValue, 352
- SP_ErrorHandler, 122, 128, 141, 156, 271, 289
- SP_ErrorMode, 77, 95
- SP_ErrorOutput, 158
- SP_Exposure, 190, 223, 237
- SP_Ext_LaserSwitch, 271, 289
- SP_ExtInputMode, 124, 143
- SP_FactorCh1..8, 331
- SP_FirstPeak, 196, 241
- SP_FreeSR, 189, 222, 237
- SP_FrontEdge_Seg1, 264, 275, 280, 294
- SP_FrontEdge_Seg2, 265, 275, 281, 294
- SP_FrontEdge_Seg3, 281, 295
- SP_FrontEdge_Seg4, 281, 295
- SP_FullRange, 184, 216, 231, 259
- SP_GateChannel, 347
- SP_HighFrequency, 192
- SP_HysteresisQ1, 69
- SP_HysteresisQ2, 69
- SP_InvalidValues, 77
- SP_IPAddress, 229
- SP_Keylock, 121, 140, 154, 169, 176, 225
- SP_LampTest, 198
- SP_LampTestThr, 198
- SP_Language, 271, 289
- SP_LaserIntensity, 271, 290
- SP_LastValid, 194, 240
- SP_LED_Off, 200
- SP_LEDIntensity, 199
- SP_Length, 183
- SP_LimitQ1-1, 70
- SP_LimitQ1-2, 70
- SP_LimitQ2-1, 70
- SP_LimitQ2-2, 71
- SP_LimitQA-1, 70
- SP_LimitQA-2, 71
- SP_LinMode, 302, 312, 332
- SP_LinPos, 303, 312, 313, 321, 322, 333, 334
- SP_LoadReg, 342
- SP_Lower_hysteresis, 155
- SP_Lower_limit, 155
- SP_LowerLimit, 274, 293
- SP_LowerWarning, 275, 293
- SP_LowFrequency, 191
- SP_Master_value, 156
- SP_MasterValue, 276, 296
- SP_MeasFrequency, 103
- SP_MeasMode, 293
- SP_MeasObject, 275, 294
- SP_MeasProgName, 274, 292
- SP_MeasProgNumber, 264, 271, 280, 289
- SP_MeasureMode, 175, 193, 223, 238
- SP_MeasureTime, 81
- SP_MeasValue, 147
- SP_Median, 294
- SP_Median_OnOff, 129
- SP_MedianIndex, 122, 140
- SP_ModeQ1, 71
- SP_ModeQ2, 71
- SP_MovingCount, 122, 140
- SP_NormQ1, 72
- SP_NormQ2, 72

- SP_NormQA, [72](#)
- SP_NumberOfPoints, [205](#), [248](#)
- SP_NumberOfSegments, [294](#)
- SP_Offset, [69](#), [76](#), [94](#), [183](#), [331](#)
- SP_Option, [203](#), [245](#)
- SP_OutNr, [181](#), [210](#), [253](#)
- SP_OutputContent, [92](#)
- SP_OutputFormat, [75](#), [92](#)
- SP_OutputMode, [123](#), [143](#)
- SP_OutputTime, [124](#), [143](#)
- SP_OutputType, [123](#), [128](#), [133](#), [142](#), [157](#)
- SP_PeakSearching, [125](#), [144](#)
- SP_PilotLaser, [69](#), [103](#)
- SP_PrecedingValues, [76](#)
- SP_Q1Hysteresis, [97](#)
- SP_Q1Negation, [97](#)
- SP_Q1Start, [97](#)
- SP_Q1Width, [97](#)
- SP_Q2Hysteresis, [98](#), [106](#), [110](#)
- SP_Q2Negation, [98](#), [106](#), [110](#)
- SP_Q2Start, [98](#), [105](#), [110](#)
- SP_Q2Width, [98](#), [106](#), [110](#)
- SP_RangeBegin, [79](#), [96](#)
- SP_RangeEnd, [80](#), [96](#)
- SP_ReadMode, [215](#), [258](#)
- SP_RearEdge_Seg1, [265](#), [276](#), [281](#), [295](#)
- SP_RearEdge_Seg2, [265](#), [276](#), [281](#), [295](#)
- SP_RearEdge_Seg3, [281](#), [295](#)
- SP_RearEdge_Seg4, [281](#), [295](#)
- SP_RefractIndex, [163](#), [175](#), [193](#), [224](#), [239](#)
- SP_RefractIndexFile, [202](#), [245](#)
- SP_RefractIndexFileIdx, [201](#), [202](#), [243](#), [245](#)
- SP_Reserve_10, [296](#)
- SP_Reserve_2, [271](#), [290](#)
- SP_Reserve_3, [272](#), [293](#)
- SP_Reserve_4, [275](#), [294](#)
- SP_Reserve_6, [295](#)
- SP_Reserve_7, [295](#)
- SP_Reserve_9, [296](#)
- SP_Reverse, [203](#), [246](#)
- SP_RS232_Baudrate, [272](#), [290](#)
- SP_RS232_Parity, [272](#), [290](#)
- SP_RS232_StopBits, [272](#), [291](#)
- SP_RS232_TimeoutRecv, [273](#), [291](#)
- SP_RS232_TimeoutSend, [272](#), [291](#)
- SP_RS422_Baudrate, [273](#), [291](#)
- SP_RS422_Parity, [273](#), [291](#)
- SP_RS422_StopBits, [273](#), [291](#)
- SP_RS422_TimeoutRecv, [273](#), [292](#)
- SP_RS422_TimeoutSend, [273](#), [291](#)
- SP_SaveSettingsMode, [124](#), [143](#)
- SP_ScaleFactor, [75](#), [94](#)
- SP_Sensor, [172](#), [187](#), [220](#), [234](#)
- SP_SensorBaudrate, [83](#), [101](#), [122](#), [141](#), [156](#), [212](#), [229](#), [255](#)
- SP_SensorReset12, [343](#)
- SP_SensorReset34, [343](#)
- SP_SerialOutFormat, [289](#)
- SP_ShowChannels, [330](#)
- SP_ShowLinearized, [330](#)
- SP_SOD, [181](#), [211](#), [254](#)
- SP_SpectralAv, [177](#)
- SP_Speed, [141](#), [157](#)
- SP_SRIndex, [173](#), [189](#), [221](#), [236](#), [299](#), [309](#), [317](#), [327](#)
- SP_SSIFormat, [93](#)
- SP_Stand-by, [73](#)
- SP_SumFreq, [174](#)
- SP_Sync_TrgMode, [156](#)
- SP_Target_Distance, [275](#)
- SP_TeachValue1, [124](#), [144](#)
- SP_TeachValue2, [124](#), [144](#)
- SP_TempCoeffParam, [318](#), [319](#)
- SP_TemperatureCoefficient, [318](#)
- SP_TerminationChar, [92](#)
- SP_Threshold, [125](#), [144](#), [176](#), [194](#), [224](#), [239](#)
- SP_Threshold1, [195](#), [240](#)
- SP_Threshold2, [195](#), [241](#)
- SP_TimerFrequency, [347](#)
- SP_TimerNumber, [347](#)
- SP_TimerRatio, [348](#)
- SP_TransmitIntensity, [170](#), [192](#)
- SP_TrgChannel1..6, [354](#)
- SP_TrgMode, [300](#), [310](#), [317](#), [328](#)
- SP_TrgValue1..6, [361](#)
- SP_TriggerDelay, [81](#), [100](#)
- SP_TriggerEdge, [81](#), [100](#)
- SP_TriggerMode_Edge, [205](#), [248](#)
- SP_TriggerMode_State, [206](#), [249](#)
- SP_TxDChannel1..6, [353](#)
- SP_TxDValue1..6, [361](#)
- SP_Unit, [133](#)
- SP_Upper_hysteresis, [155](#)
- SP_Upper_limit, [155](#)
- SP_UpperLimit, [274](#), [293](#)
- SP_UpperWarning, [274](#), [293](#)
- SP_UserMeasProgNumber, [273](#), [292](#)
- SP_ValidRange, [76](#)
- SP_VideoMode, [158](#)
- SP_WaitSpectrumTimeout, [215](#), [258](#)
- SP_Watchdog, [206](#), [249](#)
- SP_WatchdogPeriod, [207](#), [250](#)
- SP_Weighting, [173](#), [188](#), [221](#), [235](#)
- SP_WindowMax, [96](#)

SP_WindowMin, [95](#)
SP_X1..16, [209](#), [252](#)
SP_X1..8, [180](#), [228](#)
SP_Zero, [212](#), [255](#)

TransferData, [40](#)

versionStr, [42](#)