

LibTiePie

0.3.1

Generated by Doxygen 1.8.1.1

Thu Jul 5 2012 11:04:56

Contents

1	Main Page	1
1.1	Supported instruments	1
1.2	Error handling	1
2	Module Index	3
2.1	Modules	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Constants	7
4.1.1	Detailed Description	8
4.1.2	Macro Definition Documentation	8
4.1.2.1	CKN_COUPLING_COUNT	8
4.1.2.2	FMN_GENERATORMODE_COUNT	8
4.1.2.3	MMN_MEASUREMODE_COUNT	8
4.1.2.4	STN_SIGNALTYPE_COUNT	8
4.1.2.5	TKN_KIND_COUNT	8
4.1.2.6	TSN_CHANNEL_COUNT	8
4.2	Clock modes	9
4.2.1	Detailed Description	9
4.2.2	Macro Definition Documentation	9
4.2.2.1	CM_EXTERNAL	9
4.2.2.2	CM_INTERNAL	9
4.3	Coupling type bit numbers	10
4.3.1	Detailed Description	10
4.3.2	Macro Definition Documentation	10
4.3.2.1	CKB_ACA	10
4.3.2.2	CKB_ACV	10
4.3.2.3	CKB_DCA	10
4.3.2.4	CKB_DCV	10

4.3.2.5	CKB_OHM	10
4.4	Coupling types	11
4.4.1	Detailed Description	11
4.4.2	Macro Definition Documentation	11
4.4.2.1	CK_ACA	11
4.4.2.2	CK_ACV	11
4.4.2.3	CK_DCA	11
4.4.2.4	CK_DCV	11
4.4.2.5	CK_OHM	11
4.4.2.6	CK_UNKNOWN	12
4.5	Coupling type masks	13
4.5.1	Detailed Description	13
4.5.2	Macro Definition Documentation	13
4.5.2.1	CKM_A	13
4.5.2.2	CKM_OHM	13
4.5.2.3	CKM_V	13
4.6	Generator mode bit numbers	14
4.6.1	Detailed Description	14
4.6.2	Macro Definition Documentation	14
4.6.2.1	FMB_SAMPLEFREQUENCY	14
4.6.2.2	FMB_SIGNALFREQUENCY	14
4.7	Generator modes	15
4.7.1	Detailed Description	15
4.7.2	Macro Definition Documentation	15
4.7.2.1	FM_SAMPLEFREQUENCY	15
4.7.2.2	FM_SIGNALFREQUENCY	15
4.7.2.3	FM_UNKNOWN	15
4.8	Measure modes bit numbers	16
4.8.1	Detailed Description	16
4.8.2	Macro Definition Documentation	16
4.8.2.1	MMB_BLOCK	16
4.8.2.2	MMB_STREAM	16
4.9	Measure modes	17
4.9.1	Detailed Description	17
4.9.2	Macro Definition Documentation	17
4.9.2.1	MM_BLOCK	17
4.9.2.2	MM_STREAM	17
4.9.2.3	MM_UNKNOWN	17
4.10	Signal type bit numbers	18
4.10.1	Detailed Description	18

4.10.2	Macro Definition Documentation	18
4.10.2.1	STB_ARBITRARY	18
4.10.2.2	STB_DC	18
4.10.2.3	STB_NOISE	18
4.10.2.4	STB_SINE	18
4.10.2.5	STB_SQUARE	18
4.10.2.6	STB_TRIANGLE	18
4.11	Signal types	19
4.11.1	Detailed Description	19
4.11.2	Macro Definition Documentation	19
4.11.2.1	ST_ARBITRARY	19
4.11.2.2	ST_DC	19
4.11.2.3	ST_NOISE	19
4.11.2.4	ST_SINE	19
4.11.2.5	ST_SQUARE	19
4.11.2.6	ST_TRIANGLE	19
4.11.2.7	ST_UNKNOWN	19
4.12	Signal type masks	20
4.12.1	Detailed Description	20
4.12.2	Macro Definition Documentation	20
4.12.2.1	STM_AMPLITUDE	20
4.12.2.2	STM_FREQUENCY	20
4.12.2.3	STM_OFFSET	20
4.12.2.4	STM_PHASE	20
4.12.2.5	STM_SYMMETRY	20
4.13	Trigger holdoff	21
4.13.1	Detailed Description	21
4.13.2	Macro Definition Documentation	21
4.13.2.1	TH_ALLPRESAMPLES	21
4.14	Trigger kinds bit numbers	22
4.14.1	Detailed Description	22
4.14.2	Macro Definition Documentation	22
4.14.2.1	TKB_DROPINWINDOW	22
4.14.2.2	TKB_DROPOUTWINDOW	22
4.14.2.3	TKB_EDGE	22
4.14.2.4	TKB_FALLING	22
4.14.2.5	TKB_INWINDOW	22
4.14.2.6	TKB_OUTWINDOW	22
4.14.2.7	TKB_RISING	22
4.15	Trigger kinds	23

4.15.1 Detailed Description	23
4.15.2 Macro Definition Documentation	23
4.15.2.1 TK_DROPINWINDOW	23
4.15.2.2 TK_DROPOUTWINDOW	23
4.15.2.3 TK_EDGE	23
4.15.2.4 TK_FALLING	23
4.15.2.5 TK_INWINDOW	23
4.15.2.6 TK_OUTWINDOW	23
4.15.2.7 TK_RISING	23
4.15.2.8 TK_UNKNOWN	23
4.16 Trigger kind masks	24
4.16.1 Detailed Description	24
4.16.2 Macro Definition Documentation	24
4.16.2.1 TKM_ALL	24
4.16.2.2 TKM_EDGE	24
4.16.2.3 TKM_NONE	24
4.16.2.4 TKM_WINDOW	24
4.17 Time outs	25
4.17.1 Detailed Description	25
4.17.2 Macro Definition Documentation	25
4.17.2.1 TO_INFINITY	25
4.18 Trigger sources	26
4.18.1 Detailed Description	27
4.18.2 Macro Definition Documentation	27
4.18.2.1 TS_CH1	27
4.18.2.2 TS_CH10	27
4.18.2.3 TS_CH11	27
4.18.2.4 TS_CH12	28
4.18.2.5 TS_CH13	28
4.18.2.6 TS_CH14	28
4.18.2.7 TS_CH15	28
4.18.2.8 TS_CH16	28
4.18.2.9 TS_CH17	28
4.18.2.10 TS_CH18	28
4.18.2.11 TS_CH19	28
4.18.2.12 TS_CH2	28
4.18.2.13 TS_CH20	29
4.18.2.14 TS_CH21	29
4.18.2.15 TS_CH22	29
4.18.2.16 TS_CH23	29

4.18.2.17 TS_CH24	29
4.18.2.18 TS_CH25	29
4.18.2.19 TS_CH26	29
4.18.2.20 TS_CH27	29
4.18.2.21 TS_CH28	29
4.18.2.22 TS_CH29	30
4.18.2.23 TS_CH3	30
4.18.2.24 TS_CH30	30
4.18.2.25 TS_CH31	30
4.18.2.26 TS_CH32	30
4.18.2.27 TS_CH4	30
4.18.2.28 TS_CH5	30
4.18.2.29 TS_CH6	30
4.18.2.30 TS_CH7	30
4.18.2.31 TS_CH8	31
4.18.2.32 TS_CH9	31
4.18.2.33 TS_EXT	31
4.18.2.34 TS_EXT2	31
4.18.2.35 TS_EXTANALOG	31
4.18.2.36 TS_GENNEW	31
4.18.2.37 TS_GENSTART	31
4.18.2.38 TS_GENSTOP	31
4.18.2.39 TS_NONE	31
4.19 Trigger source masks	32
4.19.1 Detailed Description	32
4.19.2 Macro Definition Documentation	32
4.19.2.1 TSM_ALL	32
4.19.2.2 TSM_CHANNELS	32
4.19.2.3 TSM_GENALL	32
4.19.2.4 TSM_NONCHANNELS	32
4.19.2.5 TSM_NONE	32
4.20 TiePie device handles	33
4.20.1 Detailed Description	33
4.20.2 Macro Definition Documentation	33
4.20.2.1 TPDEVICEHANDLE_INVALID	33
4.21 Device type	34
4.21.1 Detailed Description	34
4.21.2 Macro Definition Documentation	34
4.21.2.1 DEVICETYPE_GENERATOR	34
4.21.2.2 DEVICETYPE_I2CHOST	34

4.21.2.3	DEVICETYPE_OSCILLOSCOPE	34
4.22	Status return codes	35
4.22.1	Detailed Description	35
4.22.2	Macro Definition Documentation	35
4.22.2.1	LIBTIEPIESTATUS_DEVICE_GONE	35
4.22.2.2	LIBTIEPIESTATUS_INTERNAL_ADDRESS	35
4.22.2.3	LIBTIEPIESTATUS_INVALID_CHANNEL	35
4.22.2.4	LIBTIEPIESTATUS_INVALID_DEVICE_ID	35
4.22.2.5	LIBTIEPIESTATUS_INVALID_DEVICE_INDEX	35
4.22.2.6	LIBTIEPIESTATUS_INVALID_DEVICE_SERIALNUMBER	36
4.22.2.7	LIBTIEPIESTATUS_INVALID_DEVICE_TYPE	36
4.22.2.8	LIBTIEPIESTATUS_INVALID_HANDLE	36
4.22.2.9	LIBTIEPIESTATUS_INVALID_TRIGGER_SOURCE	36
4.22.2.10	LIBTIEPIESTATUS_INVALID_VALUE	36
4.22.2.11	LIBTIEPIESTATUS_NOT_CONTROLLABLE	36
4.22.2.12	LIBTIEPIESTATUS_NOT_SUPPORTED	36
4.22.2.13	LIBTIEPIESTATUS_SUCCESS	36
4.22.2.14	LIBTIEPIESTATUS_UNSUCCESSFUL	36
4.22.2.15	LIBTIEPIESTATUS_VALUE_CLIPPED	36
4.22.2.16	LIBTIEPIESTATUS_VALUE_MODIFIED	36
4.23	Connector types	37
4.23.1	Detailed Description	37
4.23.2	Macro Definition Documentation	37
4.23.2.1	CONNECTORTYPE_BANANA	37
4.23.2.2	CONNECTORTYPE_BNC	37
4.23.2.3	CONNECTORTYPE_MASK	37
4.23.2.4	CONNECTORTYPE_UNKNOWN	37
4.24	Raw data types	38
4.24.1	Detailed Description	38
4.24.2	Macro Definition Documentation	38
4.24.2.1	DATARAWTYPE_INT16	38
4.24.2.2	DATARAWTYPE_INT32	38
4.24.2.3	DATARAWTYPE_INT64	38
4.24.2.4	DATARAWTYPE_INT8	38
4.24.2.5	DATARAWTYPE_UINT16	38
4.24.2.6	DATARAWTYPE_UINT32	38
4.24.2.7	DATARAWTYPE_UINT64	38
4.24.2.8	DATARAWTYPE_UINT8	38
4.24.2.9	DATARAWTYPE_UNKNOWN	39
4.25	DeviceID masks	40

4.25.1 Detailed Description	40
4.25.2 Macro Definition Documentation	40
4.25.2.1 IDM_ALL	40
4.25.2.2 IDM_DEVICEID	40
4.26 DeviceID bits	41
4.26.1 Detailed Description	41
4.26.2 Macro Definition Documentation	41
4.26.2.1 IDB_HL0516	41
4.26.2.2 IDB_HP3	41
4.26.2.3 IDB_HS3	41
4.26.2.4 IDB_HS4	41
4.26.2.5 IDB_HS4D	41
4.26.2.6 IDB_HS5	41
4.26.2.7 IDB_HS805	41
4.26.2.8 IDB_PA1	41
4.27 DeviceID's	42
4.27.1 Detailed Description	42
4.27.2 Macro Definition Documentation	42
4.27.2.1 ID_HL0516	42
4.27.2.2 ID_HP3	42
4.27.2.3 ID_HS3	42
4.27.2.4 ID_HS4	42
4.27.2.5 ID_HS4D	42
4.27.2.6 ID_HS5	43
4.27.2.7 ID_HS805	43
4.27.2.8 ID_PA1	43
4.28 Types	44
4.28.1 Detailed Description	44
4.28.2 Typedef Documentation	44
4.28.2.1 bool8_t	44
4.28.2.2 LibTiePieStatus_t	44
4.28.2.3 TpDate_t	44
4.28.2.4 TpDeviceHandle_t	44
4.28.2.5 TpVersion_t	44
4.29 Macro's	45
4.29.1 Detailed Description	45
4.29.2 Macro Definition Documentation	45
4.29.2.1 TPDATE_DAY	45
4.29.2.2 TPDATE_MONTH	45
4.29.2.3 TPDATE_YEAR	45

4.29.2.4	TPVERSION_BUILD	45
4.29.2.5	TPVERSION_MAJOR	45
4.29.2.6	TPVERSION_MINOR	45
4.29.2.7	TPVERSION_RELEASE	45
4.30	Messages	46
4.30.1	Detailed Description	46
4.30.2	Macro Definition Documentation	46
4.30.2.1	WM_LIBTIEPIE	46
4.30.2.2	WM_LIBTIEPIE_DEV_REMOVED	46
4.30.2.3	WM_LIBTIEPIE_GEN_BURSTCOMPLETED	46
4.30.2.4	WM_LIBTIEPIE_GEN_CONTROLLABLECHANGED	46
4.30.2.5	WM_LIBTIEPIE_LST_DEVICEADDED	46
4.30.2.6	WM_LIBTIEPIE_LST_DEVICEREMOVED	46
4.30.2.7	WM_LIBTIEPIE_SCP_DATAOVERFLOW	46
4.30.2.8	WM_LIBTIEPIE_SCP_DATAREADY	46
4.31	Functions	47
4.31.1	Detailed Description	47
4.32	Library	48
4.32.1	Detailed Description	48
4.32.2	Function Documentation	48
4.32.2.1	LibGetConfig	48
4.32.2.2	LibGetLastStatus	48
4.32.2.3	LibGetLastStatusStr	49
4.32.2.4	LibGetVersion	49
4.33	Device list	50
4.33.1	Detailed Description	50
4.33.2	Function Documentation	50
4.33.2.1	LstGetCount	50
4.33.2.2	LstGetDeviceCanOpen	50
4.33.2.3	LstGetDeviceName	50
4.33.2.4	LstGetDeviceNameShort	51
4.33.2.5	LstGetDeviceProductId	51
4.33.2.6	LstGetDeviceSerialNumber	51
4.33.2.7	LstGetDeviceVendorId	51
4.33.2.8	LstOpenDevice	52
4.33.2.9	LstRemoveDevice	52
4.33.2.10	LstUpdate	52
4.34	Events	53
4.34.1	Detailed Description	53
4.34.2	Function Documentation	53

4.34.2.1	LstSetCallbackDeviceAdded	53
4.34.2.2	LstSetCallbackDeviceRemoved	53
4.34.2.3	LstSetEventDeviceAdded	53
4.34.2.4	LstSetEventDeviceRemoved	53
4.34.2.5	LstSetMessageDeviceAdded	53
4.34.2.6	LstSetMessageDeviceRemoved	54
4.35	Device	55
4.35.1	Detailed Description	55
4.35.2	Function Documentation	55
4.35.2.1	DevClose	55
4.35.2.2	DevGetCalibrationDate	55
4.35.2.3	DevGetDriverVersion	56
4.35.2.4	DevGetFirmwareVersion	56
4.35.2.5	DevGetName	56
4.35.2.6	DevGetNameShort	57
4.35.2.7	DevGetProductId	57
4.35.2.8	DevGetSerialNumber	58
4.35.2.9	DevGetVendorId	58
4.35.2.10	DevIsRemoved	58
4.36	Events	59
4.36.1	Detailed Description	59
4.36.2	Function Documentation	59
4.36.2.1	DevSetCallbackRemoved	59
4.36.2.2	DevSetEventRemoved	59
4.36.2.3	DevSetMessageRemoved	59
4.37	Oscilloscope	60
4.37.1	Detailed Description	60
4.38	Input channels	61
4.38.1	Detailed Description	61
4.38.2	Function Documentation	61
4.38.2.1	ScpChGetAutoRanging	61
4.38.2.2	ScpChGetConnectorType	61
4.38.2.3	ScpChGetCoupling	62
4.38.2.4	ScpChGetCouplings	62
4.38.2.5	ScpChGetEnabled	62
4.38.2.6	ScpChGetProbeGain	63
4.38.2.7	ScpChGetRange	63
4.38.2.8	ScpChGetRanges	63
4.38.2.9	ScpChGetRangesEx	64
4.38.2.10	ScpChIsDifferential	64

4.38.2.11	ScpChIsDualRateCapable	64
4.38.2.12	ScpChSetAutoRanging	64
4.38.2.13	ScpChSetCoupling	65
4.38.2.14	ScpChSetEnabled	65
4.38.2.15	ScpChSetProbeGain	65
4.38.2.16	ScpChSetRange	65
4.38.2.17	ScpGetChannelCount	66
4.38.2.18	ScpGetSharedChannelGroup	66
4.38.2.19	ScpGetSharedChannelGroupCount	66
4.39	Data	67
4.39.1	Detailed Description	67
4.39.2	Function Documentation	67
4.39.2.1	ScpChGetDataRawType	67
4.39.2.2	ScpChGetDataRawValueMax	67
4.39.2.3	ScpChGetDataRawValueMin	68
4.39.2.4	ScpChGetDataRawValueRange	68
4.39.2.5	ScpChGetDataRawValueZero	68
4.39.2.6	ScpChGetDataValueMax	68
4.39.2.7	ScpChGetDataValueMin	68
4.39.2.8	ScpChGetDataValueRange	68
4.39.2.9	ScpChIsRangeMaxReachable	69
4.39.2.10	ScpGetData	69
4.39.2.11	ScpGetData1Ch	69
4.39.2.12	ScpGetData2Ch	69
4.39.2.13	ScpGetData3Ch	69
4.39.2.14	ScpGetData4Ch	70
4.39.2.15	ScpGetDataRaw	70
4.39.2.16	ScpGetDataRaw1Ch	70
4.39.2.17	ScpGetDataRaw2Ch	71
4.39.2.18	ScpGetDataRaw3Ch	71
4.39.2.19	ScpGetDataRaw4Ch	71
4.39.2.20	ScpGetValidPreSampleCount	71
4.40	Events	72
4.40.1	Detailed Description	72
4.40.2	Function Documentation	72
4.40.2.1	ScpSetCallbackDataOverflow	72
4.40.2.2	ScpSetCallbackDataReady	72
4.40.2.3	ScpSetEventDataOverflow	72
4.40.2.4	ScpSetEventDataReady	72
4.40.2.5	ScpSetMessageDataOverflow	72

4.40.2.6	ScpSetMessageDataReady	73
4.41	Measurements	74
4.41.1	Detailed Description	74
4.41.2	Function Documentation	74
4.41.2.1	ScpForceTrigger	74
4.41.2.2	ScpGetMeasureMode	74
4.41.2.3	ScpGetMeasureModes	74
4.41.2.4	ScplsDataOverflow	74
4.41.2.5	ScplsDataReady	75
4.41.2.6	ScplsRunning	75
4.41.2.7	ScplsTriggered	75
4.41.2.8	ScpSetMeasureMode	75
4.41.2.9	ScpStart	76
4.41.2.10	ScpStop	76
4.42	Resolution	77
4.42.1	Detailed Description	77
4.42.2	Function Documentation	77
4.42.2.1	ScpGetResolution	77
4.42.2.2	ScpGetResolutions	77
4.42.2.3	ScplsResolutionEnhanced	77
4.42.2.4	ScplsResolutionEnhancedEx	78
4.42.2.5	ScpSetResolution	78
4.43	Timebase	79
4.43.1	Detailed Description	79
4.43.2	Function Documentation	79
4.43.2.1	ScpGetClockSource	79
4.43.2.2	ScpGetPreSampleRatio	79
4.43.2.3	ScpGetRecordLength	80
4.43.2.4	ScpGetRecordLengthMax	80
4.43.2.5	ScpGetRecordLengthMaxEx	80
4.43.2.6	ScpGetSampleFrequency	80
4.43.2.7	ScpGetSampleFrequencyMax	80
4.43.2.8	ScpGetTriggerHoldOffCount	81
4.43.2.9	ScpGetTriggerHoldOffCountMax	81
4.43.2.10	ScpGetTriggerHoldOffCountMaxEx	81
4.43.2.11	ScpSetClockSource	81
4.43.2.12	ScpSetPreSampleRatio	82
4.43.2.13	ScpSetRecordLength	82
4.43.2.14	ScpSetSampleFrequency	82
4.43.2.15	ScpSetTriggerHoldOffCount	83

4.43.2.16 ScpVerifyRecordLength	83
4.43.2.17 ScpVerifyRecordLengthEx	83
4.43.2.18 ScpVerifySampleFrequency	83
4.43.2.19 ScpVerifySampleFrequencyEx	84
4.44 Trigger system	85
4.44.1 Detailed Description	85
4.44.2 Function Documentation	85
4.44.2.1 ScpChGetTriggerHysteresis	85
4.44.2.2 ScpChGetTriggerKind	86
4.44.2.3 ScpChGetTriggerKinds	86
4.44.2.4 ScpChGetTriggerKindsEx	86
4.44.2.5 ScpChGetTriggerLevel	87
4.44.2.6 ScpChGetTriggerPulseTime	87
4.44.2.7 ScpChSetTriggerHysteresis	87
4.44.2.8 ScpChSetTriggerKind	87
4.44.2.9 ScpChSetTriggerLevel	88
4.44.2.10 ScpChSetTriggerPulseTime	88
4.44.2.11 ScpGetTriggerHysteresis	88
4.44.2.12 ScpGetTriggerKind	88
4.44.2.13 ScpGetTriggerKinds	89
4.44.2.14 ScpGetTriggerKindsEx	89
4.44.2.15 ScpGetTriggerLevel	90
4.44.2.16 ScpGetTriggerSourceAND	90
4.44.2.17 ScpGetTriggerSourceOR	90
4.44.2.18 ScpGetTriggerSources	90
4.44.2.19 ScpGetTriggerSourcesEx	91
4.44.2.20 ScpGetTriggerTimeOut	91
4.44.2.21 ScpSetTriggerHysteresis	91
4.44.2.22 ScpSetTriggerKind	92
4.44.2.23 ScpSetTriggerLevel	92
4.44.2.24 ScpSetTriggerSourceAND	92
4.44.2.25 ScpSetTriggerSourceOR	93
4.44.2.26 ScpSetTriggerTimeOut	93
4.44.2.27 ScpVerifyTriggerTimeOut	93
4.45 Generator	95
4.45.1 Detailed Description	95
4.46 Info	96
4.46.1 Detailed Description	96
4.46.2 Function Documentation	96
4.46.2.1 GenGetConnectorType	96

4.46.2.2	GenGetImpedance	96
4.46.2.3	GenIsDifferential	96
4.47	Control	97
4.47.1	Detailed Description	97
4.47.2	Function Documentation	97
4.47.2.1	GenGetAmplitude	97
4.47.2.2	GenGetAmplitudeMax	98
4.47.2.3	GenGetAmplitudeMin	98
4.47.2.4	GenGetBurstCount	98
4.47.2.5	GenGetBurstCountMax	98
4.47.2.6	GenGetFrequency	98
4.47.2.7	GenGetFrequencyMax	99
4.47.2.8	GenGetFrequencyMin	99
4.47.2.9	GenGetFrequencyMinMax	99
4.47.2.10	GenGetMode	99
4.47.2.11	GenGetModes	100
4.47.2.12	GenGetModesEx	100
4.47.2.13	GenGetOffset	100
4.47.2.14	GenGetOffsetMax	100
4.47.2.15	GenGetOffsetMin	101
4.47.2.16	GenGetOutputOn	101
4.47.2.17	GenGetPhase	101
4.47.2.18	GenGetSignalType	101
4.47.2.19	GenGetSignalTypes	102
4.47.2.20	GenGetSymmetry	102
4.47.2.21	GenIsBurstActive	102
4.47.2.22	GenIsControllable	102
4.47.2.23	GenSetAmplitude	103
4.47.2.24	GenSetBurstCount	103
4.47.2.25	GenSetFrequency	103
4.47.2.26	GenSetMode	103
4.47.2.27	GenSetOffset	104
4.47.2.28	GenSetOutputOn	104
4.47.2.29	GenSetPhase	104
4.47.2.30	GenSetSignalType	104
4.47.2.31	GenSetSymmetry	105
4.47.2.32	GenStart	105
4.47.2.33	GenStop	105
4.47.2.34	GenVerifyAmplitude	105
4.47.2.35	GenVerifyFrequency	105

4.47.2.36	GenVerifyFrequencyEx	105
4.47.2.37	GenVerifyOffset	106
4.47.2.38	GenVerifyPhase	106
4.47.2.39	GenVerifySymmetry	106
4.48	Events	107
4.48.1	Detailed Description	107
4.48.2	Function Documentation	107
4.48.2.1	GenSetCallbackBurstCompleted	107
4.48.2.2	GenSetCallbackControllableChanged	107
4.48.2.3	GenSetEventBurstCompleted	107
4.48.2.4	GenSetEventControllableChanged	107
4.48.2.5	GenSetMessageBurstCompleted	107
4.48.2.6	GenSetMessageControllableChanged	108
4.49	Range	109
4.49.1	Detailed Description	109
4.49.2	Function Documentation	109
4.49.2.1	GenGetAutoRanging	109
4.49.2.2	GenGetRange	109
4.49.2.3	GenGetRanges	109
4.49.2.4	GenSetAutoRanging	109
4.49.2.5	GenSetRange	109
4.50	Trigger system	110
4.50.1	Detailed Description	110
4.50.2	Function Documentation	110
4.50.2.1	GenGetTriggerSourceAND	110
4.50.2.2	GenGetTriggerSourceOR	110
4.50.2.3	GenGetTriggerSources	110
4.50.2.4	GenSetTriggerSourceAND	111
4.50.2.5	GenSetTriggerSourceOR	111
4.51	Arbitrary	112
4.51.1	Detailed Description	112
4.51.2	Function Documentation	112
4.51.2.1	GenGetDataLength	112
4.51.2.2	GenGetDataLengthMax	112
4.51.2.3	GenGetDataRawType	112
4.51.2.4	GenSetData	113
4.51.2.5	GenSetDataRaw	113
4.51.2.6	GenVerifyDataLength	113
4.52	I2CHost	114
4.52.1	Detailed Description	114

4.52.2	Function Documentation	114
4.52.2.1	I2CGetSpeed	114
4.52.2.2	I2CGetSpeedMax	114
4.52.2.3	I2CIsInternalAddress	114
4.52.2.4	I2CRead	114
4.52.2.5	I2CReadByte	114
4.52.2.6	I2CReadWord	114
4.52.2.7	I2CSetSpeed	114
4.52.2.8	I2CVerifySpeed	114
4.52.2.9	I2CWrite	114
4.52.2.10	I2CWriteByte	114
4.52.2.11	I2CWriteByteByte	114
4.52.2.12	I2CWriteByteWord	114
4.52.2.13	I2CWriteWord	114
4.53	Types	115
4.53.1	Detailed Description	115
4.53.2	Typedef Documentation	115
4.53.2.1	TpCallback_t	115
5	File Documentation	117
5.1	libtiepie.h File Reference	117
5.1.1	Detailed Description	127

Chapter 1

Main Page

1.1 Supported instruments

- Handyscope HS5

1.2 Error handling

On each function call [LibGetLastStatus\(\)](#) is set, see [Status return codes](#).

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Constants	7
Clock modes	9
Coupling type bit numbers	10
Coupling types	11
Coupling type masks	13
Generator mode bit numbers	14
Generator modes	15
Measure modes bit numbers	16
Measure modes	17
Signal type bit numbers	18
Signal types	19
Signal type masks	20
Trigger holdoff	21
Trigger kinds bit numbers	22
Trigger kinds	23
Trigger kind masks	24
Time outs	25
Trigger sources	26
Trigger source masks	32
TiePie device handles	33
Device type	34
Status return codes	35
Connector types	37
Raw data types	38
DeviceID masks	40
DeviceID bits	41
DeviceID's	42
Types	44
Macro's	45
Messages	46
Functions	47
Library	48
Device list	50
Events	53
Device	55
Events	59
Oscilloscope	60

Input channels	61
Data	67
Events	72
Measurements	74
Resolution	77
Timebase	79
Trigger system	85
Generator	95
Info	96
Control	97
Events	107
Range	109
Trigger system	110
Arbitrary	112
I2CHost	114
Types	115

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

libtiepie.h	Header for libtiepie	117
-----------------------------	--------------------------------	-----

Chapter 4

Module Documentation

4.1 Constants

Modules

- [Clock modes](#)
- [Coupling type bit numbers](#)
- [Coupling types](#)
- [Coupling type masks](#)
- [Generator mode bit numbers](#)
- [Generator modes](#)
- [Measure modes bit numbers](#)
- [Measure modes](#)
- [Signal type bit numbers](#)
- [Signal types](#)
- [Signal type masks](#)
- [Trigger holdoff](#)
- [Trigger kinds bit numbers](#)
- [Trigger kinds](#)
- [Trigger kind masks](#)
- [Time outs](#)
- [Trigger sources](#)
- [Trigger source masks](#)
- [TiePie device handles](#)
- [Device type](#)
- [Status return codes](#)
- [Connector types](#)
- [Raw data types](#)
- [DeviceID masks](#)
- [DeviceID bits](#)
- [DeviceID's](#)

Macros

- `#define CKN_COUPLING_COUNT 5`
Number of couplings.
- `#define FMN_GENERATORMODE_COUNT 2`
Number of generator modes.
- `#define MMN_MEASUREMODE_COUNT 2`

- Number of measure modes.*
 - `#define STN_SIGNALTYPE_COUNT` 6
- Number of types.*
 - `#define TKN_KIND_COUNT` 7
- Number of kinds.*
 - `#define TSN_CHANNEL_COUNT` 32
- Number of LSBs reserved for channel trigger sources.*

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 `#define CKN_COUPLING_COUNT` 5

Number of couplings.

Definition at line 58 of file libtiepie.h.

4.1.2.2 `#define FMN_GENERATORMODE_COUNT` 2

Number of generator modes.

Definition at line 102 of file libtiepie.h.

4.1.2.3 `#define MMN_MEASUREMODE_COUNT` 2

Number of measure modes.

Definition at line 130 of file libtiepie.h.

4.1.2.4 `#define STN_SIGNALTYPE_COUNT` 6

Number of types.

Definition at line 158 of file libtiepie.h.

4.1.2.5 `#define TKN_KIND_COUNT` 7

Number of kinds.

Definition at line 217 of file libtiepie.h.

4.1.2.6 `#define TSN_CHANNEL_COUNT` 32

Number of LSBs reserved for channel trigger sources.

Definition at line 325 of file libtiepie.h.

4.2 Clock modes

Macros

- `#define CM_INTERNAL 0`
Internal clock source.
- `#define CM_EXTERNAL 1`
External clock source.

4.2.1 Detailed Description

4.2.2 Macro Definition Documentation

4.2.2.1 `#define CM_EXTERNAL 1`

External clock source.

Definition at line 49 of file libtiepie.h.

4.2.2.2 `#define CM_INTERNAL 0`

Internal clock source.

Definition at line 48 of file libtiepie.h.

4.3 Coupling type bit numbers

Macros

- `#define CKB_DCV 0`
Volt DC.
- `#define CKB_ACV 1`
Volt AC.
- `#define CKB_DCA 2`
Ampere DC.
- `#define CKB_ACA 3`
Ampere AC.
- `#define CKB_OHM 4`
Ohm.

4.3.1 Detailed Description

4.3.2 Macro Definition Documentation

4.3.2.1 `#define CKB_ACA 3`

Ampere AC.

Definition at line 68 of file libtiepie.h.

4.3.2.2 `#define CKB_ACV 1`

Volt AC.

Definition at line 66 of file libtiepie.h.

4.3.2.3 `#define CKB_DCA 2`

Ampere DC.

Definition at line 67 of file libtiepie.h.

4.3.2.4 `#define CKB_DCV 0`

Volt DC.

Definition at line 65 of file libtiepie.h.

4.3.2.5 `#define CKB_OHM 4`

Ohm.

Definition at line 69 of file libtiepie.h.

4.4 Coupling types

Macros

- #define CK_UNKNOWN 0x0000000000000000
Invalid coupling type.
- #define CK_DCV (1 << CKB_DCV)
Volt DC.
- #define CK_ACV (1 << CKB_ACV)
Volt AC.
- #define CK_DCA (1 << CKB_DCA)
Ampere DC.
- #define CK_ACA (1 << CKB_ACA)
Ampere AC.
- #define CK_OHM (1 << CKB_OHM)
Ohm.

4.4.1 Detailed Description

4.4.2 Macro Definition Documentation

4.4.2.1 #define CK_ACA (1 << CKB_ACA)

Ampere AC.

Definition at line 82 of file libtiepie.h.

4.4.2.2 #define CK_ACV (1 << CKB_ACV)

Volt AC.

Definition at line 80 of file libtiepie.h.

4.4.2.3 #define CK_DCA (1 << CKB_DCA)

Ampere DC.

Definition at line 81 of file libtiepie.h.

4.4.2.4 #define CK_DCV (1 << CKB_DCV)

Volt DC.

Definition at line 79 of file libtiepie.h.

4.4.2.5 #define CK_OHM (1 << CKB_OHM)

Ohm.

Definition at line 83 of file libtiepie.h.

4.4.2.6 `#define CK_UNKNOWN 0x0000000000000000`

Invalid coupling type.

Definition at line 77 of file libtiepie.h.

4.5 Coupling type masks

Macros

- `#define CKM_V ( CK_DCV | CK_ACV )`
Volt.
- `#define CKM_A ( CK_DCA | CK_ACA )`
Ampere.
- `#define CKM_OHM ( CK_OHM )`
Ohm.

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 `#define CKM_A ( CK_DCA | CK_ACA )`

Ampere.

Definition at line 92 of file libtiepie.h.

4.5.2.2 `#define CKM_OHM ( CK_OHM )`

Ohm.

Definition at line 93 of file libtiepie.h.

4.5.2.3 `#define CKM_V ( CK_DCV | CK_ACV )`

Volt.

Definition at line 91 of file libtiepie.h.

4.6 Generator mode bit numbers

Macros

- `#define FMB_SIGNALFREQUENCY 0`
- `#define FMB_SAMPLEFREQUENCY 1`

4.6.1 Detailed Description

4.6.2 Macro Definition Documentation

4.6.2.1 `#define FMB_SAMPLEFREQUENCY 1`

Definition at line 110 of file libtiepie.h.

4.6.2.2 `#define FMB_SIGNALFREQUENCY 0`

Definition at line 109 of file libtiepie.h.

4.7 Generator modes

Macros

- `#define FM_UNKNOWN 0x00000000`
- `#define FM_SIGNALFREQUENCY ( 1 << FMB_SIGNALFREQUENCY )`
- `#define FM_SAMPLEFREQUENCY ( 1 << FMB_SAMPLEFREQUENCY )`

4.7.1 Detailed Description

4.7.2 Macro Definition Documentation

4.7.2.1 `#define FM_SAMPLEFREQUENCY ( 1 << FMB_SAMPLEFREQUENCY )`

Definition at line 121 of file libtiepie.h.

4.7.2.2 `#define FM_SIGNALFREQUENCY ( 1 << FMB_SIGNALFREQUENCY )`

Definition at line 120 of file libtiepie.h.

4.7.2.3 `#define FM_UNKNOWN 0x00000000`

Definition at line 118 of file libtiepie.h.

4.8 Measure modes bit numbers

Macros

- `#define MMB_STREAM 0`
Stream mode bit number.
- `#define MMB_BLOCK 1`
Block mode bit number.

4.8.1 Detailed Description

4.8.2 Macro Definition Documentation

4.8.2.1 `#define MMB_BLOCK 1`

Block mode bit number.

Definition at line 138 of file libtiepie.h.

4.8.2.2 `#define MMB_STREAM 0`

Stream mode bit number.

Definition at line 137 of file libtiepie.h.

4.9 Measure modes

Macros

- `#define MM_UNKNOWN 0x00000000`
- `#define MM_STREAM ( 1 << MMB_STREAM )`
- `#define MM_BLOCK ( 1 << MMB_BLOCK )`

4.9.1 Detailed Description

4.9.2 Macro Definition Documentation

4.9.2.1 `#define MM_BLOCK ( 1 << MMB_BLOCK )`

Definition at line 149 of file libtiepie.h.

4.9.2.2 `#define MM_STREAM ( 1 << MMB_STREAM )`

Definition at line 148 of file libtiepie.h.

4.9.2.3 `#define MM_UNKNOWN 0x00000000`

Definition at line 146 of file libtiepie.h.

4.10 Signal type bit numbers

Macros

- `#define STB_SINE 0`
- `#define STB_TRIANGLE 1`
- `#define STB_SQUARE 2`
- `#define STB_DC 3`
- `#define STB_NOISE 4`
- `#define STB_ARBITRARY 5`

4.10.1 Detailed Description

4.10.2 Macro Definition Documentation

4.10.2.1 `#define STB_ARBITRARY 5`

Definition at line 170 of file libtiepie.h.

4.10.2.2 `#define STB_DC 3`

Definition at line 168 of file libtiepie.h.

4.10.2.3 `#define STB_NOISE 4`

Definition at line 169 of file libtiepie.h.

4.10.2.4 `#define STB_SINE 0`

Definition at line 165 of file libtiepie.h.

4.10.2.5 `#define STB_SQUARE 2`

Definition at line 167 of file libtiepie.h.

4.10.2.6 `#define STB_TRIANGLE 1`

Definition at line 166 of file libtiepie.h.

4.11 Signal types

Macros

- `#define ST_UNKNOWN 0x0000000000000000`
- `#define ST_SINE ( 1 << STB_SINE )`
- `#define ST_TRIANGLE ( 1 << STB_TRIANGLE )`
- `#define ST_SQUARE ( 1 << STB_SQUARE )`
- `#define ST_DC ( 1 << STB_DC )`
- `#define ST_NOISE ( 1 << STB_NOISE )`
- `#define ST_ARBITRARY ( 1 << STB_ARBITRARY )`

4.11.1 Detailed Description

4.11.2 Macro Definition Documentation

4.11.2.1 `#define ST_ARBITRARY ( 1 << STB_ARBITRARY )`

Definition at line 185 of file libtiepie.h.

4.11.2.2 `#define ST_DC ( 1 << STB_DC )`

Definition at line 183 of file libtiepie.h.

4.11.2.3 `#define ST_NOISE ( 1 << STB_NOISE )`

Definition at line 184 of file libtiepie.h.

4.11.2.4 `#define ST_SINE ( 1 << STB_SINE )`

Definition at line 180 of file libtiepie.h.

4.11.2.5 `#define ST_SQUARE ( 1 << STB_SQUARE )`

Definition at line 182 of file libtiepie.h.

4.11.2.6 `#define ST_TRIANGLE ( 1 << STB_TRIANGLE )`

Definition at line 181 of file libtiepie.h.

4.11.2.7 `#define ST_UNKNOWN 0x0000000000000000`

Definition at line 178 of file libtiepie.h.

4.12 Signal type masks

Macros

- `#define STM_AMPLITUDE ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_NOISE | ST_ARBITRARY )`
- `#define STM_OFFSET ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_DC | ST_NOISE | ST_ARBITRARY )`
- `#define STM_FREQUENCY ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY )`
- `#define STM_PHASE ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY )`
- `#define STM_SYMMETRY ( ST_SINE | ST_TRIANGLE | ST_SQUARE )`

4.12.1 Detailed Description

4.12.2 Macro Definition Documentation

4.12.2.1 `#define STM_AMPLITUDE ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_NOISE | ST_ARBITRARY )`

Definition at line 193 of file libtiepie.h.

4.12.2.2 `#define STM_FREQUENCY ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY )`

Definition at line 195 of file libtiepie.h.

4.12.2.3 `#define STM_OFFSET ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_DC | ST_NOISE | ST_ARBITRARY )`

Definition at line 194 of file libtiepie.h.

4.12.2.4 `#define STM_PHASE ( ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY )`

Definition at line 196 of file libtiepie.h.

4.12.2.5 `#define STM_SYMMETRY ( ST_SINE | ST_TRIANGLE | ST_SQUARE )`

Definition at line 197 of file libtiepie.h.

4.13 Trigger holdoff

Macros

- `#define TH_ALLPRESAMPLES 0xffffffffffff`
All presamples.

4.13.1 Detailed Description

4.13.2 Macro Definition Documentation

4.13.2.1 `#define TH_ALLPRESAMPLES 0xffffffffffff`

All presamples.

Definition at line 208 of file libtiepie.h.

4.14 Trigger kinds bit numbers

Macros

- `#define TKB_RISING 0`
- `#define TKB_FALLING 1`
- `#define TKB_INWINDOW 2`
- `#define TKB_OUTWINDOW 3`
- `#define TKB_EDGE 4`
- `#define TKB_DROPINWINDOW 5`
- `#define TKB_DROPOUTWINDOW 6`

4.14.1 Detailed Description

4.14.2 Macro Definition Documentation

4.14.2.1 `#define TKB_DROPINWINDOW 5`

Definition at line 229 of file libtiepie.h.

4.14.2.2 `#define TKB_DROPOUTWINDOW 6`

Definition at line 230 of file libtiepie.h.

4.14.2.3 `#define TKB_EDGE 4`

Definition at line 228 of file libtiepie.h.

4.14.2.4 `#define TKB_FALLING 1`

Definition at line 225 of file libtiepie.h.

4.14.2.5 `#define TKB_INWINDOW 2`

Definition at line 226 of file libtiepie.h.

4.14.2.6 `#define TKB_OUTWINDOW 3`

Definition at line 227 of file libtiepie.h.

4.14.2.7 `#define TKB_RISING 0`

Definition at line 224 of file libtiepie.h.

4.15 Trigger kinds

Macros

- `#define TK_UNKNOWN 0x0000000000000000ULL`
- `#define TK_RISING ( 1ULL << TKB_RISING )`
- `#define TK_FALLING ( 1ULL << TKB_FALLING )`
- `#define TK_INWINDOW ( 1ULL << TKB_INWINDOW )`
- `#define TK_OUTWINDOW ( 1ULL << TKB_OUTWINDOW )`
- `#define TK_EDGE ( 1ULL << TKB_EDGE )`
- `#define TK_DROPINWINDOW ( 1ULL << TKB_DROPINWINDOW )`
- `#define TK_DROPOUTWINDOW ( 1ULL << TKB_DROPOUTWINDOW )`

4.15.1 Detailed Description

4.15.2 Macro Definition Documentation

4.15.2.1 `#define TK_DROPINWINDOW ( 1ULL << TKB_DROPINWINDOW )`

Definition at line 245 of file libtiepie.h.

4.15.2.2 `#define TK_DROPOUTWINDOW ( 1ULL << TKB_DROPOUTWINDOW )`

Definition at line 246 of file libtiepie.h.

4.15.2.3 `#define TK_EDGE ( 1ULL << TKB_EDGE )`

Definition at line 244 of file libtiepie.h.

4.15.2.4 `#define TK_FALLING ( 1ULL << TKB_FALLING )`

Definition at line 241 of file libtiepie.h.

4.15.2.5 `#define TK_INWINDOW ( 1ULL << TKB_INWINDOW )`

Definition at line 242 of file libtiepie.h.

4.15.2.6 `#define TK_OUTWINDOW ( 1ULL << TKB_OUTWINDOW )`

Definition at line 243 of file libtiepie.h.

4.15.2.7 `#define TK_RISING ( 1ULL << TKB_RISING )`

Definition at line 240 of file libtiepie.h.

4.15.2.8 `#define TK_UNKNOWN 0x0000000000000000ULL`

Definition at line 238 of file libtiepie.h.

4.16 Trigger kind masks

Macros

- `#define TKM_NONE 0x0000000000000000ULL`
No trigger kinds.
- `#define TKM_EDGE ( TK_RISING | TK_FALLING | TK_EDGE )`
- `#define TKM_WINDOW ( TK_INWINDOW | TK_OUTWINDOW | TK_DROPINWINDOW | TK_DROPOUTWINDOW )`
- `#define TKM_ALL ( ( 1ULL << TKN_KIND_COUNT ) - 1 )`
All trigger kinds.

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 `#define TKM_ALL ( ( 1ULL << TKN_KIND_COUNT ) - 1 )`

All trigger kinds.

Definition at line 257 of file libtiepie.h.

4.16.2.2 `#define TKM_EDGE ( TK_RISING | TK_FALLING | TK_EDGE )`

Definition at line 255 of file libtiepie.h.

4.16.2.3 `#define TKM_NONE 0x0000000000000000ULL`

No trigger kinds.

Definition at line 254 of file libtiepie.h.

4.16.2.4 `#define TKM_WINDOW ( TK_INWINDOW | TK_OUTWINDOW | TK_DROPINWINDOW | TK_DROPOUTWINDOW )`

Definition at line 256 of file libtiepie.h.

4.17 Time outs

Macros

- `#define TO_INFINITY -1`
No time out.

4.17.1 Detailed Description

4.17.2 Macro Definition Documentation

4.17.2.1 `#define TO_INFINITY -1`

No time out.

Definition at line 268 of file libtiepie.h.

4.18 Trigger sources

Macros

- #define `TS_NONE` 0x0000000000000000ULL
No trigger source.
- #define `TS_CH1` 0x0000000000000001ULL
Channel 1.
- #define `TS_CH2` 0x0000000000000002ULL
Channel 2.
- #define `TS_CH3` 0x0000000000000004ULL
Channel 3.
- #define `TS_CH4` 0x0000000000000008ULL
Channel 4.
- #define `TS_CH5` 0x0000000000000010ULL
Channel 5.
- #define `TS_CH6` 0x0000000000000020ULL
Channel 6.
- #define `TS_CH7` 0x0000000000000040ULL
Channel 7.
- #define `TS_CH8` 0x0000000000000080ULL
Channel 8.
- #define `TS_CH9` 0x0000000000000100ULL
Channel 9.
- #define `TS_CH10` 0x0000000000000200ULL
Channel 10.
- #define `TS_CH11` 0x0000000000000400ULL
Channel 11.
- #define `TS_CH12` 0x0000000000000800ULL
Channel 12.
- #define `TS_CH13` 0x0000000000001000ULL
Channel 13.
- #define `TS_CH14` 0x0000000000002000ULL
Channel 14.
- #define `TS_CH15` 0x0000000000004000ULL
Channel 15.
- #define `TS_CH16` 0x0000000000008000ULL
Channel 16.
- #define `TS_CH17` 0x0000000000010000ULL
Channel 17.
- #define `TS_CH18` 0x0000000000020000ULL
Channel 18.
- #define `TS_CH19` 0x0000000000040000ULL
Channel 19.
- #define `TS_CH20` 0x0000000000080000ULL
Channel 20.
- #define `TS_CH21` 0x0000000000100000ULL
Channel 21.
- #define `TS_CH22` 0x0000000000200000ULL
Channel 22.
- #define `TS_CH23` 0x0000000000400000ULL

- Channel 23.*
- #define **TS_CH24** 0x0000000000800000ULL
- Channel 24.*
- #define **TS_CH25** 0x0000000001000000ULL
- Channel 25.*
- #define **TS_CH26** 0x0000000002000000ULL
- Channel 26.*
- #define **TS_CH27** 0x0000000004000000ULL
- Channel 27.*
- #define **TS_CH28** 0x0000000008000000ULL
- Channel 28.*
- #define **TS_CH29** 0x0000000010000000ULL
- Channel 29.*
- #define **TS_CH30** 0x0000000020000000ULL
- Channel 30.*
- #define **TS_CH31** 0x0000000040000000ULL
- Channel 31.*
- #define **TS_CH32** 0x0000000080000000ULL
- Channel 32.*
- #define **TS_GENSTOP** 0x0400000000000000ULL
- Generator stop.*
- #define **TS_GENNEW** 0x0800000000000000ULL
- Generator new period.*
- #define **TS_GENSTART** 0x1000000000000000ULL
- Generator start.*
- #define **TS_EXT2** 0x2000000000000000ULL
- External 2 (TTL)*
- #define **TS_EXTANALOG** 0x4000000000000000ULL
- External (Analog)*
- #define **TS_EXT** 0x8000000000000000ULL
- External (TTL)*

4.18.1 Detailed Description

4.18.2 Macro Definition Documentation

4.18.2.1 #define TS_CH1 0x0000000000000001ULL

Channel 1.

Definition at line 281 of file libtiepie.h.

4.18.2.2 #define TS_CH10 0x0000000000000200ULL

Channel 10.

Definition at line 290 of file libtiepie.h.

4.18.2.3 #define TS_CH11 0x0000000000000400ULL

Channel 11.

Definition at line 291 of file libtiepie.h.

4.18.2.4 `#define TS_CH12 0x0000000000000800ULL`

Channel 12.

Definition at line 292 of file libtiepie.h.

4.18.2.5 `#define TS_CH13 0x0000000000001000ULL`

Channel 13.

Definition at line 293 of file libtiepie.h.

4.18.2.6 `#define TS_CH14 0x0000000000002000ULL`

Channel 14.

Definition at line 294 of file libtiepie.h.

4.18.2.7 `#define TS_CH15 0x0000000000004000ULL`

Channel 15.

Definition at line 295 of file libtiepie.h.

4.18.2.8 `#define TS_CH16 0x0000000000008000ULL`

Channel 16.

Definition at line 296 of file libtiepie.h.

4.18.2.9 `#define TS_CH17 0x0000000000010000ULL`

Channel 17.

Definition at line 297 of file libtiepie.h.

4.18.2.10 `#define TS_CH18 0x0000000000020000ULL`

Channel 18.

Definition at line 298 of file libtiepie.h.

4.18.2.11 `#define TS_CH19 0x0000000000040000ULL`

Channel 19.

Definition at line 299 of file libtiepie.h.

4.18.2.12 `#define TS_CH2 0x0000000000000002ULL`

Channel 2.

Definition at line 282 of file libtiepie.h.

4.18.2.13 `#define TS_CH20 0x0000000000080000ULL`

Channel 20.

Definition at line 300 of file libtiepie.h.

4.18.2.14 `#define TS_CH21 0x0000000000100000ULL`

Channel 21.

Definition at line 301 of file libtiepie.h.

4.18.2.15 `#define TS_CH22 0x0000000000200000ULL`

Channel 22.

Definition at line 302 of file libtiepie.h.

4.18.2.16 `#define TS_CH23 0x0000000000400000ULL`

Channel 23.

Definition at line 303 of file libtiepie.h.

4.18.2.17 `#define TS_CH24 0x0000000000800000ULL`

Channel 24.

Definition at line 304 of file libtiepie.h.

4.18.2.18 `#define TS_CH25 0x0000000001000000ULL`

Channel 25.

Definition at line 305 of file libtiepie.h.

4.18.2.19 `#define TS_CH26 0x0000000002000000ULL`

Channel 26.

Definition at line 306 of file libtiepie.h.

4.18.2.20 `#define TS_CH27 0x0000000004000000ULL`

Channel 27.

Definition at line 307 of file libtiepie.h.

4.18.2.21 `#define TS_CH28 0x0000000008000000ULL`

Channel 28.

Definition at line 308 of file libtiepie.h.

4.18.2.22 `#define TS_CH29 0x0000000010000000ULL`

Channel 29.

Definition at line 309 of file libtiepie.h.

4.18.2.23 `#define TS_CH3 0x0000000000000004ULL`

Channel 3.

Definition at line 283 of file libtiepie.h.

4.18.2.24 `#define TS_CH30 0x0000000020000000ULL`

Channel 30.

Definition at line 310 of file libtiepie.h.

4.18.2.25 `#define TS_CH31 0x0000000040000000ULL`

Channel 31.

Definition at line 311 of file libtiepie.h.

4.18.2.26 `#define TS_CH32 0x0000000080000000ULL`

Channel 32.

Definition at line 312 of file libtiepie.h.

4.18.2.27 `#define TS_CH4 0x0000000000000008ULL`

Channel 4.

Definition at line 284 of file libtiepie.h.

4.18.2.28 `#define TS_CH5 0x0000000000000010ULL`

Channel 5.

Definition at line 285 of file libtiepie.h.

4.18.2.29 `#define TS_CH6 0x0000000000000020ULL`

Channel 6.

Definition at line 286 of file libtiepie.h.

4.18.2.30 `#define TS_CH7 0x0000000000000040ULL`

Channel 7.

Definition at line 287 of file libtiepie.h.

4.18.2.31 `#define TS_CH8 0x0000000000000080ULL`

Channel 8.

Definition at line 288 of file libtiepie.h.

4.18.2.32 `#define TS_CH9 0x0000000000000100ULL`

Channel 9.

Definition at line 289 of file libtiepie.h.

4.18.2.33 `#define TS_EXT 0x8000000000000000ULL`

External (TTL)

Definition at line 319 of file libtiepie.h.

4.18.2.34 `#define TS_EXT2 0x2000000000000000ULL`

External 2 (TTL)

Definition at line 317 of file libtiepie.h.

4.18.2.35 `#define TS_EXTANALOG 0x4000000000000000ULL`

External (Analog)

Definition at line 318 of file libtiepie.h.

4.18.2.36 `#define TS_GENNEW 0x0800000000000000ULL`

Generator new period.

Definition at line 315 of file libtiepie.h.

4.18.2.37 `#define TS_GENSTART 0x1000000000000000ULL`

Generator start.

Definition at line 316 of file libtiepie.h.

4.18.2.38 `#define TS_GENSTOP 0x0400000000000000ULL`

Generator stop.

Definition at line 314 of file libtiepie.h.

4.18.2.39 `#define TS_NONE 0x0000000000000000ULL`

No trigger source.

Definition at line 279 of file libtiepie.h.

4.19 Trigger source masks

Macros

- `#define TSM_NONE 0x0000000000000000ULL`
No trigger sources.
- `#define TSM_ALL 0xFFFFFFFFFFFFFFFFFULL`
All trigger sources.
- `#define TSM_CHANNELS ( ( 1ULL << TSN_CHANNEL_COUNT ) - 1 )`
All channel trigger sources.
- `#define TSM_NONCHANNELS ( TSM_ALL - TSM_CHANNELS )`
All non-channel trigger sources.
- `#define TSM_GENALL ( TS_GENSTART | TS_GENNEW | TS_GENSTOP )`
All generator trigger sources.

4.19.1 Detailed Description

4.19.2 Macro Definition Documentation

4.19.2.1 `#define TSM_ALL 0xFFFFFFFFFFFFFFFFFULL`

All trigger sources.

Definition at line 333 of file libtiepie.h.

4.19.2.2 `#define TSM_CHANNELS ( ( 1ULL << TSN_CHANNEL_COUNT ) - 1 )`

All channel trigger sources.

Definition at line 334 of file libtiepie.h.

4.19.2.3 `#define TSM_GENALL ( TS_GENSTART | TS_GENNEW | TS_GENSTOP )`

All generator trigger sources.

Definition at line 336 of file libtiepie.h.

4.19.2.4 `#define TSM_NONCHANNELS ( TSM_ALL - TSM_CHANNELS )`

All non-channel trigger sources.

Definition at line 335 of file libtiepie.h.

4.19.2.5 `#define TSM_NONE 0x0000000000000000ULL`

No trigger sources.

Definition at line 332 of file libtiepie.h.

4.20 TiePie device handles

Macros

- `#define TPDEVICEHANDLE_INVALID 0`

4.20.1 Detailed Description

4.20.2 Macro Definition Documentation

4.20.2.1 `#define TPDEVICEHANDLE_INVALID 0`

Definition at line 363 of file libtiepie.h.

4.21 Device type

Macros

- `#define DEVICETYPE_OSCILLOSCOPE 0x00000001`
- `#define DEVICETYPE_GENERATOR 0x00000002`
- `#define DEVICETYPE_I2CHOST 0x00000004`

4.21.1 Detailed Description

4.21.2 Macro Definition Documentation

4.21.2.1 `#define DEVICETYPE_GENERATOR 0x00000002`

Definition at line 372 of file libtiepie.h.

4.21.2.2 `#define DEVICETYPE_I2CHOST 0x00000004`

Definition at line 373 of file libtiepie.h.

4.21.2.3 `#define DEVICETYPE_OSCILLOSCOPE 0x00000001`

Definition at line 371 of file libtiepie.h.

4.22 Status return codes

Macros

- `#define LIBTIEPIESTATUS_SUCCESS 0`
- `#define LIBTIEPIESTATUS_VALUE_CLIPPED 1`
- `#define LIBTIEPIESTATUS_VALUE_MODIFIED 2`
- `#define LIBTIEPIESTATUS_UNSUCCESSFUL -1`
- `#define LIBTIEPIESTATUS_NOT_SUPPORTED -2`
- `#define LIBTIEPIESTATUS_INVALID_HANDLE -3`
- `#define LIBTIEPIESTATUS_INVALID_VALUE -4`
- `#define LIBTIEPIESTATUS_INVALID_CHANNEL -5`
- `#define LIBTIEPIESTATUS_INVALID_TRIGGER_SOURCE -6`
- `#define LIBTIEPIESTATUS_INVALID_DEVICE_TYPE -7`
- `#define LIBTIEPIESTATUS_INVALID_DEVICE_INDEX -8`
- `#define LIBTIEPIESTATUS_INVALID_DEVICE_ID -9`
- `#define LIBTIEPIESTATUS_INVALID_DEVICE_SERIALNUMBER -10`
- `#define LIBTIEPIESTATUS_DEVICE_GONE -11`
- `#define LIBTIEPIESTATUS_INTERNAL_ADDRESS -12`
- `#define LIBTIEPIESTATUS_NOT_CONTROLLABLE -13`

4.22.1 Detailed Description

0 means ok

<0 means error

>0 means ok, but with a side effect

4.22.2 Macro Definition Documentation

4.22.2.1 `#define LIBTIEPIESTATUS_DEVICE_GONE -11`

Definition at line 398 of file libtiepie.h.

4.22.2.2 `#define LIBTIEPIESTATUS_INTERNAL_ADDRESS -12`

Definition at line 399 of file libtiepie.h.

4.22.2.3 `#define LIBTIEPIESTATUS_INVALID_CHANNEL -5`

Definition at line 392 of file libtiepie.h.

4.22.2.4 `#define LIBTIEPIESTATUS_INVALID_DEVICE_ID -9`

Definition at line 396 of file libtiepie.h.

4.22.2.5 `#define LIBTIEPIESTATUS_INVALID_DEVICE_INDEX -8`

Definition at line 395 of file libtiepie.h.

4.22.2.6 `#define LIBTIEPIESTATUS_INVALID_DEVICE_SERIALNUMBER -10`

Definition at line 397 of file libtiepie.h.

4.22.2.7 `#define LIBTIEPIESTATUS_INVALID_DEVICE_TYPE -7`

Definition at line 394 of file libtiepie.h.

4.22.2.8 `#define LIBTIEPIESTATUS_INVALID_HANDLE -3`

Definition at line 390 of file libtiepie.h.

4.22.2.9 `#define LIBTIEPIESTATUS_INVALID_TRIGGER_SOURCE -6`

Definition at line 393 of file libtiepie.h.

4.22.2.10 `#define LIBTIEPIESTATUS_INVALID_VALUE -4`

Definition at line 391 of file libtiepie.h.

4.22.2.11 `#define LIBTIEPIESTATUS_NOT_CONTROLLABLE -13`

Definition at line 400 of file libtiepie.h.

4.22.2.12 `#define LIBTIEPIESTATUS_NOT_SUPPORTED -2`

Definition at line 389 of file libtiepie.h.

4.22.2.13 `#define LIBTIEPIESTATUS_SUCCESS 0`

Definition at line 385 of file libtiepie.h.

4.22.2.14 `#define LIBTIEPIESTATUS_UNSUCCESSFUL -1`

Definition at line 388 of file libtiepie.h.

4.22.2.15 `#define LIBTIEPIESTATUS_VALUE_CLIPPED 1`

Definition at line 386 of file libtiepie.h.

4.22.2.16 `#define LIBTIEPIESTATUS_VALUE_MODIFIED 2`

Definition at line 387 of file libtiepie.h.

4.23 Connector types

Macros

- `#define CONNECTORTYPE_UNKNOWN 0x00000000`
- `#define CONNECTORTYPE_BNC 0x00000001`
- `#define CONNECTORTYPE_BANANA 0x00000002`
- `#define CONNECTORTYPE_MASK ( CONNECTORTYPE_BNC | CONNECTORTYPE_BANANA )`

4.23.1 Detailed Description

4.23.2 Macro Definition Documentation

4.23.2.1 `#define CONNECTORTYPE_BANANA 0x00000002`

Definition at line 411 of file libtiepie.h.

4.23.2.2 `#define CONNECTORTYPE_BNC 0x00000001`

Definition at line 410 of file libtiepie.h.

4.23.2.3 `#define CONNECTORTYPE_MASK ( CONNECTORTYPE_BNC | CONNECTORTYPE_BANANA )`

Definition at line 413 of file libtiepie.h.

4.23.2.4 `#define CONNECTORTYPE_UNKNOWN 0x00000000`

Definition at line 408 of file libtiepie.h.

4.24 Raw data types

Macros

- `#define DATARAWTYPE_UNKNOWN 0x00000000`
- `#define DATARAWTYPE_INT8 0x00000001`
- `#define DATARAWTYPE_INT16 0x00000002`
- `#define DATARAWTYPE_INT32 0x00000004`
- `#define DATARAWTYPE_INT64 0x00000008`
- `#define DATARAWTYPE_UINT8 0x00000010`
- `#define DATARAWTYPE_UINT16 0x00000020`
- `#define DATARAWTYPE_UINT32 0x00000040`
- `#define DATARAWTYPE_UINT64 0x00000080`

4.24.1 Detailed Description

4.24.2 Macro Definition Documentation

4.24.2.1 `#define DATARAWTYPE_INT16 0x00000002`

Definition at line 424 of file libtiepie.h.

4.24.2.2 `#define DATARAWTYPE_INT32 0x00000004`

Definition at line 425 of file libtiepie.h.

4.24.2.3 `#define DATARAWTYPE_INT64 0x00000008`

Definition at line 426 of file libtiepie.h.

4.24.2.4 `#define DATARAWTYPE_INT8 0x00000001`

Definition at line 423 of file libtiepie.h.

4.24.2.5 `#define DATARAWTYPE_UINT16 0x00000020`

Definition at line 429 of file libtiepie.h.

4.24.2.6 `#define DATARAWTYPE_UINT32 0x00000040`

Definition at line 430 of file libtiepie.h.

4.24.2.7 `#define DATARAWTYPE_UINT64 0x00000080`

Definition at line 431 of file libtiepie.h.

4.24.2.8 `#define DATARAWTYPE_UINT8 0x00000010`

Definition at line 428 of file libtiepie.h.

4.24.2.9 `#define DATARAWTYPE_UNKNOWN 0x00000000`

Definition at line 421 of file libtiepie.h.

4.25 DeviceID masks

Macros

- `#define IDM_DEVICEID 0x80000000`
- `#define IDM_ALL 0xffffffff`

4.25.1 Detailed Description

4.25.2 Macro Definition Documentation

4.25.2.1 `#define IDM_ALL 0xffffffff`

Definition at line 440 of file libtiepie.h.

4.25.2.2 `#define IDM_DEVICEID 0x80000000`

Definition at line 439 of file libtiepie.h.

4.26 DeviceID bits

Macros

- `#define IDB_HS3 0`
- `#define IDB_HS4 1`
- `#define IDB_HS4D 2`
- `#define IDB_HS805 3`
- `#define IDB_HP3 4`
- `#define IDB_HS5 5`
- `#define IDB_HL0516 6`
- `#define IDB_PA1 7`

4.26.1 Detailed Description

4.26.2 Macro Definition Documentation

4.26.2.1 `#define IDB_HL0516 6`

Definition at line 454 of file libtiepie.h.

4.26.2.2 `#define IDB_HP3 4`

Definition at line 452 of file libtiepie.h.

4.26.2.3 `#define IDB_HS3 0`

Definition at line 448 of file libtiepie.h.

4.26.2.4 `#define IDB_HS4 1`

Definition at line 449 of file libtiepie.h.

4.26.2.5 `#define IDB_HS4D 2`

Definition at line 450 of file libtiepie.h.

4.26.2.6 `#define IDB_HS5 5`

Definition at line 453 of file libtiepie.h.

4.26.2.7 `#define IDB_HS805 3`

Definition at line 451 of file libtiepie.h.

4.26.2.8 `#define IDB_PA1 7`

Definition at line 455 of file libtiepie.h.

4.27 DeviceID's

Macros

- `#define ID_HS3 ( IDM_DEVICEID | ( 1 << IDB_HS3 ) )`
Handyscope HS3.
- `#define ID_HS4 ( IDM_DEVICEID | ( 1 << IDB_HS4 ) )`
Handyscope HS4.
- `#define ID_HS4D ( IDM_DEVICEID | ( 1 << IDB_HS4D ) )`
Handyscope HS4 DIFF.
- `#define ID_HS805 ( IDM_DEVICEID | ( 1 << IDB_HS805 ) )`
TiePieSCOPE HS805.
- `#define ID_HP3 ( IDM_DEVICEID | ( 1 << IDB_HP3 ) )`
Handyprobe HP3.
- `#define ID_HS5 ( IDM_DEVICEID | ( 1 << IDB_HS5 ) )`
Handyscope HS5.
- `#define ID_HL0516 ( IDM_DEVICEID | ( 1 << IDB_HL0516 ) )`
HL0516.
- `#define ID_PA1 ( IDM_DEVICEID | ( 1 << IDB_PA1 ) )`
Power Analyzer PA1.

4.27.1 Detailed Description

4.27.2 Macro Definition Documentation

4.27.2.1 `#define ID_HL0516 ( IDM_DEVICEID | ( 1 << IDB_HL0516 ) )`

HL0516.

Definition at line 469 of file libtiepie.h.

4.27.2.2 `#define ID_HP3 ( IDM_DEVICEID | ( 1 << IDB_HP3 ) )`

Handyprobe HP3.

Definition at line 467 of file libtiepie.h.

4.27.2.3 `#define ID_HS3 ( IDM_DEVICEID | ( 1 << IDB_HS3 ) )`

Handyscope HS3.

Definition at line 463 of file libtiepie.h.

4.27.2.4 `#define ID_HS4 ( IDM_DEVICEID | ( 1 << IDB_HS4 ) )`

Handyscope HS4.

Definition at line 464 of file libtiepie.h.

4.27.2.5 `#define ID_HS4D ( IDM_DEVICEID | ( 1 << IDB_HS4D ) )`

Handyscope HS4 DIFF.

Definition at line 465 of file libtiepie.h.

4.27.2.6 `#define ID_HS5 ( IDM_DEVICEID | ( 1 << IDB_HS5 ) )`

Handyscope HS5.

Definition at line 468 of file libtiepie.h.

4.27.2.7 `#define ID_HS805 ( IDM_DEVICEID | ( 1 << IDB_HS805 ) )`

TiePieSCOPE HS805.

Definition at line 466 of file libtiepie.h.

4.27.2.8 `#define ID_PA1 ( IDM_DEVICEID | ( 1 << IDB_PA1 ) )`

Power Analyzer PA1.

Definition at line 470 of file libtiepie.h.

4.28 Types

Typedefs

- typedef int32_t [LibTiePieStatus_t](#)
libTiePie status codes
- typedef uint32_t [TpDeviceHandle_t](#)
device handle
- typedef uint64_t [TpVersion_t](#)
- typedef uint32_t [TpDate_t](#)
- typedef uint8_t [bool8_t](#)
boolean one byte wide

4.28.1 Detailed Description

4.28.2 Typedef Documentation

4.28.2.1 typedef uint8_t bool8_t

boolean one byte wide

Definition at line 483 of file libtiepie.h.

4.28.2.2 typedef int32_t LibTiePieStatus_t

libTiePie status codes

See also

[Status return codes](#)

Definition at line 479 of file libtiepie.h.

4.28.2.3 typedef uint32_t TpDate_t

See also

[DevGetCalibrationDate\(\)](#)

Definition at line 482 of file libtiepie.h.

4.28.2.4 typedef uint32_t TpDeviceHandle_t

device handle

Definition at line 480 of file libtiepie.h.

4.28.2.5 typedef uint64_t TpVersion_t

See also

[LibGetVersion\(\)](#)

[DevGetDriverVersion\(\)](#)

[DevGetFirmwareVersion\(\)](#)

Definition at line 481 of file libtiepie.h.

4.29 Macro's

Macros

- `#define TPVERSION_MAJOR(x) ( x >> 48 )`
- `#define TPVERSION_MINOR(x) ( ( x >> 32 ) & 0xffff )`
- `#define TPVERSION_RELEASE(x) ( ( x >> 16 ) & 0xffff )`
- `#define TPVERSION_BUILD(x) ( x & 0xffff )`
- `#define TPDATE_YEAR(x) ( x >> 16 )`
- `#define TPDATE_MONTH(x) ( ( x >> 8 ) & 0xff )`
- `#define TPDATE_DAY(x) ( x & 0xff )`

4.29.1 Detailed Description

4.29.2 Macro Definition Documentation

4.29.2.1 `#define TPDATE_DAY( x ) ( x & 0xff )`

Definition at line 498 of file libtiepie.h.

4.29.2.2 `#define TPDATE_MONTH( x ) ( ( x >> 8 ) & 0xff )`

Definition at line 497 of file libtiepie.h.

4.29.2.3 `#define TPDATE_YEAR( x ) ( x >> 16 )`

Definition at line 496 of file libtiepie.h.

4.29.2.4 `#define TPVERSION_BUILD( x ) ( x & 0xffff )`

Definition at line 494 of file libtiepie.h.

4.29.2.5 `#define TPVERSION_MAJOR( x ) ( x >> 48 )`

Definition at line 491 of file libtiepie.h.

4.29.2.6 `#define TPVERSION_MINOR( x ) ( ( x >> 32 ) & 0xffff )`

Definition at line 492 of file libtiepie.h.

4.29.2.7 `#define TPVERSION_RELEASE( x ) ( ( x >> 16 ) & 0xffff )`

Definition at line 493 of file libtiepie.h.

4.30 Messages

Macros

- `#define WM_LIBTIEPIE ( WM_USER + 1337 )`
- `#define WM_LIBTIEPIE_LST_DEVICEADDED ( WM_LIBTIEPIE + 2 )`
- `#define WM_LIBTIEPIE_LST_DEVICEREMOVED ( WM_LIBTIEPIE + 3 )`
- `#define WM_LIBTIEPIE_DEV_REMOVED ( WM_LIBTIEPIE + 4 )`
- `#define WM_LIBTIEPIE_SCP_DATAREADY ( WM_LIBTIEPIE + 0 )`
- `#define WM_LIBTIEPIE_SCP_DATAOVERFLOW ( WM_LIBTIEPIE + 1 )`
- `#define WM_LIBTIEPIE_GEN_BURSTCOMPLETED ( WM_LIBTIEPIE + 5 )`
- `#define WM_LIBTIEPIE_GEN_CONTROLLABLECHANGED ( WM_LIBTIEPIE + 6 )`

4.30.1 Detailed Description

4.30.2 Macro Definition Documentation

4.30.2.1 `#define WM_LIBTIEPIE ( WM_USER + 1337 )`

Definition at line 511 of file libtiepie.h.

4.30.2.2 `#define WM_LIBTIEPIE_DEV_REMOVED ( WM_LIBTIEPIE + 4 )`

Definition at line 516 of file libtiepie.h.

4.30.2.3 `#define WM_LIBTIEPIE_GEN_BURSTCOMPLETED ( WM_LIBTIEPIE + 5 )`

Definition at line 521 of file libtiepie.h.

4.30.2.4 `#define WM_LIBTIEPIE_GEN_CONTROLLABLECHANGED ( WM_LIBTIEPIE + 6 )`

Definition at line 522 of file libtiepie.h.

4.30.2.5 `#define WM_LIBTIEPIE_LST_DEVICEADDED ( WM_LIBTIEPIE + 2 )`

Definition at line 513 of file libtiepie.h.

4.30.2.6 `#define WM_LIBTIEPIE_LST_DEVICEREMOVED ( WM_LIBTIEPIE + 3 )`

Definition at line 514 of file libtiepie.h.

4.30.2.7 `#define WM_LIBTIEPIE_SCP_DATAOVERFLOW ( WM_LIBTIEPIE + 1 )`

Definition at line 519 of file libtiepie.h.

4.30.2.8 `#define WM_LIBTIEPIE_SCP_DATAREADY ( WM_LIBTIEPIE + 0 )`

Definition at line 518 of file libtiepie.h.

4.31 Functions

Modules

- [Library](#)
- [Device list](#)
- [Device](#)
- [Oscilloscope](#)
- [Generator](#)
- [I2CHost](#)

4.31.1 Detailed Description

4.32 Library

Functions

- [TpVersion_t LibGetVersion](#) ()
- [uint32_t LibGetConfig](#) (uint8_t *pBuffer, uint32_t dwBufferLength)
- [LibTiePieStatus_t LibGetLastStatus](#) ()
- const char * [LibGetLastStatusStr](#) ()

4.32.1 Detailed Description

4.32.2 Function Documentation

4.32.2.1 uint32_t LibGetConfig (uint8_t * pBuffer, uint32_t dwBufferLength)

Get library configuration number.

Example:

```
uint32_t dwLength = LibGetConfig( NULL , 0 );
uint8_t Buffer[ dwLength ];
dwLength = LibGetConfig( Buffer , dwLength );

printf( "LibGetConfig = 0x" );
for( uint32_t i = 0 ; i < dwLength ; i++ )
    printf( "%02x" , Buffer[ i ] );
printf( "\n" );
```

Parameters

<i>pBuffer</i>	
<i>dwBufferLength</i>	

Returns

Config number length

4.32.2.2 LibTiePieStatus_t LibGetLastStatus ()

Get current last status value, last status is set after each call to the library.

Example:

```
printf( "LibGetLastStatus = %d\n" , LibGetLastStatus() );
```

See also

[Status return codes](#)
[LibGetLastStatusStr](#)

Returns

Status code

4.32.2.3 `const char* LibGetLastStatusStr ( )`

Get current last status value as text, last status is set after each call to the library.

Example:

```
printf( "LibGetLastStatusStr = %s\n" , LibGetLastStatusStr  
      () );
```

See also

[LibGetLastStatus](#)

Returns

Status code as text

4.32.2.4 `TpVersion_t LibGetVersion ( )`

Get library version number.

Example:

```
TpVersion_t Version = LibGetVersion();  
uint16_t wMajor      = TPVERSION_MAJOR( Version );  
uint16_t wMinor      = TPVERSION_MINOR( Version );  
uint16_t wRelease     = TPVERSION_RELEASE( Version );  
uint16_t wBuild       = TPVERSION_BUILD( Version );  
  
printf( "LibGetVersion = %u.%u.%u.%u\n" , wMajor , wMinor , wRelease ,  
      wBuild );
```

Returns

Library version number

4.33 Device list

Modules

- [Events](#)

Functions

- `uint32_t LstGetCount` (`uint32_t dwDeviceType`)
- `bool8_t LstGetDeviceCanOpen` (`uint32_t dwDeviceType, uint32_t dwId`)
- `uint32_t LstGetDeviceProductId` (`uint32_t dwDeviceType, uint32_t dwId`)
- `uint32_t LstGetDeviceVendorId` (`uint32_t dwDeviceType, uint32_t dwId`)
- `uint32_t LstGetDeviceName` (`uint32_t dwDeviceType, uint32_t dwId, char *pBuffer, uint32_t dwBufferLength`)
- `uint32_t LstGetDeviceNameShort` (`uint32_t dwDeviceType, uint32_t dwId, char *pBuffer, uint32_t dwBufferLength`)
- `uint32_t LstGetDeviceSerialNumber` (`uint32_t dwDeviceType, uint32_t dwId`)
- `TpDeviceHandle_t LstOpenDevice` (`uint32_t dwDeviceType, uint32_t dwId`)
- `void LstRemoveDevice` (`uint32_t dwSerialNumber`)
- `void LstUpdate` (`uint32_t dwDeviceIdMask`)

4.33.1 Detailed Description

4.33.2 Function Documentation

4.33.2.1 `uint32_t LstGetCount ( uint32_t dwDeviceType )`

Get number of devices available in list

Parameters

<i>dwDeviceType</i>	a device type
---------------------	-------------------------------

Returns

number of devices available in list

4.33.2.2 `bool8_t LstGetDeviceCanOpen ( uint32_t dwDeviceType, uint32_t dwId )`

Check whether the instrument can be opened.

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number

Returns

1 if true or 0 if false

4.33.2.3 `uint32_t LstGetDeviceName ( uint32_t dwDeviceType, uint32_t dwId, char * pBuffer, uint32_t dwBufferLength )`

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number
<i>pBuffer</i>	
<i>dwBufferLength</i>	

Returns

4.33.2.4 `uint32_t LstGetDeviceNameShort ( uint32_t dwDeviceType, uint32_t dwId, char * pBuffer, uint32_t dwBufferLength )`

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number
<i>pBuffer</i>	
<i>dwBufferLength</i>	

Returns

4.33.2.5 `uint32_t LstGetDeviceProductId ( uint32_t dwDeviceType, uint32_t dwId )`

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number

Returns

4.33.2.6 `uint32_t LstGetDeviceSerialNumber ( uint32_t dwDeviceType, uint32_t dwId )`

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number

Returns

4.33.2.7 `uint32_t LstGetDeviceVendorId ( uint32_t dwDeviceType, uint32_t dwId )`

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number

Returns

4.33.2.8 `TPDeviceHandle_t LstOpenDevice ( uint32_t dwDeviceType, uint32_t dwId )`

Get handle to device, for each device the handle is only assigned once.

Parameters

<i>dwDeviceType</i>	a device type
<i>dwId</i>	device index or a Device ID or a serial number

Returns

a device handle, or [TPDEVICEHANDLE_INVALID](#) on error

4.33.2.9 `void LstRemoveDevice ( uint32_t dwSerialNumber )`

Remove a instrument from the device lists so it can be used by other applications.

Parameters

<i>dwSerialNumber</i>	instruments serial number
-----------------------	---------------------------

4.33.2.10 `void LstUpdate ( uint32_t dwDeviceIdMask )`

Search for new instruments.

Example 1:

```
LstUpdate( IDM_ALL ); // Search for all instruments
```

Example 2:

```
LstUpdate( ID_HS5 ); // Search for Handyscope HS5's
```

Parameters

<i>dwDeviceIdMask</i>	a ORed mask of Device ID's
-----------------------	--

4.34 Events

Functions

- void [LstSetCallbackDeviceAdded](#) (uint32_t dwDeviceType, [TpCallback_t](#) pCallback, void *pData)
- void [LstSetCallbackDeviceRemoved](#) (uint32_t dwDeviceType, [TpCallback_t](#) pCallback, void *pData)
- void [LstSetEventDeviceAdded](#) (uint32_t dwDeviceType, HANDLE hEvent)
- void [LstSetEventDeviceRemoved](#) (uint32_t dwDeviceType, HANDLE hEvent)
- void [LstSetMessageDeviceAdded](#) (uint32_t dwDeviceType, HWND hWnd)
- void [LstSetMessageDeviceRemoved](#) (uint32_t dwDeviceType, HWND hWnd)

4.34.1 Detailed Description

4.34.2 Function Documentation

4.34.2.1 void [LstSetCallbackDeviceAdded](#) (uint32_t *dwDeviceType*, [TpCallback_t](#) *pCallback*, void * *pData*)

Parameters

<i>dwDeviceType</i>	
<i>pCallback</i>	
<i>pData</i>	

4.34.2.2 void [LstSetCallbackDeviceRemoved](#) (uint32_t *dwDeviceType*, [TpCallback_t](#) *pCallback*, void * *pData*)

Parameters

<i>dwDeviceType</i>	
<i>pCallback</i>	
<i>pData</i>	

4.34.2.3 void [LstSetEventDeviceAdded](#) (uint32_t *dwDeviceType*, HANDLE *hEvent*)

Parameters

<i>dwDeviceType</i>	
<i>hEvent</i>	

4.34.2.4 void [LstSetEventDeviceRemoved](#) (uint32_t *dwDeviceType*, HANDLE *hEvent*)

Parameters

<i>dwDeviceType</i>	
<i>hEvent</i>	

4.34.2.5 void [LstSetMessageDeviceAdded](#) (uint32_t *dwDeviceType*, HWND *hWnd*)

Parameters

<i>dwDeviceType</i>	
<i>hWnd</i>	

4.34.2.6 void LstSetMessageDeviceRemoved (uint32_t *dwDeviceType*, HWND *hWnd*)

Parameters

<i>dwDeviceType</i>	
<i>hWnd</i>	

4.35 Device

Modules

- [Events](#)

Functions

- void [DevClose](#) (TpDeviceHandle_t hDevice)
- bool8_t [DevIsRemoved](#) (TpDeviceHandle_t hDevice)
- TpVersion_t [DevGetDriverVersion](#) (TpDeviceHandle_t hDevice)
- TpVersion_t [DevGetFirmwareVersion](#) (TpDeviceHandle_t hDevice)
- TpDate_t [DevGetCalibrationDate](#) (TpDeviceHandle_t hDevice)
- uint32_t [DevGetSerialNumber](#) (TpDeviceHandle_t hDevice)
- uint32_t [DevGetProductId](#) (TpDeviceHandle_t hDevice)
- uint32_t [DevGetVendorId](#) (TpDeviceHandle_t hDevice)
- uint32_t [DevGetName](#) (TpDeviceHandle_t hDevice, char *pBuffer, uint32_t dwBufferLength)
- uint32_t [DevGetNameShort](#) (TpDeviceHandle_t hDevice, char *pBuffer, uint32_t dwBufferLength)

4.35.1 Detailed Description

4.35.2 Function Documentation

4.35.2.1 void DevClose (TpDeviceHandle_t hDevice)

Close a device handle.

Parameters

<i>hDevice</i>	the handle to close
----------------	---------------------

4.35.2.2 TpDate_t DevGetCalibrationDate (TpDeviceHandle_t hDevice)

Get the calibration date of the instrument.

Example:

```
TpDate_t Date      = DevGetCalibrationDate (
    hDevice );
uint16_t wYear     = TPDATE_YEAR( Date );
uint8_t  byMonth   = TPDATE_MONTH( Date );
uint8_t  byDay     = TPDATE_DAY( Date );

printf( "DevGetCalibrationDate = %u-%u-%u\n" , wYear , byMonth , byDay );
```

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Instruments calibration date

4.35.2.3 TpVersion_t DevGetDriverVersion (TpDeviceHandle_t hDevice)

Get version number of the driver.

Example:

```
TpVersion_t Version = DevGetDriverVersion(
    hDevice );
uint16_t wMajor      = TPVERSION_MAJOR( Version );
uint16_t wMinor      = TPVERSION_MINOR( Version );
uint16_t wRelease     = TPVERSION_RELEASE( Version );
uint16_t wBuild       = TPVERSION_BUILD( Version );

printf( "DevGetDriverVersion = %u.%u.%u.%u\n" , wMajor , wMinor , wRelease ,
    wBuild );
```

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Library version number

4.35.2.4 TpVersion_t DevGetFirmwareVersion (TpDeviceHandle_t hDevice)

Get version number of the firmware.

Example:

```
TpVersion_t Version = DevGetFirmwareVersion(
    hDevice );
uint16_t wMajor      = TPVERSION_MAJOR( Version );
uint16_t wMinor      = TPVERSION_MINOR( Version );
uint16_t wRelease     = TPVERSION_RELEASE( Version );
uint16_t wBuild       = TPVERSION_BUILD( Version );

printf( "DevGetFirmwareVersion = %u.%u.%u.%u\n" , wMajor , wMinor , wRelease
    , wBuild );
```

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Firmware version number

4.35.2.5 uint32_t DevGetName (TpDeviceHandle_t hDevice, char * pBuffer, uint32_t dwBufferLength)

Get the instruments name.

Example:

```
uint32_t dwLength = DevGetName( hDevice , NULL , 0 ) + 1; // Add
    one for the terminating zero
char sName[ dwLength ];
```

```
dwLength = DevGetName( hDevice , sName , dwLength );
printf( "DevGetName = %s\n" , Name );
```

See also

[DevGetNameShort](#)

Parameters

<i>hDevice</i>	a device handle
<i>pBuffer</i>	pointer to buffer to write to
<i>dwBufferLength</i>	length of the buffer

Returns

Length of the short name excluding excluding terminating zero.

4.35.2.6 uint32_t DevGetNameShort (TpDeviceHandle_t hDevice, char * pBuffer, uint32_t dwBufferLength)

Get the instruments short name.

Example:

```
uint32_t dwLength = DevGetNameShort( hDevice , NULL , 0 ) + 1
; // Add one for the terminating zero
char sNameShort[ dwLength ];
dwLength = DevGetNameShort( hDevice , sNameShort , dwLength )
;

printf( "DevGetNameShort = %s\n" , Name );
```

See also

[DevGetName](#)

Parameters

<i>hDevice</i>	a device handle
<i>pBuffer</i>	pointer to buffer to write to
<i>dwBufferLength</i>	length of the buffer

Returns

Length of the short name excluding excluding terminating zero.

4.35.2.7 uint32_t DevGetProductId (TpDeviceHandle_t hDevice)

Get the instruments Product ID.

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Instruments Product ID

4.35.2.8 uint32_t DevGetSerialNumber (TpDeviceHandle_t hDevice)

Get the instruments serial number.

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Instruments serial number

4.35.2.9 uint32_t DevGetVendorId (TpDeviceHandle_t hDevice)

Get the instruments Vendor ID.

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Instruments Vendor ID

4.35.2.10 bool8_t DevIsRemoved (TpDeviceHandle_t hDevice)

Check whether an instrument is removed.

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if removed else 0

4.36 Events

Functions

- void [DevSetCallbackRemoved](#) (TpDeviceHandle_t hDevice, TpCallback_t pCallback, void *pData)
- void [DevSetEventRemoved](#) (TpDeviceHandle_t hDevice, HANDLE hEvent)
- void [DevSetMessageRemoved](#) (TpDeviceHandle_t hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)

4.36.1 Detailed Description

4.36.2 Function Documentation

4.36.2.1 void DevSetCallbackRemoved (TpDeviceHandle_t hDevice, TpCallback_t pCallback, void * pData)

Set callback function which is called when the instrument is removed.

Parameters

<i>hDevice</i>	a device handle
<i>pCallback</i>	pointer to callback function
<i>pData</i>	optional user data

4.36.2.2 void DevSetEventRemoved (TpDeviceHandle_t hDevice, HANDLE hEvent)

Parameters

<i>hDevice</i>	
<i>hEvent</i>	

4.36.2.3 void DevSetMessageRemoved (TpDeviceHandle_t hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)

Parameters

<i>hDevice</i>	
<i>hWnd</i>	
<i>wParam</i>	
<i>lParam</i>	

4.37 Oscilloscope

Modules

- [Input channels](#)
- [Data](#)
- [Events](#)
- [Measurements](#)
- [Resolution](#)
- [Timebase](#)
- [Trigger system](#)

4.37.1 Detailed Description

4.38 Input channels

Functions

- `uint16_t ScpGetChannelCount (TpDeviceHandle_t hDevice)`
- `uint32_t ScpGetSharedChannelGroupCount (TpDeviceHandle_t hDevice)`
- `uint32_t ScpGetSharedChannelGroup (TpDeviceHandle_t hDevice, uint32_t dwGroupIndex, uint16_t *pChannelNumbers, uint32_t dwLength)`
- `bool8_t ScpChGetAutoRanging (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `bool8_t ScpChSetAutoRanging (TpDeviceHandle_t hDevice, uint16_t wCh, bool8_t bEnable)`
- `uint32_t ScpChGetConnectorType (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `uint64_t ScpChGetCouplings (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `uint64_t ScpChGetCoupling (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `uint64_t ScpChSetCoupling (TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwCoupling)`
- `bool8_t ScpChGetEnabled (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `bool8_t ScpChSetEnabled (TpDeviceHandle_t hDevice, uint16_t wCh, bool8_t bEnable)`
- `bool8_t ScpChIsDifferential (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `bool8_t ScpChIsDualRateCapable (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `double ScpChGetProbeGain (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `double ScpChSetProbeGain (TpDeviceHandle_t hDevice, uint16_t wCh, double dProbeGain)`
- `uint32_t ScpChGetRanges (TpDeviceHandle_t hDevice, uint16_t wCh, double *pList, uint32_t dwLength)`
- `uint32_t ScpChGetRangesEx (TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwCoupling, double *pList, uint32_t dwLength)`
- `double ScpChGetRange (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `double ScpChSetRange (TpDeviceHandle_t hDevice, uint16_t wCh, double dRange)`

4.38.1 Detailed Description

4.38.2 Function Documentation

4.38.2.1 `bool8_t ScpChGetAutoRanging ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Check whether auto ranging is enabled.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . <code>ScpGetChannelCount ()</code> - 1

Returns

1 if enabled, 0 if disabled

4.38.2.2 `uint32_t ScpChGetConnectorType ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Get channel [connector type](#).

See also

[Connector types](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Channel [connector type](#).

4.38.2.3 `uint64_t ScpChGetCoupling ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Get current coupling.

See also

[Coupling types](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1

Returns

Current coupling, a [CK_*](#) value.

4.38.2.4 `uint64_t ScpChGetCouplings ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Get supported couplings.

See also

[Coupling types](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1

Returns

Supported couplings, set of ORed [CK_*](#) values.

4.38.2.5 `bool8_t ScpChGetEnabled ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Check whether channel is enabled.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1

Returns

1 if enabled, 0 if disabled.

4.38.2.6 double ScpChGetProbeGain (TpDeviceHandle_t hDevice, uint16_t wCh)

Get channel probe gain.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0.. ScpGetChannelCount() - 1

Returns

Current probe gain.

4.38.2.7 double ScpChGetRange (TpDeviceHandle_t hDevice, uint16_t wCh)

Get current range.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0.. ScpGetChannelCount() - 1

Returns

Current range.

4.38.2.8 uint32_t ScpChGetRanges (TpDeviceHandle_t hDevice, uint16_t wCh, double * pList, uint32_t dwLength)

Get supported ranges for current coupling.

Example:

```
uint32_t dwRangeCount = ScpChGetRanges( hDevice , wCh , NULL ,
0 );
double Ranges[ dwRangeCount ];
dwRangeCount = ScpChGetRanges( hDevice , wCh , Ranges ,
dwRangeCount );

printf( "ScpChGetRanges (Ch%u):\n" , wCh + 1 );
for( uint32_t i = 0 ; i < dwRangeCount ; i++ )
    printf( "- %f\n" , Ranges[ i ] );
```

See also

[ScpChGetRangesEx](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0.. ScpGetChannelCount() - 1
<i>pList</i>	pointer to array
<i>dwLength</i>	number of elements in array

Returns

Total number of ranges.

4.38.2.9 `uint32_t ScpChGetRangesEx ( TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwCoupling, double * pList, uint32_t dwLength )`

Get supported ranges by coupling.

See also

[ScpChGetRanges](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>qwCoupling</i>	coupling: a CK_* value
<i>pList</i>	pointer to array
<i>dwLength</i>	number of elements in array

Returns

total number of ranges

4.38.2.10 `bool8_t ScpChIsDifferential ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Check whether the channel has a differential input.

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if differential, else 0.

4.38.2.11 `bool8_t ScpChIsDualRateCapable ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.38.2.12 `bool8_t ScpChSetAutoRanging ( TpDeviceHandle_t hDevice, uint16_t wCh, bool8_t bEnable )`

Set auto ranging.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>bEnable</i>	?

Returns

1 if enabled, 0 if disabled

4.38.2.13 `uint64_t ScpChSetCoupling ( TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwCoupling )`

Set coupling.

See also

[Coupling types](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>qwCoupling</i>	coupling: a CK_* value

Returns

Current coupling, a [CK_*](#) value.

4.38.2.14 `bool8_t ScpChSetEnabled ( TpDeviceHandle_t hDevice, uint16_t wCh, bool8_t bEnable )`

Set channel enable.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>bEnable</i>	?

Returns

1 if enabled, 0 if disabled.

4.38.2.15 `double ScpChSetProbeGain ( TpDeviceHandle_t hDevice, uint16_t wCh, double dProbeGain )`

Set channel probe gain.

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>dProbeGain</i>	probe gain

Returns

Current probe gain.

4.38.2.16 `double ScpChSetRange ( TpDeviceHandle_t hDevice, uint16_t wCh, double dRange )`

Set range.

Example:

```
double dRange = 10;

dRange = ScpChSetRange( hDevice , wCh , dRange );

printf( "ScpChSetRange = %f" , dRange );
```

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0.. ScpGetChannelCount() - 1
<i>dRange</i>	maximum values that must fit within range

Returns

current range

4.38.2.17 `uint16_t ScpGetChannelCount ( TpDeviceHandle_t hDevice )`

Get total number of channels supported by the oscilloscope

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

number of channels

4.38.2.18 `uint32_t ScpGetSharedChannelGroup ( TpDeviceHandle_t hDevice, uint32_t dwGroupIndex, uint16_t * pChannelNumbers, uint32_t dwLength )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.38.2.19 `uint32_t ScpGetSharedChannelGroupCount ( TpDeviceHandle_t hDevice )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39 Data

Functions

- `uint64_t ScpGetData (TpDeviceHandle_t hDevice, float **pBuffers, uint16_t wChannelCount, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetData1Ch (TpDeviceHandle_t hDevice, float *pBufferCh1, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetData2Ch (TpDeviceHandle_t hDevice, float *pBufferCh1, float *pBufferCh2, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetData3Ch (TpDeviceHandle_t hDevice, float *pBufferCh1, float *pBufferCh2, float *pBufferCh3, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetData4Ch (TpDeviceHandle_t hDevice, float *pBufferCh1, float *pBufferCh2, float *pBufferCh3, float *pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetDataRaw (TpDeviceHandle_t hDevice, void **pBuffers, uint16_t wChannelCount, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetDataRaw1Ch (TpDeviceHandle_t hDevice, void *pBufferCh1, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetDataRaw2Ch (TpDeviceHandle_t hDevice, void *pBufferCh1, void *pBufferCh2, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetDataRaw3Ch (TpDeviceHandle_t hDevice, void *pBufferCh1, void *pBufferCh2, void *pBufferCh3, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `uint64_t ScpGetDataRaw4Ch (TpDeviceHandle_t hDevice, void *pBufferCh1, void *pBufferCh2, void *pBufferCh3, void *pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount)`
- `bool8_t ScpChIsRangeMaxReachable (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `uint32_t ScpChGetDataRawType (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `uint64_t ScpGetValidPreSampleCount (TpDeviceHandle_t hDevice)`
- `void ScpChGetDataValueRange (TpDeviceHandle_t hDevice, uint16_t wCh, double *pMin, double *pMax)`
- `double ScpChGetDataValueMax (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `double ScpChGetDataValueMin (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `void ScpChGetDataRawValueRange (TpDeviceHandle_t hDevice, uint16_t wCh, int64_t *pMin, int64_t *pZero, int64_t *pMax)`
- `int64_t ScpChGetDataRawValueMax (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `int64_t ScpChGetDataRawValueZero (TpDeviceHandle_t hDevice, uint16_t wCh)`
- `int64_t ScpChGetDataRawValueMin (TpDeviceHandle_t hDevice, uint16_t wCh)`

4.39.1 Detailed Description

4.39.2 Function Documentation

4.39.2.1 `uint32_t ScpChGetDataRawType ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . <code>ScpGetChannelCount ()</code> - 1

4.39.2.2 `int64_t ScpChGetDataRawValueMax ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39.2.3 `int64_t ScpChGetDataRawValueMin ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39.2.4 `void ScpChGetDataRawValueRange ( TpDeviceHandle_t hDevice, uint16_t wCh, int64_t * pMin, int64_t * pZero, int64_t * pMax )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39.2.5 `int64_t ScpChGetDataRawValueZero ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39.2.6 `double ScpChGetDataValueMax ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Get possible maximum value for current available data

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1

Returns

possible maximum value

4.39.2.7 `double ScpChGetDataValueMin ( TpDeviceHandle_t hDevice, uint16_t wCh )`

Get possible minimum value for current available data

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1

Returns

possible minimum value

4.39.2.8 `void ScpChGetDataValueRange ( TpDeviceHandle_t hDevice, uint16_t wCh, double * pMin, double * pMax )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.39.2.9 **bool8_t** ScpChlsRangeMaxReachable (**TpDeviceHandle_t** *hDevice*, **uint16_t** *wCh*)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount() - 1

4.39.2.10 **uint64_t** ScpGetData (**TpDeviceHandle_t** *hDevice*, **float **** *pBuffers*, **uint16_t** *wChannelCount*, **uint64_t** *qwStartIndex*, **uint64_t** *qwSampleCount*)

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.11 **uint64_t** ScpGetData1Ch (**TpDeviceHandle_t** *hDevice*, **float *** *pBufferCh1*, **uint64_t** *qwStartIndex*, **uint64_t** *qwSampleCount*)

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.12 **uint64_t** ScpGetData2Ch (**TpDeviceHandle_t** *hDevice*, **float *** *pBufferCh1*, **float *** *pBufferCh2*, **uint64_t** *qwStartIndex*, **uint64_t** *qwSampleCount*)

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.13 **uint64_t** ScpGetData3Ch (**TpDeviceHandle_t** *hDevice*, **float *** *pBufferCh1*, **float *** *pBufferCh2*, **float *** *pBufferCh3*, **uint64_t** *qwStartIndex*, **uint64_t** *qwSampleCount*)

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.14 `uint64_t ScpGetData4Ch ( TpDeviceHandle_t hDevice, float * pBufferCh1, float * pBufferCh2, float * pBufferCh3, float * pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.15 `uint64_t ScpGetDataRaw ( TpDeviceHandle_t hDevice, void ** pBuffers, uint16_t wChannelCount, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.16 `uint64_t ScpGetDataRaw1Ch ( TpDeviceHandle_t hDevice, void * pBufferCh1, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.17 `uint64_t ScpGetDataRaw2Ch ( TpDeviceHandle_t hDevice, void * pBufferCh1, void * pBufferCh2, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.18 `uint64_t ScpGetDataRaw3Ch ( TpDeviceHandle_t hDevice, void * pBufferCh1, void * pBufferCh2, void * pBufferCh3, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.19 `uint64_t ScpGetDataRaw4Ch ( TpDeviceHandle_t hDevice, void * pBufferCh1, void * pBufferCh2, void * pBufferCh3, void * pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount )`

...

Parameters

<i>hDevice</i>	a device handle
<i>qwSampleCount</i>	number of samples to write into buffer

Returns

number of samples ready

4.39.2.20 `uint64_t ScpGetValidPreSampleCount ( TpDeviceHandle_t hDevice )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40 Events

Functions

- void [ScpSetCallbackDataReady](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, void *pData)
- void [ScpSetCallbackDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, void *pData)
- void [ScpSetEventDataReady](#) ([TpDeviceHandle_t](#) hDevice, HANDLE hEvent)
- void [ScpSetEventDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, HANDLE hEvent)
- void [ScpSetMessageDataReady](#) ([TpDeviceHandle_t](#) hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)
- void [ScpSetMessageDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)

4.40.1 Detailed Description

4.40.2 Function Documentation

4.40.2.1 void [ScpSetCallbackDataOverflow](#) ([TpDeviceHandle_t](#) *hDevice*, [TpCallback_t](#) *pCallback*, void * *pData*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40.2.2 void [ScpSetCallbackDataReady](#) ([TpDeviceHandle_t](#) *hDevice*, [TpCallback_t](#) *pCallback*, void * *pData*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40.2.3 void [ScpSetEventDataOverflow](#) ([TpDeviceHandle_t](#) *hDevice*, HANDLE *hEvent*)

Note

Windows only

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40.2.4 void [ScpSetEventDataReady](#) ([TpDeviceHandle_t](#) *hDevice*, HANDLE *hEvent*)

Note

Windows only

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40.2.5 void [ScpSetMessageDataOverflow](#) ([TpDeviceHandle_t](#) *hDevice*, HWND *hWnd*, WPARAM *wParam*, LPARAM *lParam*)

Note

Windows only

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.40.2.6 void ScpSetMessageDataReady (TpDeviceHandle_t *hDevice*, HWND *hWnd*, WPARAM *wParam*, LPARAM *lParam*)

Note

Windows only

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.41 Measurements

Functions

- [bool8_t ScpStart](#) ([TpDeviceHandle_t](#) hDevice)
- [void ScpStop](#) ([TpDeviceHandle_t](#) hDevice)
- [void ScpForceTrigger](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScplsDataReady](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScplsDataOverflow](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScplsRunning](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScplsTriggered](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpGetMeasureModes](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpGetMeasureMode](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpSetMeasureMode](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMeasureMode)

4.41.1 Detailed Description

4.41.2 Function Documentation

4.41.2.1 [void ScpForceTrigger](#) ([TpDeviceHandle_t](#) *hDevice*)

Force trigger

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.41.2.2 [uint32_t ScpGetMeasureMode](#) ([TpDeviceHandle_t](#) *hDevice*)

Get oscilloscope measure mode

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

measure mode: a [CM_*](#) value

4.41.2.3 [uint32_t ScpGetMeasureModes](#) ([TpDeviceHandle_t](#) *hDevice*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.41.2.4 [bool8_t ScplsDataOverflow](#) ([TpDeviceHandle_t](#) *hDevice*)

Check whether overflow occurred

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if overflow occurred, 0 otherwise

4.41.2.5 **bool8_t** ScplsDataReady (**TpDeviceHandle_t** *hDevice*)

Check whether data is ready

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if measuring done, 0 otherwise

4.41.2.6 **bool8_t** ScplsRunning (**TpDeviceHandle_t** *hDevice*)

Check whether instrument is measuring

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if instrument is measuring, 0 otherwise

4.41.2.7 **bool8_t** ScplsTriggered (**TpDeviceHandle_t** *hDevice*)

Check is instrument has triggered

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if oscilloscope has triggerd, 0 otherwise

4.41.2.8 **uint32_t** ScpSetMeasureMode (**TpDeviceHandle_t** *hDevice*, **uint32_t** *dwMeasureMode*)

Set oscilloscope measure mode

Note

Only **MM_BLOCK** is supported

Parameters

<i>hDevice</i>	a device handle
<i>dwMeasure-Mode</i>	measure mode, a CM_* value

Returns

current measure mode, a [CM_*](#) value

4.41.2.9 bool8_t ScpStart (TpDeviceHandle_t *hDevice*)

Start measuring

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

see ScplsRunning

4.41.2.10 void ScpStop (TpDeviceHandle_t *hDevice*)

Stop measuring

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.42 Resolution

Functions

- `uint32_t ScpGetResolutions (TpDeviceHandle_t hDevice, uint8_t *pList, uint32_t dwLength)`
- `uint8_t ScpGetResolution (TpDeviceHandle_t hDevice)`
- `uint8_t ScpSetResolution (TpDeviceHandle_t hDevice, uint8_t byResolution)`
- `bool8_t ScplsResolutionEnhanced (TpDeviceHandle_t hDevice)`
- `bool8_t ScplsResolutionEnhancedEx (TpDeviceHandle_t hDevice, uint8_t byResolution)`

4.42.1 Detailed Description

4.42.2 Function Documentation

4.42.2.1 `uint8_t ScpGetResolution ( TpDeviceHandle_t hDevice )`

Get current resolution

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current resolution

4.42.2.2 `uint32_t ScpGetResolutions ( TpDeviceHandle_t hDevice, uint8_t * pList, uint32_t dwLength )`

Get array with supported resolutions

Parameters

<i>hDevice</i>	a device handle
<i>pList</i>	pointer to array
<i>dwLength</i>	number of elements in array

Returns

total number of resolutions

4.42.2.3 `bool8_t ScplsResolutionEnhanced ( TpDeviceHandle_t hDevice )`

Check whether current resolution is enhanced

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if current resolution is enhanced, 0 otherwise

4.42.2.4 `bool8_t ScplsResolutionEnhancedEx ( TpDeviceHandle_t hDevice, uint8_t byResolution )`

Check whether resolution is enhanced

Parameters

<i>hDevice</i>	a device handle
<i>byResolution</i>	resolution in bits

Returns

1 if resolution is enhanced, 0 otherwise

4.42.2.5 `uint8_t ScpSetResolution ( TpDeviceHandle_t hDevice, uint8_t byResolution )`

Set resolution

Parameters

<i>hDevice</i>	a device handle
<i>byResolution</i>	resolution in bits

Returns

current resolution

4.43 Timebase

Functions

- `uint32_t ScpGetClockSource (TpDeviceHandle_t hDevice)`
- `uint32_t ScpSetClockSource (TpDeviceHandle_t hDevice, uint32_t dwClockSource)`
- `double ScpGetPreSampleRatio (TpDeviceHandle_t hDevice)`
- `double ScpSetPreSampleRatio (TpDeviceHandle_t hDevice, double dPreSampleRatio)`
- `uint64_t ScpGetTriggerHoldOffCountMax (TpDeviceHandle_t hDevice)`
- `uint64_t ScpGetTriggerHoldOffCountMaxEx (TpDeviceHandle_t hDevice, uint32_t dwMeasureMode)`
- `uint64_t ScpGetTriggerHoldOffCount (TpDeviceHandle_t hDevice)`
- `uint64_t ScpSetTriggerHoldOffCount (TpDeviceHandle_t hDevice, uint64_t qwTriggerHoldOffCount)`
- `uint64_t ScpGetRecordLengthMax (TpDeviceHandle_t hDevice)`
- `uint64_t ScpGetRecordLengthMaxEx (TpDeviceHandle_t hDevice, uint32_t dwMeasureMode, uint8_t by-Resolution)`
- `uint64_t ScpGetRecordLength (TpDeviceHandle_t hDevice)`
- `uint64_t ScpSetRecordLength (TpDeviceHandle_t hDevice, uint64_t qwRecordLength)`
- `uint64_t ScpVerifyRecordLength (TpDeviceHandle_t hDevice, uint64_t qwRecordLength)`
- `uint64_t ScpVerifyRecordLengthEx (TpDeviceHandle_t hDevice, uint64_t qwRecordLength, uint32_t dw-MeasureMode, uint8_t byResolution, uint64_t qwActiveChannelMask)`
- `double ScpGetSampleFrequencyMax (TpDeviceHandle_t hDevice)`
- `double ScpGetSampleFrequency (TpDeviceHandle_t hDevice)`
- `double ScpSetSampleFrequency (TpDeviceHandle_t hDevice, double dSampleFrequency)`
- `double ScpVerifySampleFrequency (TpDeviceHandle_t hDevice, double dSampleFrequency)`
- `double ScpVerifySampleFrequencyEx (TpDeviceHandle_t hDevice, double dSampleFrequency, uint32_t dw-MeasureMode, uint8_t byResolution, uint64_t qwActiveChannelMask)`

4.43.1 Detailed Description

4.43.2 Function Documentation

4.43.2.1 `uint32_t ScpGetClockSource ( TpDeviceHandle_t hDevice )`

Get instruments clock source

See also

[Clock modes](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current clock source, a [CM_*](#) value

4.43.2.2 `double ScpGetPreSampleRatio ( TpDeviceHandle_t hDevice )`

Get pre sample ratio

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current pre sample ratio 0 . . 1

4.43.2.3 uint64_t ScpGetRecordLength (TpDeviceHandle_t hDevice)

Get current record length

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current record length

4.43.2.4 uint64_t ScpGetRecordLengthMax (TpDeviceHandle_t hDevice)

Get instruments maximum record length

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

maximum record length

4.43.2.5 uint64_t ScpGetRecordLengthMaxEx (TpDeviceHandle_t hDevice, uint32_t dwMeasureMode, uint8_t byResolution)**Parameters**

<i>hDevice</i>	a device handle
----------------	-----------------

4.43.2.6 double ScpGetSampleFrequency (TpDeviceHandle_t hDevice)

Get current sample frequency

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current sample frequency in Hz

4.43.2.7 double ScpGetSampleFrequencyMax (TpDeviceHandle_t hDevice)

Get instruments maximum sample frequency

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

maximum sample frequency

4.43.2.8 `uint64_t ScpGetTriggerHoldOffCount ( TpDeviceHandle_t hDevice )`

Get ...

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

...

4.43.2.9 `uint64_t ScpGetTriggerHoldOffCountMax ( TpDeviceHandle_t hDevice )`

Get ...

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

...

4.43.2.10 `uint64_t ScpGetTriggerHoldOffCountMaxEx ( TpDeviceHandle_t hDevice, uint32_t dwMeasureMode )`

Get ...

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

...

4.43.2.11 `uint32_t ScpSetClockSource ( TpDeviceHandle_t hDevice, uint32_t dwClockSource )`

Set instruments clock source

Note

Only `CM_INTERNAL` is supported

See also

[Clock modes](#)

Parameters

<i>hDevice</i>	a device handle
<i>dwClockSource</i>	clock source: a CM_* value

Returns

current clock source, a [CM_*](#) value

4.43.2.12 double ScpSetPreSampleRatio (TpDeviceHandle_t hDevice, double dPreSampleRatio)

Set pre sample ratio

Parameters

<i>hDevice</i>	a device handle
<i>dPreSample-Ratio</i>	pre sample ratio 0 . . 1

Returns

current pre sample ratio 0 . . 1

4.43.2.13 uint64_t ScpSetRecordLength (TpDeviceHandle_t hDevice, uint64_t qwRecordLength)

Set record length

Parameters

<i>hDevice</i>	a device handle
<i>qwRecordLength</i>	record length

Returns

current record length

4.43.2.14 double ScpSetSampleFrequency (TpDeviceHandle_t hDevice, double dSampleFrequency)

Set sample frequency

Parameters

<i>hDevice</i>	a device handle
<i>dSample-Frequency</i>	sample frequency in Hz

Returns

current sample frequency in Hz

4.43.2.15 `uint64_t ScpSetTriggerHoldOffCount ( TpDeviceHandle_t hDevice, uint64_t qwTriggerHoldOffCount )`

Set ...

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerHoldOffCount</i>	...

Returns

...

4.43.2.16 `uint64_t ScpVerifyRecordLength ( TpDeviceHandle_t hDevice, uint64_t qwRecordLength )`

Parameters

<i>hDevice</i>	a device handle
<i>qwRecordLength</i>	

Returns

4.43.2.17 `uint64_t ScpVerifyRecordLengthEx ( TpDeviceHandle_t hDevice, uint64_t qwRecordLength, uint32_t dwMeasureMode, uint8_t byResolution, uint64_t qwActiveChannelMask )`

Parameters

<i>hDevice</i>	a device handle
<i>qwRecordLength</i>	
<i>dwMeasureMode</i>	
<i>byResolution</i>	
<i>qwActiveChannelMask</i>	

Returns

4.43.2.18 `double ScpVerifySampleFrequency ( TpDeviceHandle_t hDevice, double dSampleFrequency )`

Verify sample frequency

Parameters

<i>hDevice</i>	a device handle
<i>dSampleFrequency</i>	sample frequency in Hz

Returns

sample frequency in Hz when set

4.43.2.19 `double ScpVerifySampleFrequencyEx ( TpDeviceHandle_t hDevice, double dSampleFrequency, uint32_t dwMeasureMode, uint8_t byResolution, uint64_t qwActiveChannelMask )`

Verify sample frequency

Parameters

<i>hDevice</i>	a device handle
<i>dSample-Frequency</i>	sample frequency in Hz
<i>dwMeasure-Mode</i>	...
<i>byResolution</i>	...

Returns

sample frequency in Hz when set

4.44 Trigger system

Functions

- double [ScpGetTriggerTimeOut](#) (TpDeviceHandle_t hDevice)
- double [ScpSetTriggerTimeOut](#) (TpDeviceHandle_t hDevice, double dTimeout)
- double [ScpVerifyTriggerTimeOut](#) (TpDeviceHandle_t hDevice, double dTimeout)
- uint64_t [ScpGetTriggerSources](#) (TpDeviceHandle_t hDevice)
- uint64_t [ScpGetTriggerSourcesEx](#) (TpDeviceHandle_t hDevice, uint32_t dwMeasureMode)
- uint64_t [ScpGetTriggerSourceOR](#) (TpDeviceHandle_t hDevice)
- uint64_t [ScpSetTriggerSourceOR](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)
- uint64_t [ScpGetTriggerSourceAND](#) (TpDeviceHandle_t hDevice)
- uint64_t [ScpSetTriggerSourceAND](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)
- uint64_t [ScpGetTriggerKinds](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)
- uint64_t [ScpGetTriggerKindsEx](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask, uint32_t dwMeasureMode)
- uint64_t [ScpGetTriggerKind](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource)
- uint64_t [ScpSetTriggerKind](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint64_t qwTriggerKind)
- double [ScpGetTriggerLevel](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex)
- double [ScpSetTriggerLevel](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex, double dLevel)
- double [ScpGetTriggerHysteresis](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex)
- double [ScpSetTriggerHysteresis](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex, double dHysteresis)
- uint64_t [ScpChGetTriggerKinds](#) (TpDeviceHandle_t hDevice, uint16_t wCh)
- uint64_t [ScpChGetTriggerKindsEx](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwMeasureMode)
- uint64_t [ScpChGetTriggerKind](#) (TpDeviceHandle_t hDevice, uint16_t wCh)
- uint64_t [ScpChSetTriggerKind](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwTriggerKind)
- double [ScpChGetTriggerLevel](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex)
- double [ScpChSetTriggerLevel](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex, double dLevel)
- double [ScpChGetTriggerHysteresis](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex)
- double [ScpChSetTriggerHysteresis](#) (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex, double dHysteresis)
- double [ScpChGetTriggerPulseTime](#) (TpDeviceHandle_t hDevice, uint16_t wCh)
- double [ScpChSetTriggerPulseTime](#) (TpDeviceHandle_t hDevice, uint16_t wCh, double dPulseTime)

4.44.1 Detailed Description

4.44.2 Function Documentation

4.44.2.1 double ScpChGetTriggerHysteresis (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex)

Get current trigger hysteresis

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>dwIndex</i>	...

Returns

current trigger hysteresis: 0 . . 1

4.44.2.2 uint64_t ScpChGetTriggerKind (TpDeviceHandle_t hDevice, uint16_t wCh)

Get current trigger kind for channel

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1

Returns

[TK_*](#) value

4.44.2.3 uint64_t ScpChGetTriggerKinds (TpDeviceHandle_t hDevice, uint16_t wCh)

Get all available trigger kinds for channel

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1

Returns

set of ORed [TK_*](#) values

4.44.2.4 uint64_t ScpChGetTriggerKindsEx (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwMeasureMode)

Get all available trigger kinds for channel

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1
<i>dwMeasure-Mode</i>	...

Returns

set of ORed [TK_*](#) values

4.44.2.5 double ScpChGetTriggerLevel (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex)

Get current trigger level

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1
<i>dwIndex</i>	...

Returns

relative trigger level: 0 .. 1

4.44.2.6 double ScpChGetTriggerPulseTime (TpDeviceHandle_t hDevice, uint16_t wCh)

Get current trigger pulse time

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1

Returns

4.44.2.7 double ScpChSetTriggerHysteresis (TpDeviceHandle_t hDevice, uint16_t wCh, uint32_t dwIndex, double dHysteresis)

Set trigger hysteresis

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1
<i>dwIndex</i>	...
<i>dHysteresis</i>	trigger hysteresis: 0 .. 1

Returns

current trigger hysteresis: 0 .. 1

4.44.2.8 uint64_t ScpChSetTriggerKind (TpDeviceHandle_t hDevice, uint16_t wCh, uint64_t qwTriggerKind)

Set trigger kind for channel

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 .. ScpGetChannelCount () - 1
<i>qwTriggerKind</i>	trigger kind: a TK_* value

Returns

TK_* value

4.44.2.9 double ScpChSetTriggerLevel (TpDeviceHandle_t *hDevice*, uint16_t *wCh*, uint32_t *dwIndex*, double *dLevel*)

Set trigger level

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>dwIndex</i>	...
<i>dLevel</i>	trigger level: 0 . . 1

Returns

current trigger level: 0 . . 1

4.44.2.10 double ScpChSetTriggerPulseTime (TpDeviceHandle_t *hDevice*, uint16_t *wCh*, double *dPulseTime*)

Set trigger pulse time

Parameters

<i>hDevice</i>	a device handle
<i>wCh</i>	channel number: 0 . . ScpGetChannelCount () - 1
<i>dPulseTime</i>	

Returns

4.44.2.11 double ScpGetTriggerHysteresis (TpDeviceHandle_t *hDevice*, uint64_t *qwTriggerSource*, uint32_t *dwIndex*)

Get current trigger hysteresis

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value
<i>dwIndex</i>	...

Returns

current trigger hysteresis: 0 . . 1

4.44.2.12 uint64_t ScpGetTriggerKind (TpDeviceHandle_t *hDevice*, uint64_t *qwTriggerSource*)

Get current trigger kind

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value

Returns

[TK_*](#) value

4.44.2.13 uint64_t ScpGetTriggerKinds (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)

Get available trigger kinds by source(s)

When getting available trigger kinds of multiple source the result is a set of ORed [TK_*](#) values which are supported by all sources.

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTrigger-SourceMask</i>	TS_* value or multiple ORed TS_* values

Returns

set of ORed [TK_*](#) values

4.44.2.14 uint64_t ScpGetTriggerKindsEx (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask, uint32_t dwMeasureMode)

Get available trigger kinds by source(s)

When getting available trigger kinds of multiple source the result is a set of ORed [TK_*](#) values which are supported by all sources.

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTrigger-SourceMask</i>	TS_* value or multiple ORed TS_* values
<i>dwMeasure-Mode</i>	...

Returns

set of ORed [TK_*](#) values

4.44.2.15 `double ScpGetTriggerLevel ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex )`

Get current trigger level

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value
<i>dwIndex</i>	...

Returns

relative trigger level: 0 . . 1

4.44.2.16 `uint64_t ScpGetTriggerSourceAND ( TpDeviceHandle_t hDevice )`

Set trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current trigger sources, set of ORed [TS_*](#) values

4.44.2.17 `uint64_t ScpGetTriggerSourceOR ( TpDeviceHandle_t hDevice )`

Get current trigger sources OR

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current trigger sources, set of ORed [TS_*](#) values

4.44.2.18 `uint64_t ScpGetTriggerSources ( TpDeviceHandle_t hDevice )`

Get supported trigger sources

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

supported trigger sources, set of ORed [TK_*](#) values

4.44.2.19 `uint64_t ScpGetTriggerSourcesEx ( TpDeviceHandle_t hDevice, uint32_t dwMeasureMode )`

Get supported trigger sources

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
<i>dwMeasure-Mode</i>	...

Returns

supported trigger sources, set of ORed [TK_*](#) values

4.44.2.20 `double ScpGetTriggerTimeOut ( TpDeviceHandle_t hDevice )`

Get current trigger timeout in seconds

See also

to

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

trigger timeout in seconds or [TO_INFINITY](#)

4.44.2.21 `double ScpSetTriggerHysteresis ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex, double dHysteresis )`

Set trigger hysteresis

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value
<i>dwIndex</i>	...
<i>dHysteresis</i>	trigger hysteresis: 0 . . 1

Returns

current trigger hysteresis: 0 . . 1

4.44.2.22 `uint64_t ScpSetTriggerKind ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint64_t qwTriggerKind )`

Set trigger kind

See also

[Trigger kinds](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value
<i>qwTriggerKind</i>	trigger kind: a TK_* value

Returns

[TK_*](#) value

4.44.2.23 `double ScpSetTriggerLevel ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSource, uint32_t dwIndex, double dLevel )`

Set trigger level

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSource</i>	TS_* value
<i>dwIndex</i>	...
<i>dLevel</i>	trigger level: 0 . . 1

Returns

current trigger level: 0 . . 1

4.44.2.24 `uint64_t ScpSetTriggerSourceAND ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask )`

Get current trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSourceMask</i>	trigger sources, set of ORed TK_* values

Returns

current trigger sources, set of ORed [TS_*](#) values

4.44.2.25 `uint64_t ScpSetTriggerSourceOR ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask )`

Set trigger sources OR

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTriggerSourceMask</i>	trigger sources, set of ORed TK_* values

Returns

current trigger sources, set of ORed [TS_*](#) values

4.44.2.26 `double ScpSetTriggerTimeOut ( TpDeviceHandle_t hDevice, double dTimeout )`

Set trigger timeout in seconds

See also

to

Parameters

<i>hDevice</i>	a device handle
<i>dTimeout</i>	trigger timeout in seconds or TO_INFINITY

Returns

trigger timeout in seconds or [TO_INFINITY](#)

4.44.2.27 `double ScpVerifyTriggerTimeOut ( TpDeviceHandle_t hDevice, double dTimeout )`

Verify trigger timeout in seconds

See also

to

Parameters

<i>hDevice</i>	a device handle
<i>dTimeout</i>	trigger timeout in seconds or TO_INFINITY

Returns

trigger timeout in seconds or [TO_INFINITY](#)

4.45 Generator

Modules

- [Info](#)
- [Control](#)
- [Events](#)
- [Range](#)
- [Trigger system](#)
- [Arbitrary](#)

4.45.1 Detailed Description

4.46 Info

Functions

- `uint32_t GenGetConnectorType (TpDeviceHandle_t hDevice)`
- `bool8_t GenIsDifferential (TpDeviceHandle_t hDevice)`
- `double GenGetImpedance (TpDeviceHandle_t hDevice)`

4.46.1 Detailed Description

4.46.2 Function Documentation

4.46.2.1 `uint32_t GenGetConnectorType ( TpDeviceHandle_t hDevice )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Output [connector type](#).

4.46.2.2 `double GenGetImpedance ( TpDeviceHandle_t hDevice )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

Output impedance in Ohm.

4.46.2.3 `bool8_t GenIsDifferential ( TpDeviceHandle_t hDevice )`

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if output is differential, 0 otherwise.

4.47 Control

Functions

- [bool8_t GenIsControllable](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenGetOutputOn](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenSetOutputOn](#) ([TpDeviceHandle_t](#) hDevice, [bool8_t](#) bOutputOn)
- [void GenStart](#) ([TpDeviceHandle_t](#) hDevice)
- [void GenStop](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenIsBurstActive](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetBurstCount](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetBurstCountMax](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenSetBurstCount](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwBurstCount)
- [uint32_t GenGetModes](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenGetModesEx](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwSignalType)
- [uint32_t GenGetMode](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenSetMode](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMode)
- [uint32_t GenGetSignalTypes](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenGetSignalType](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenSetSignalType](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwSignalType)
- [double GenGetAmplitudeMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetAmplitudeMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetAmplitude](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetAmplitude](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dAmplitude)
- [double GenVerifyAmplitude](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dAmplitude)
- [void GenGetFrequencyMinMax](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMode, [double](#) *pMin, [double](#) *pMax)
- [double GenGetFrequencyMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetFrequencyMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetFrequency](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency)
- [double GenVerifyFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency)
- [double GenVerifyFrequencyEx](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency, [uint32_t](#) dwMode)
- [double GenGetOffsetMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetOffsetMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetOffset](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetOffset](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dOffset)
- [double GenVerifyOffset](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dOffset)
- [double GenGetPhase](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetPhase](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dPhase)
- [double GenVerifyPhase](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dPhase)
- [double GenGetSymmetry](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetSymmetry](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSymmetry)
- [double GenVerifySymmetry](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSymmetry)

4.47.1 Detailed Description

4.47.2 Function Documentation

4.47.2.1 [double GenGetAmplitude](#) ([TpDeviceHandle_t](#) hDevice)

Get current signal amplitude

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

generator amplitude

4.47.2.2 double GenGetAmplitudeMax (TpDeviceHandle_t hDevice)

Get maximum signal amplitude

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

maximum generator amplitude

4.47.2.3 double GenGetAmplitudeMin (TpDeviceHandle_t hDevice)

Get minimum signal amplitude

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

minimum generator amplitude

4.47.2.4 uint64_t GenGetBurstCount (TpDeviceHandle_t hDevice)**Parameters**

<i>hDevice</i>	a device handle
----------------	-----------------

Returns**4.47.2.5 uint64_t GenGetBurstCountMax (TpDeviceHandle_t hDevice)****Parameters**

<i>hDevice</i>	
----------------	--

Returns**4.47.2.6 double GenGetFrequency (TpDeviceHandle_t hDevice)**

Get current signal/sample frequency

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current signal/sample frequency

4.47.2.7 double GenGetFrequencyMax (TpDeviceHandle_t *hDevice*)

Get maximum signal/sample frequency

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

minimum signal/sample frequency

4.47.2.8 double GenGetFrequencyMin (TpDeviceHandle_t *hDevice*)

Get minimum signal/sample frequency

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

minimum signal/sample frequency

4.47.2.9 void GenGetFrequencyMinMax (TpDeviceHandle_t *hDevice*, uint32_t *dwMode*, double * *pMin*, double * *pMax*)

Parameters

<i>hDevice</i>	
<i>dwMode</i>	
<i>pMin</i>	
<i>pMax</i>	

4.47.2.10 uint32_t GenGetMode (TpDeviceHandle_t *hDevice*)

Get generator mode

See also

[Generator modes](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current generator mode, a [FM_*](#) value

4.47.2.11 `uint32_t GenGetModes ( TpDeviceHandle_t hDevice )`

Get generator modes

See also

[Generator modes](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

available generator modes, a set of [FM_*](#) values

4.47.2.12 `uint32_t GenGetModesEx ( TpDeviceHandle_t hDevice, uint32_t dwSignalType )`

Get generator modes

See also

[Generator modes](#)

Parameters

<i>hDevice</i>	a device handle
<i>dwSignalType</i>	a ST_* signal type

Returns

available generator modes of signal type, a set of [FM_*](#) values

4.47.2.13 `double GenGetOffset ( TpDeviceHandle_t hDevice )`

Get current signal offset

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current generator offset in Volts

4.47.2.14 `double GenGetOffsetMax ( TpDeviceHandle_t hDevice )`

Get maximum signal offset

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

maximum generator offset in Volts

4.47.2.15 double GenGetOffsetMin (TpDeviceHandle_t *hDevice*)

Get minimum signal offset

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

minimum generator offset in Volts

4.47.2.16 bool8_t GenGetOutputOn (TpDeviceHandle_t *hDevice*)

Check whether generator output is on

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if output is on, 0 if output is off.

4.47.2.17 double GenGetPhase (TpDeviceHandle_t *hDevice*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

4.47.2.18 uint32_t GenGetSignalType (TpDeviceHandle_t *hDevice*)

Get the currently selected signal type.

See also

[Signal types](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

the currently selected signal type, a [ST_*](#) value

4.47.2.19 `uint32_t GenGetSignalTypes ( TpDeviceHandle_t hDevice )`

Get available signal types.

See also

[Signal types](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

available signal types, set of [ST_*](#) values

4.47.2.20 `double GenGetSymmetry ( TpDeviceHandle_t hDevice )`

Get current signal symmetry

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current signal symmetry 0 . . . 1

4.47.2.21 `bool8_t GenIsBurstActive ( TpDeviceHandle_t hDevice )`

Check whether burst is active

Parameters

<i>hDevice</i>	
----------------	--

Returns

1 if active 0 otherwise

4.47.2.22 `bool8_t GenIsControllable ( TpDeviceHandle_t hDevice )`**Parameters**

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

1 if controllable, 0 otherwise.

4.47.2.23 double GenSetAmplitude (TpDeviceHandle_t hDevice, double dAmplitude)

Set signal amplitude

Parameters

<i>hDevice</i>	a device handle
<i>dAmplitude</i>	generator amplitude

Returns

generator amplitude

4.47.2.24 uint64_t GenSetBurstCount (TpDeviceHandle_t hDevice, uint64_t qwBurstCount)**Parameters**

<i>hDevice</i>	
<i>qwBurstCount</i>	

Returns**4.47.2.25 double GenSetFrequency (TpDeviceHandle_t hDevice, double dFrequency)**

Set signal/sample frequency

Parameters

<i>hDevice</i>	a device handle
<i>dFrequency</i>	signal/sample frequency

Returns

current signal/sample frequency

4.47.2.26 uint32_t GenSetMode (TpDeviceHandle_t hDevice, uint32_t dwMode)

Set generator mode

See also

[Generator modes](#)

Parameters

<i>hDevice</i>	a device handle
<i>dwMode</i>	generator mode, a FM_* value

Returns

current generator mode, a [FM_*](#) value

4.47.2.27 double GenSetOffset (TpDeviceHandle_t hDevice, double dOffset)

Set signal offset

Parameters

<i>hDevice</i>	a device handle
<i>dOffset</i>	generator offset in Volts

Returns

current generator offset in Volts

4.47.2.28 bool8_t GenSetOutputOn (TpDeviceHandle_t hDevice, bool8_t bOutputOn)

Enable or disable generator output

Parameters

<i>hDevice</i>	a device handle
<i>bOutputOn</i>	?

Returns

1 if output is on, 0 if output is off.

4.47.2.29 double GenSetPhase (TpDeviceHandle_t hDevice, double dPhase)**Parameters**

<i>hDevice</i>	a device handle
<i>dPhase</i>	

Returns**4.47.2.30 uint32_t GenSetSignalType (TpDeviceHandle_t hDevice, uint32_t dwSignalType)**

Sets the signal type of the generator.

See also

[Signal types](#)

Parameters

<i>hDevice</i>	a device handle
<i>dwSignalType</i>	signal type, a ST_* value

Returns

the currently selected signal type, a [ST_*](#) value

4.47.2.31 **double** GenSetSymmetry (**TPDeviceHandle_t** *hDevice*, **double** *dSymmetry*)

Set signal symmetry

Parameters

<i>hDevice</i>	a device handle
<i>dSymmetry</i>	signal symmetry 0 . . . 1

Returns

current signal symmetry 0 . . . 1

4.47.2.32 **void** GenStart (**TPDeviceHandle_t** *hDevice*)

Start generator

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.47.2.33 **void** GenStop (**TPDeviceHandle_t** *hDevice*)

Stop generator

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.47.2.34 **double** GenVerifyAmplitude (**TPDeviceHandle_t** *hDevice*, **double** *dAmplitude*)

Parameters

<i>hDevice</i>	
<i>dAmplitude</i>	

Returns

4.47.2.35 **double** GenVerifyFrequency (**TPDeviceHandle_t** *hDevice*, **double** *dFrequency*)

Parameters

<i>hDevice</i>	
<i>dFrequency</i>	

Returns

4.47.2.36 **double** GenVerifyFrequencyEx (**TPDeviceHandle_t** *hDevice*, **double** *dFrequency*, **uint32_t** *dwMode*)

Parameters

<i>hDevice</i>	
<i>dFrequency</i>	
<i>dwMode</i>	

Returns

4.47.2.37 double GenVerifyOffset (TpDeviceHandle_t *hDevice*, double *dOffset*)

Parameters

<i>hDevice</i>	
<i>dOffset</i>	

Returns

4.47.2.38 double GenVerifyPhase (TpDeviceHandle_t *hDevice*, double *dPhase*)

Parameters

<i>hDevice</i>	
<i>dPhase</i>	

Returns

4.47.2.39 double GenVerifySymmetry (TpDeviceHandle_t *hDevice*, double *dSymmetry*)

Parameters

<i>hDevice</i>	
<i>dSymmetry</i>	

Returns

4.48 Events

Functions

- void [GenSetCallbackBurstCompleted](#) (TpDeviceHandle_t hDevice, TpCallback_t pCallback, void *pData)
- void [GenSetCallbackControllableChanged](#) (TpDeviceHandle_t hDevice, TpCallback_t pCallback, void *pData)
- void [GenSetEventBurstCompleted](#) (TpDeviceHandle_t hDevice, HANDLE hEvent)
- void [GenSetEventControllableChanged](#) (TpDeviceHandle_t hDevice, HANDLE hEvent)
- void [GenSetMessageBurstCompleted](#) (TpDeviceHandle_t hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)
- void [GenSetMessageControllableChanged](#) (TpDeviceHandle_t hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)

4.48.1 Detailed Description

4.48.2 Function Documentation

4.48.2.1 void [GenSetCallbackBurstCompleted](#) (TpDeviceHandle_t *hDevice*, TpCallback_t *pCallback*, void * *pData*)

Parameters

<i>hDevice</i>	a device handle
<i>pCallback</i>	
<i>pData</i>	

4.48.2.2 void [GenSetCallbackControllableChanged](#) (TpDeviceHandle_t *hDevice*, TpCallback_t *pCallback*, void * *pData*)

Parameters

<i>hDevice</i>	a device handle
<i>pCallback</i>	
<i>pData</i>	

4.48.2.3 void [GenSetEventBurstCompleted](#) (TpDeviceHandle_t *hDevice*, HANDLE *hEvent*)

Parameters

<i>hDevice</i>	a device handle
<i>hEvent</i>	

4.48.2.4 void [GenSetEventControllableChanged](#) (TpDeviceHandle_t *hDevice*, HANDLE *hEvent*)

Parameters

<i>hDevice</i>	a device handle
<i>hEvent</i>	

4.48.2.5 void [GenSetMessageBurstCompleted](#) (TpDeviceHandle_t *hDevice*, HWND *hWnd*, WPARAM *wParam*, LPARAM *lParam*)

Parameters

<i>hDevice</i>	a device handle
<i>hWnd</i>	
<i>wParam</i>	
<i>lParam</i>	

4.48.2.6 void GenSetMessageControllableChanged (TpDeviceHandle_t *hDevice*, HWND *hWnd*, WPARAM *wParam*, LPARAM *lParam*)

Parameters

<i>hDevice</i>	a device handle
<i>hWnd</i>	
<i>wParam</i>	
<i>lParam</i>	

4.49 Range

Functions

- [bool8_t GenGetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenSetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice, [bool8_t](#) bEnable)
- [uint32_t GenGetRanges](#) ([TpDeviceHandle_t](#) hDevice, double *pList, [uint32_t](#) dwLength)
- [double GenGetRange](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetRange](#) ([TpDeviceHandle_t](#) hDevice, double dRange)

4.49.1 Detailed Description

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.49.2 Function Documentation

4.49.2.1 [bool8_t GenGetAutoRanging](#) ([TpDeviceHandle_t](#) *hDevice*)

4.49.2.2 [double GenGetRange](#) ([TpDeviceHandle_t](#) *hDevice*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.49.2.3 [uint32_t GenGetRanges](#) ([TpDeviceHandle_t](#) *hDevice*, double * *pList*, [uint32_t](#) *dwLength*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.49.2.4 [bool8_t GenSetAutoRanging](#) ([TpDeviceHandle_t](#) *hDevice*, [bool8_t](#) *bEnable*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.49.2.5 [double GenSetRange](#) ([TpDeviceHandle_t](#) *hDevice*, double *dRange*)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

4.50 Trigger system

Functions

- uint64_t [GenGetTriggerSources](#) (TpDeviceHandle_t hDevice)
- uint64_t [GenGetTriggerSourceAND](#) (TpDeviceHandle_t hDevice)
- uint64_t [GenSetTriggerSourceAND](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)
- uint64_t [GenGetTriggerSourceOR](#) (TpDeviceHandle_t hDevice)
- uint64_t [GenSetTriggerSourceOR](#) (TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask)

4.50.1 Detailed Description

4.50.2 Function Documentation

4.50.2.1 uint64_t GenGetTriggerSourceAND (TpDeviceHandle_t hDevice)

Get current trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current trigger sources, set of ORed [TS_*](#) values

4.50.2.2 uint64_t GenGetTriggerSourceOR (TpDeviceHandle_t hDevice)

Get current trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current trigger sources, set of ORed [TS_*](#) values

4.50.2.3 uint64_t GenGetTriggerSources (TpDeviceHandle_t hDevice)

Get all available trigger sources

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

set of ORed [TS_*](#) values

4.50.2.4 `uint64_t GenSetTriggerSourceAND ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask )`

Set trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTrigger-SourceMask</i>	trigger sources, set of ORed TS_* values

Returns

current trigger sources, set of ORed [TS_*](#) values

4.50.2.5 `uint64_t GenSetTriggerSourceOR ( TpDeviceHandle_t hDevice, uint64_t qwTriggerSourceMask )`

Set trigger sources AND

See also

[Trigger sources](#)

Parameters

<i>hDevice</i>	a device handle
<i>qwTrigger-SourceMask</i>	trigger sources, set of ORed TS_* values

Returns

current trigger sources, set of ORed [TS_*](#) values

4.51 Arbitrary

Functions

- uint64_t [GenGetDataLengthMax](#) (TpDeviceHandle_t hDevice)
- uint64_t [GenGetDataLength](#) (TpDeviceHandle_t hDevice)
- uint64_t [GenVerifyDataLength](#) (TpDeviceHandle_t hDevice, uint64_t qwDataLength)
- uint32_t [GenGetDataRawType](#) (TpDeviceHandle_t hDevice)
- void [GenSetData](#) (TpDeviceHandle_t hDevice, float *pBuffer, uint64_t qwSampleCount)
- void [GenSetDataRaw](#) (TpDeviceHandle_t hDevice, void *pBuffer, uint64_t qwSampleCount)

4.51.1 Detailed Description

4.51.2 Function Documentation

4.51.2.1 uint64_t GenGetDataLength (TpDeviceHandle_t hDevice)

Get current size of arbitrary sample buffer

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

current datalength in samples

4.51.2.2 uint64_t GenGetDataLengthMax (TpDeviceHandle_t hDevice)

Get maximum size of arbitrary sample buffer

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

maximum datalength in samples

4.51.2.3 uint32_t GenGetDataRawType (TpDeviceHandle_t hDevice)

Get arbitrary raw data type

Parameters

<i>hDevice</i>	a device handle
----------------	-----------------

Returns

4.51.2.4 void GenSetData (TpDeviceHandle_t *hDevice*, float * *pBuffer*, uint64_t *qwSampleCount*)

Load arbitrary data into memory

Parameters

<i>hDevice</i>	a device handle
<i>pBuffer</i>	pointer to buffer with sample data
<i>qwSampleCount</i>	number of samples in buffer

4.51.2.5 void GenSetDataRaw (TpDeviceHandle_t *hDevice*, void * *pBuffer*, uint64_t *qwSampleCount*)

Load arbitrary data into memory

Parameters

<i>hDevice</i>	a device handle
<i>pBuffer</i>	pointer to buffer with sample data
<i>qwSampleCount</i>	number of samples in buffer

4.51.2.6 uint64_t GenVerifyDataLength (TpDeviceHandle_t *hDevice*, uint64_t *qwDataLength*)

Verify size of arbitrary sample buffer

Parameters

<i>hDevice</i>	a device handle
<i>qwDataLength</i>	

Returns

4.52 I2CHost

Functions

- [bool8_t I2CRead](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, void *pBuffer, [uint32_t](#) dwSize, [bool8_t](#) bStop)
- [bool8_t I2CReadByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) *pValue)
- [bool8_t I2CReadWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint16_t](#) *pValue)
- [bool8_t I2CWrite](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, void *pBuffer, [uint32_t](#) dwSize, [bool8_t](#) bStop)
- [bool8_t I2CWriteByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue)
- [bool8_t I2CWriteByteByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue1, [uint8_t](#) byValue2)
- [bool8_t I2CWriteByteWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue1, [uint16_t](#) wValue2)
- [bool8_t I2CWriteWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint16_t](#) wValue)
- [double I2CGetSpeed](#) ([TpDeviceHandle_t](#) hDevice)
- [double I2CGetSpeedMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double I2CSetSpeed](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSpeed)
- [double I2CVerifySpeed](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSpeed)
- [bool8_t I2CIsInternalAddress](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress)

4.52.1 Detailed Description

4.52.2 Function Documentation

4.52.2.1 [double I2CGetSpeed](#) ([TpDeviceHandle_t](#) *hDevice*)

4.52.2.2 [double I2CGetSpeedMax](#) ([TpDeviceHandle_t](#) *hDevice*)

4.52.2.3 [bool8_t I2CIsInternalAddress](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*)

4.52.2.4 [bool8_t I2CRead](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, void * *pBuffer*, [uint32_t](#) *dwSize*, [bool8_t](#) *bStop*)

4.52.2.5 [bool8_t I2CReadByte](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint8_t](#) * *pValue*)

4.52.2.6 [bool8_t I2CReadWord](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint16_t](#) * *pValue*)

4.52.2.7 [double I2CSetSpeed](#) ([TpDeviceHandle_t](#) *hDevice*, [double](#) *dSpeed*)

4.52.2.8 [double I2CVerifySpeed](#) ([TpDeviceHandle_t](#) *hDevice*, [double](#) *dSpeed*)

4.52.2.9 [bool8_t I2CWrite](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, void * *pBuffer*, [uint32_t](#) *dwSize*, [bool8_t](#) *bStop*)

4.52.2.10 [bool8_t I2CWriteByte](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint8_t](#) *byValue*)

4.52.2.11 [bool8_t I2CWriteByteByte](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint8_t](#) *byValue1*, [uint8_t](#) *byValue2*)

4.52.2.12 [bool8_t I2CWriteByteWord](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint8_t](#) *byValue1*, [uint16_t](#) *wValue2*)

4.52.2.13 [bool8_t I2CWriteWord](#) ([TpDeviceHandle_t](#) *hDevice*, [uint16_t](#) *wAddress*, [uint16_t](#) *wValue*)

4.53 Types

Typedefs

- `typedef void(* TpCallback\_t)(void *pData)`

4.53.1 Detailed Description

4.53.2 Typedef Documentation

4.53.2.1 `typedef void(* TpCallback_t)(void *pData)`

Definition at line 345 of file libtiepie.h.

Chapter 5

File Documentation

5.1 libtiepie.h File Reference

Header for libtiepie.

```
#include "stdint.h"
```

Macros

- `#define CM_INTERNAL 0`
Internal clock source.
- `#define CM_EXTERNAL 1`
External clock source.
- `#define CKN_COUPLING_COUNT 5`
Number of couplings.
- `#define CKB_DCV 0`
Volt DC.
- `#define CKB_ACV 1`
Volt AC.
- `#define CKB_DCA 2`
Ampere DC.
- `#define CKB_ACA 3`
Ampere AC.
- `#define CKB_OHM 4`
Ohm.
- `#define CK_UNKNOWN 0x0000000000000000`
Invalid coupling type.
- `#define CK_DCV ( 1 << CKB_DCV )`
Volt DC.
- `#define CK_ACV ( 1 << CKB_ACV )`
Volt AC.
- `#define CK_DCA ( 1 << CKB_DCA )`
Ampere DC.
- `#define CK_ACA ( 1 << CKB_ACA )`
Ampere AC.
- `#define CK_OHM ( 1 << CKB_OHM )`
Ohm.

- #define CKM_V (CK_DCV | CK_ACV)
Volt.
- #define CKM_A (CK_DCA | CK_ACA)
Ampere.
- #define CKM_OHM (CK_OHM)
Ohm.
- #define FMN_GENERATORMODE_COUNT 2
Number of generator modes.
- #define FMB_SIGNALFREQUENCY 0
- #define FMB_SAMPLEFREQUENCY 1
- #define FM_UNKNOWN 0x00000000
- #define FM_SIGNALFREQUENCY (1 << FMB_SIGNALFREQUENCY)
- #define FM_SAMPLEFREQUENCY (1 << FMB_SAMPLEFREQUENCY)
- #define MMN_MEASUREMODE_COUNT 2
Number of measure modes.
- #define MMB_STREAM 0
Stream mode bit number.
- #define MMB_BLOCK 1
Block mode bit number.
- #define MM_UNKNOWN 0x00000000
- #define MM_STREAM (1 << MMB_STREAM)
- #define MM_BLOCK (1 << MMB_BLOCK)
- #define STN_SIGNALTYPE_COUNT 6
Number of types.
- #define STB_SINE 0
- #define STB_TRIANGLE 1
- #define STB_SQUARE 2
- #define STB_DC 3
- #define STB_NOISE 4
- #define STB_ARBITRARY 5
- #define ST_UNKNOWN 0x0000000000000000
- #define ST_SINE (1 << STB_SINE)
- #define ST_TRIANGLE (1 << STB_TRIANGLE)
- #define ST_SQUARE (1 << STB_SQUARE)
- #define ST_DC (1 << STB_DC)
- #define ST_NOISE (1 << STB_NOISE)
- #define ST_ARBITRARY (1 << STB_ARBITRARY)
- #define STM_AMPLITUDE (ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_NOISE | ST_ARBITRARY)
- #define STM_OFFSET (ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_DC | ST_NOISE | ST_ARBITRARY)
- #define STM_FREQUENCY (ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY)
- #define STM_PHASE (ST_SINE | ST_TRIANGLE | ST_SQUARE | ST_ARBITRARY)
- #define STM_SYMMETRY (ST_SINE | ST_TRIANGLE | ST_SQUARE)
- #define TH_ALLPRESAMPLES 0xffffffffffff
All presamples.
- #define TKN_KIND_COUNT 7
Number of kinds.
- #define TKB_RISING 0
- #define TKB_FALLING 1
- #define TKB_INWINDOW 2
- #define TKB_OUTWINDOW 3
- #define TKB_EDGE 4
- #define TKB_DROPINWINDOW 5

- #define `TKB_DROPOUTWINDOW` 6
- #define `TK_UNKNOWN` 0x0000000000000000ULL
- #define `TK_RISING` (1ULL << `TKB_RISING`)
- #define `TK_FALLING` (1ULL << `TKB_FALLING`)
- #define `TK_INWINDOW` (1ULL << `TKB_INWINDOW`)
- #define `TK_OUTWINDOW` (1ULL << `TKB_OUTWINDOW`)
- #define `TK_EDGE` (1ULL << `TKB_EDGE`)
- #define `TK_DROPINWINDOW` (1ULL << `TKB_DROPINWINDOW`)
- #define `TK_DROPOUTWINDOW` (1ULL << `TKB_DROPOUTWINDOW`)
- #define `TKM_NONE` 0x0000000000000000ULL
- No trigger kinds.*
- #define `TKM_EDGE` (`TK_RISING` | `TK_FALLING` | `TK_EDGE`)
- #define `TKM_WINDOW` (`TK_INWINDOW` | `TK_OUTWINDOW` | `TK_DROPINWINDOW` | `TK_DROPOUTW-`
`INDOW`)
- #define `TKM_ALL` ((1ULL << `TKN_KIND_COUNT`) - 1)
- All trigger kinds.*
- #define `TO_INFINITY` -1
- No time out.*
- #define `TS_NONE` 0x0000000000000000ULL
- No trigger source.*
- #define `TS_CH1` 0x0000000000000001ULL
- Channel 1.*
- #define `TS_CH2` 0x0000000000000002ULL
- Channel 2.*
- #define `TS_CH3` 0x0000000000000004ULL
- Channel 3.*
- #define `TS_CH4` 0x0000000000000008ULL
- Channel 4.*
- #define `TS_CH5` 0x0000000000000010ULL
- Channel 5.*
- #define `TS_CH6` 0x0000000000000020ULL
- Channel 6.*
- #define `TS_CH7` 0x0000000000000040ULL
- Channel 7.*
- #define `TS_CH8` 0x0000000000000080ULL
- Channel 8.*
- #define `TS_CH9` 0x0000000000000100ULL
- Channel 9.*
- #define `TS_CH10` 0x0000000000000200ULL
- Channel 10.*
- #define `TS_CH11` 0x0000000000000400ULL
- Channel 11.*
- #define `TS_CH12` 0x0000000000000800ULL
- Channel 12.*
- #define `TS_CH13` 0x0000000000001000ULL
- Channel 13.*
- #define `TS_CH14` 0x0000000000002000ULL
- Channel 14.*
- #define `TS_CH15` 0x0000000000004000ULL
- Channel 15.*
- #define `TS_CH16` 0x0000000000008000ULL

- Channel 16.*
 - #define [TS_CH17](#) 0x0000000000010000ULL
- Channel 17.*
 - #define [TS_CH18](#) 0x0000000000020000ULL
- Channel 18.*
 - #define [TS_CH19](#) 0x0000000000040000ULL
- Channel 19.*
 - #define [TS_CH20](#) 0x0000000000080000ULL
- Channel 20.*
 - #define [TS_CH21](#) 0x0000000000010000ULL
- Channel 21.*
 - #define [TS_CH22](#) 0x0000000000020000ULL
- Channel 22.*
 - #define [TS_CH23](#) 0x0000000000040000ULL
- Channel 23.*
 - #define [TS_CH24](#) 0x0000000000080000ULL
- Channel 24.*
 - #define [TS_CH25](#) 0x0000000000010000ULL
- Channel 25.*
 - #define [TS_CH26](#) 0x0000000000020000ULL
- Channel 26.*
 - #define [TS_CH27](#) 0x0000000000040000ULL
- Channel 27.*
 - #define [TS_CH28](#) 0x0000000000080000ULL
- Channel 28.*
 - #define [TS_CH29](#) 0x0000000000010000ULL
- Channel 29.*
 - #define [TS_CH30](#) 0x0000000000020000ULL
- Channel 30.*
 - #define [TS_CH31](#) 0x0000000000040000ULL
- Channel 31.*
 - #define [TS_CH32](#) 0x0000000000080000ULL
- Channel 32.*
 - #define [TS_GENSTOP](#) 0x0400000000000000ULL
- Generator stop.*
 - #define [TS_GENNEW](#) 0x0800000000000000ULL
- Generator new period.*
 - #define [TS_GENSTART](#) 0x1000000000000000ULL
- Generator start.*
 - #define [TS_EXT2](#) 0x2000000000000000ULL
- External 2 (TTL)*
 - #define [TS_EXTANALOG](#) 0x4000000000000000ULL
- External (Analog)*
 - #define [TS_EXT](#) 0x8000000000000000ULL
- External (TTL)*
 - #define [TSN_CHANNEL_COUNT](#) 32
- Number of LSBs reserved for channel trigger sources.*
 - #define [TSM_NONE](#) 0x0000000000000000ULL
- No trigger sources.*
 - #define [TSM_ALL](#) 0xFFFFFFFFFFFFFFFFULL
- All trigger sources.*

- #define `TSM_CHANNELS` ((1ULL << TSN_CHANNEL_COUNT) - 1)
All channel trigger sources.
- #define `TSM_NONCHANNELS` (`TSM_ALL` - `TSM_CHANNELS`)
All non-channel trigger sources.
- #define `TSM_GENALL` (`TS_GENSTART` | `TS_GENNEW` | `TS_GENSTOP`)
All generator trigger sources.
- #define `TPDEVICEHANDLE_INVALID` 0
- #define `DEVICETYPE_OSCILLOSCOPE` 0x00000001
- #define `DEVICETYPE_GENERATOR` 0x00000002
- #define `DEVICETYPE_I2CHOST` 0x00000004
- #define `LIBTIEPIESTATUS_SUCCESS` 0
- #define `LIBTIEPIESTATUS_VALUE_CLIPPED` 1
- #define `LIBTIEPIESTATUS_VALUE_MODIFIED` 2
- #define `LIBTIEPIESTATUS_UNSUCCESSFUL` -1
- #define `LIBTIEPIESTATUS_NOT_SUPPORTED` -2
- #define `LIBTIEPIESTATUS_INVALID_HANDLE` -3
- #define `LIBTIEPIESTATUS_INVALID_VALUE` -4
- #define `LIBTIEPIESTATUS_INVALID_CHANNEL` -5
- #define `LIBTIEPIESTATUS_INVALID_TRIGGER_SOURCE` -6
- #define `LIBTIEPIESTATUS_INVALID_DEVICE_TYPE` -7
- #define `LIBTIEPIESTATUS_INVALID_DEVICE_INDEX` -8
- #define `LIBTIEPIESTATUS_INVALID_DEVICE_ID` -9
- #define `LIBTIEPIESTATUS_INVALID_DEVICE_SERIALNUMBER` -10
- #define `LIBTIEPIESTATUS_DEVICE_GONE` -11
- #define `LIBTIEPIESTATUS_INTERNAL_ADDRESS` -12
- #define `LIBTIEPIESTATUS_NOT_CONTROLLABLE` -13
- #define `CONNECTORTYPE_UNKNOWN` 0x00000000
- #define `CONNECTORTYPE_BNC` 0x00000001
- #define `CONNECTORTYPE_BANANA` 0x00000002
- #define `CONNECTORTYPE_MASK` (`CONNECTORTYPE_BNC` | `CONNECTORTYPE_BANANA`)
- #define `DATARAWTYPE_UNKNOWN` 0x00000000
- #define `DATARAWTYPE_INT8` 0x00000001
- #define `DATARAWTYPE_INT16` 0x00000002
- #define `DATARAWTYPE_INT32` 0x00000004
- #define `DATARAWTYPE_INT64` 0x00000008
- #define `DATARAWTYPE_UINT8` 0x00000010
- #define `DATARAWTYPE_UINT16` 0x00000020
- #define `DATARAWTYPE_UINT32` 0x00000040
- #define `DATARAWTYPE_UINT64` 0x00000080
- #define `IDM_DEVICEID` 0x80000000
- #define `IDM_ALL` 0xffffffff
- #define `IDB_HS3` 0
- #define `IDB_HS4` 1
- #define `IDB_HS4D` 2
- #define `IDB_HS805` 3
- #define `IDB_HP3` 4
- #define `IDB_HS5` 5
- #define `IDB_HL0516` 6
- #define `IDB_PA1` 7
- #define `ID_HS3` (`IDM_DEVICEID` | (1 << `IDB_HS3`))
Handyscope HS3.
- #define `ID_HS4` (`IDM_DEVICEID` | (1 << `IDB_HS4`))
Handyscope HS4.
- #define `ID_HS4D` (`IDM_DEVICEID` | (1 << `IDB_HS4D`))

- Handyscope HS4 DIFF.*
- #define [ID_HS805](#) ([IDM_DEVICEID](#) | (1 << [IDB_HS805](#)))
- TiePieSCOPE HS805.*
- #define [ID_HP3](#) ([IDM_DEVICEID](#) | (1 << [IDB_HP3](#)))
- Handyprobe HP3.*
- #define [ID_HS5](#) ([IDM_DEVICEID](#) | (1 << [IDB_HS5](#)))
- Handyscope HS5.*
- #define [ID_HL0516](#) ([IDM_DEVICEID](#) | (1 << [IDB_HL0516](#)))
- HL0516.*
- #define [ID_PA1](#) ([IDM_DEVICEID](#) | (1 << [IDB_PA1](#)))
- Power Analyzer PA1.*
- #define [TPVERSION_MAJOR](#)(x) (x >> 48)
- #define [TPVERSION_MINOR](#)(x) ((x >> 32) & 0xffff)
- #define [TPVERSION_RELEASE](#)(x) ((x >> 16) & 0xffff)
- #define [TPVERSION_BUILD](#)(x) (x & 0xffff)
- #define [TPDATE_YEAR](#)(x) (x >> 16)
- #define [TPDATE_MONTH](#)(x) ((x >> 8) & 0xff)
- #define [TPDATE_DAY](#)(x) (x & 0xff)
- #define [WM_LIBTIEPIE](#) ([WM_USER](#) + 1337)
- #define [WM_LIBTIEPIE_LST_DEVICEADDED](#) ([WM_LIBTIEPIE](#) + 2)
- #define [WM_LIBTIEPIE_LST_DEVICEREMOVED](#) ([WM_LIBTIEPIE](#) + 3)
- #define [WM_LIBTIEPIE_DEV_REMOVED](#) ([WM_LIBTIEPIE](#) + 4)
- #define [WM_LIBTIEPIE_SCP_DATAREADY](#) ([WM_LIBTIEPIE](#) + 0)
- #define [WM_LIBTIEPIE_SCP_DATAOVERFLOW](#) ([WM_LIBTIEPIE](#) + 1)
- #define [WM_LIBTIEPIE_GEN_BURSTCOMPLETED](#) ([WM_LIBTIEPIE](#) + 5)
- #define [WM_LIBTIEPIE_GEN_CONTROLLABLECHANGED](#) ([WM_LIBTIEPIE](#) + 6)

Typedefs

- typedef void(* [TpCallback_t](#))(void *pData)
- typedef int32_t [LibTiePieStatus_t](#)
- libTiePie status codes*
- typedef uint32_t [TpDeviceHandle_t](#)
- device handle*
- typedef uint64_t [TpVersion_t](#)
- typedef uint32_t [TpDate_t](#)
- typedef uint8_t [bool8_t](#)
- boolean one byte wide*

Functions

- [TpVersion_t](#) [LibGetVersion](#) ()
- uint32_t [LibGetConfig](#) (uint8_t *pBuffer, uint32_t dwBufferLength)
- [LibTiePieStatus_t](#) [LibGetLastStatus](#) ()
- const char * [LibGetLastStatusStr](#) ()
- uint32_t [LstGetCount](#) (uint32_t dwDeviceType)
- [bool8_t](#) [LstGetDeviceCanOpen](#) (uint32_t dwDeviceType, uint32_t dwId)
- uint32_t [LstGetDeviceProductId](#) (uint32_t dwDeviceType, uint32_t dwId)
- uint32_t [LstGetDeviceVendorId](#) (uint32_t dwDeviceType, uint32_t dwId)
- uint32_t [LstGetDeviceName](#) (uint32_t dwDeviceType, uint32_t dwId, char *pBuffer, uint32_t dwBufferLength)
- uint32_t [LstGetDeviceNameShort](#) (uint32_t dwDeviceType, uint32_t dwId, char *pBuffer, uint32_t dwBufferLength)

- uint32_t [LstGetDeviceSerialNumber](#) (uint32_t dwDeviceType, uint32_t dwId)
- [TpDeviceHandle_t LstOpenDevice](#) (uint32_t dwDeviceType, uint32_t dwId)
- void [LstRemoveDevice](#) (uint32_t dwSerialNumber)
- void [LstUpdate](#) (uint32_t dwDeviceIdMask)
- void [LstSetCallbackDeviceAdded](#) (uint32_t dwDeviceType, [TpCallback_t](#) pCallback, void *pData)
- void [LstSetCallbackDeviceRemoved](#) (uint32_t dwDeviceType, [TpCallback_t](#) pCallback, void *pData)
- void [LstSetEventDeviceAdded](#) (uint32_t dwDeviceType, HANDLE hEvent)
- void [LstSetEventDeviceRemoved](#) (uint32_t dwDeviceType, HANDLE hEvent)
- void [LstSetMessageDeviceAdded](#) (uint32_t dwDeviceType, HWND hWnd)
- void [LstSetMessageDeviceRemoved](#) (uint32_t dwDeviceType, HWND hWnd)
- void [DevClose](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t](#) [DevIsRemoved](#) ([TpDeviceHandle_t](#) hDevice)
- [TpVersion_t](#) [DevGetDriverVersion](#) ([TpDeviceHandle_t](#) hDevice)
- [TpVersion_t](#) [DevGetFirmwareVersion](#) ([TpDeviceHandle_t](#) hDevice)
- [TpDate_t](#) [DevGetCalibrationDate](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [DevGetSerialNumber](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [DevGetProductId](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [DevGetVendorId](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [DevGetName](#) ([TpDeviceHandle_t](#) hDevice, char *pBuffer, uint32_t dwBufferLength)
- uint32_t [DevGetNameShort](#) ([TpDeviceHandle_t](#) hDevice, char *pBuffer, uint32_t dwBufferLength)
- void [DevSetCallbackRemoved](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, void *pData)
- void [DevSetEventRemoved](#) ([TpDeviceHandle_t](#) hDevice, HANDLE hEvent)
- void [DevSetMessageRemoved](#) ([TpDeviceHandle_t](#) hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)
- uint16_t [ScpGetChannelCount](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [ScpGetSharedChannelGroupCount](#) ([TpDeviceHandle_t](#) hDevice)
- uint32_t [ScpGetSharedChannelGroup](#) ([TpDeviceHandle_t](#) hDevice, uint32_t dwGroupIndex, uint16_t *pChannelNumbers, uint32_t dwLength)
- [bool8_t](#) [ScpChGetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [bool8_t](#) [ScpChSetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, [bool8_t](#) bEnable)
- uint32_t [ScpChGetConnectorType](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- uint64_t [ScpChGetCouplings](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- uint64_t [ScpChGetCoupling](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- uint64_t [ScpChSetCoupling](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, uint64_t qwCoupling)
- [bool8_t](#) [ScpChGetEnabled](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [bool8_t](#) [ScpChSetEnabled](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, [bool8_t](#) bEnable)
- [bool8_t](#) [ScpChIsDifferential](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [bool8_t](#) [ScpChIsDualRateCapable](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- double [ScpChGetProbeGain](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- double [ScpChSetProbeGain](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, double dProbeGain)
- uint32_t [ScpChGetRanges](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, double *pList, uint32_t dwLength)
- uint32_t [ScpChGetRangesEx](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, uint64_t qwCoupling, double *pList, uint32_t dwLength)
- double [ScpChGetRange](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- double [ScpChSetRange](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, double dRange)
- uint64_t [ScpGetData](#) ([TpDeviceHandle_t](#) hDevice, float **pBuffers, uint16_t wChannelCount, uint64_t qwStartIndex, uint64_t qwSampleCount)
- uint64_t [ScpGetData1Ch](#) ([TpDeviceHandle_t](#) hDevice, float *pBufferCh1, uint64_t qwStartIndex, uint64_t qwSampleCount)
- uint64_t [ScpGetData2Ch](#) ([TpDeviceHandle_t](#) hDevice, float *pBufferCh1, float *pBufferCh2, uint64_t qwStartIndex, uint64_t qwSampleCount)
- uint64_t [ScpGetData3Ch](#) ([TpDeviceHandle_t](#) hDevice, float *pBufferCh1, float *pBufferCh2, float *pBufferCh3, uint64_t qwStartIndex, uint64_t qwSampleCount)
- uint64_t [ScpGetData4Ch](#) ([TpDeviceHandle_t](#) hDevice, float *pBufferCh1, float *pBufferCh2, float *pBufferCh3, float *pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount)

- [uint64_t ScpGetDataRaw](#) ([TpDeviceHandle_t](#) hDevice, void **pBuffers, uint16_t wChannelCount, uint64_t qwStartIndex, uint64_t qwSampleCount)
- [uint64_t ScpGetDataRaw1Ch](#) ([TpDeviceHandle_t](#) hDevice, void *pBufferCh1, uint64_t qwStartIndex, uint64_t qwSampleCount)
- [uint64_t ScpGetDataRaw2Ch](#) ([TpDeviceHandle_t](#) hDevice, void *pBufferCh1, void *pBufferCh2, uint64_t qwStartIndex, uint64_t qwSampleCount)
- [uint64_t ScpGetDataRaw3Ch](#) ([TpDeviceHandle_t](#) hDevice, void *pBufferCh1, void *pBufferCh2, void *pBufferCh3, uint64_t qwStartIndex, uint64_t qwSampleCount)
- [uint64_t ScpGetDataRaw4Ch](#) ([TpDeviceHandle_t](#) hDevice, void *pBufferCh1, void *pBufferCh2, void *pBufferCh3, void *pBufferCh4, uint64_t qwStartIndex, uint64_t qwSampleCount)
- [bool8_t ScpChIsRangeMaxReachable](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [uint32_t ScpChGetDataRawType](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [uint64_t ScpGetValidPreSampleCount](#) ([TpDeviceHandle_t](#) hDevice)
- [void ScpChGetDataValueRange](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, double *pMin, double *pMax)
- [double ScpChGetDataValueMax](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [double ScpChGetDataValueMin](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [void ScpChGetDataRawValueRange](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh, int64_t *pMin, int64_t *pZero, int64_t *pMax)
- [int64_t ScpChGetDataRawValueMax](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [int64_t ScpChGetDataRawValueZero](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [int64_t ScpChGetDataRawValueMin](#) ([TpDeviceHandle_t](#) hDevice, uint16_t wCh)
- [void ScpSetCallbackDataReady](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, void *pData)
- [void ScpSetCallbackDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, void *pData)
- [void ScpSetEventDataReady](#) ([TpDeviceHandle_t](#) hDevice, HANDLE hEvent)
- [void ScpSetEventDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, HANDLE hEvent)
- [void ScpSetMessageDataReady](#) ([TpDeviceHandle_t](#) hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)
- [void ScpSetMessageDataOverflow](#) ([TpDeviceHandle_t](#) hDevice, HWND hWnd, WPARAM wParam, LPARAM lParam)
- [bool8_t ScpStart](#) ([TpDeviceHandle_t](#) hDevice)
- [void ScpStop](#) ([TpDeviceHandle_t](#) hDevice)
- [void ScpForceTrigger](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScpIsDataReady](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScpIsDataOverflow](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScpIsRunning](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScpIsTriggered](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpGetMeasureModes](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpGetMeasureMode](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpSetMeasureMode](#) ([TpDeviceHandle_t](#) hDevice, uint32_t dwMeasureMode)
- [uint32_t ScpGetResolutions](#) ([TpDeviceHandle_t](#) hDevice, uint8_t *pList, uint32_t dwLength)
- [uint8_t ScpGetResolution](#) ([TpDeviceHandle_t](#) hDevice)
- [uint8_t ScpSetResolution](#) ([TpDeviceHandle_t](#) hDevice, uint8_t byResolution)
- [bool8_t ScpIsResolutionEnhanced](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t ScpIsResolutionEnhancedEx](#) ([TpDeviceHandle_t](#) hDevice, uint8_t byResolution)
- [uint32_t ScpGetClockSource](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t ScpSetClockSource](#) ([TpDeviceHandle_t](#) hDevice, uint32_t dwClockSource)
- [double ScpGetPreSampleRatio](#) ([TpDeviceHandle_t](#) hDevice)
- [double ScpSetPreSampleRatio](#) ([TpDeviceHandle_t](#) hDevice, double dPreSampleRatio)
- [uint64_t ScpGetTriggerHoldOffCountMax](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpGetTriggerHoldOffCountMaxEx](#) ([TpDeviceHandle_t](#) hDevice, uint32_t dwMeasureMode)
- [uint64_t ScpGetTriggerHoldOffCount](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpSetTriggerHoldOffCount](#) ([TpDeviceHandle_t](#) hDevice, uint64_t qwTriggerHoldOffCount)
- [uint64_t ScpGetRecordLengthMax](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpGetRecordLengthMaxEx](#) ([TpDeviceHandle_t](#) hDevice, uint32_t dwMeasureMode, uint8_t byResolution)

- [uint64_t ScpGetRecordLength](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpSetRecordLength](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwRecordLength)
- [uint64_t ScpVerifyRecordLength](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwRecordLength)
- [uint64_t ScpVerifyRecordLengthEx](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwRecordLength, [uint32_t](#) dwMeasureMode, [uint8_t](#) byResolution, [uint64_t](#) qwActiveChannelMask)
- [double ScpGetSampleFrequencyMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double ScpGetSampleFrequency](#) ([TpDeviceHandle_t](#) hDevice)
- [double ScpSetSampleFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSampleFrequency)
- [double ScpVerifySampleFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSampleFrequency)
- [double ScpVerifySampleFrequencyEx](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSampleFrequency, [uint32_t](#) dwMeasureMode, [uint8_t](#) byResolution, [uint64_t](#) qwActiveChannelMask)
- [double ScpGetTriggerTimeOut](#) ([TpDeviceHandle_t](#) hDevice)
- [double ScpSetTriggerTimeOut](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dTimeout)
- [double ScpVerifyTriggerTimeOut](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dTimeout)
- [uint64_t ScpGetTriggerSources](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpGetTriggerSourcesEx](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMeasureMode)
- [uint64_t ScpGetTriggerSourceOR](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpSetTriggerSourceOR](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask)
- [uint64_t ScpGetTriggerSourceAND](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t ScpSetTriggerSourceAND](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask)
- [uint64_t ScpGetTriggerKinds](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask)
- [uint64_t ScpGetTriggerKindsEx](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask, [uint32_t](#) dwMeasureMode)
- [uint64_t ScpGetTriggerKind](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource)
- [uint64_t ScpSetTriggerKind](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource, [uint64_t](#) qwTriggerKind)
- [double ScpGetTriggerLevel](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource, [uint32_t](#) dwIndex)
- [double ScpSetTriggerLevel](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource, [uint32_t](#) dwIndex, [double](#) dLevel)
- [double ScpGetTriggerHysteresis](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource, [uint32_t](#) dwIndex)
- [double ScpSetTriggerHysteresis](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSource, [uint32_t](#) dwIndex, [double](#) dHysteresis)
- [uint64_t ScpChGetTriggerKinds](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh)
- [uint64_t ScpChGetTriggerKindsEx](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint32_t](#) dwMeasureMode)
- [uint64_t ScpChGetTriggerKind](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh)
- [uint64_t ScpChSetTriggerKind](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint64_t](#) qwTriggerKind)
- [double ScpChGetTriggerLevel](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint32_t](#) dwIndex)
- [double ScpChSetTriggerLevel](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint32_t](#) dwIndex, [double](#) dLevel)
- [double ScpChGetTriggerHysteresis](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint32_t](#) dwIndex)
- [double ScpChSetTriggerHysteresis](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [uint32_t](#) dwIndex, [double](#) dHysteresis)
- [double ScpChGetTriggerPulseTime](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh)
- [double ScpChSetTriggerPulseTime](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wCh, [double](#) dPulseTime)
- [uint32_t GenGetConnectorType](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenIsDifferential](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetImpedance](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenIsControllable](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenGetOutputOn](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenSetOutputOn](#) ([TpDeviceHandle_t](#) hDevice, [bool8_t](#) bOutputOn)
- [void GenStart](#) ([TpDeviceHandle_t](#) hDevice)
- [void GenStop](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenIsBurstActive](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetBurstCount](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetBurstCountMax](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenSetBurstCount](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwBurstCount)
- [uint32_t GenGetModes](#) ([TpDeviceHandle_t](#) hDevice)

- [uint32_t GenGetModesEx](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwSignalType)
- [uint32_t GenGetMode](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenSetMode](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMode)
- [uint32_t GenGetSignalTypes](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenGetSignalType](#) ([TpDeviceHandle_t](#) hDevice)
- [uint32_t GenSetSignalType](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwSignalType)
- [double GenGetAmplitudeMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetAmplitudeMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetAmplitude](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetAmplitude](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dAmplitude)
- [double GenVerifyAmplitude](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dAmplitude)
- [void GenGetFrequencyMinMax](#) ([TpDeviceHandle_t](#) hDevice, [uint32_t](#) dwMode, [double](#) *pMin, [double](#) *pMax)
- [double GenGetFrequencyMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetFrequencyMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetFrequency](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency)
- [double GenVerifyFrequency](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency)
- [double GenVerifyFrequencyEx](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dFrequency, [uint32_t](#) dwMode)
- [double GenGetOffsetMin](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetOffsetMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenGetOffset](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetOffset](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dOffset)
- [double GenVerifyOffset](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dOffset)
- [double GenGetPhase](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetPhase](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dPhase)
- [double GenVerifyPhase](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dPhase)
- [double GenGetSymmetry](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetSymmetry](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSymmetry)
- [double GenVerifySymmetry](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSymmetry)
- [void GenSetCallbackBurstCompleted](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, [void](#) *pData)
- [void GenSetCallbackControllableChanged](#) ([TpDeviceHandle_t](#) hDevice, [TpCallback_t](#) pCallback, [void](#) *p-Data)
- [void GenSetEventBurstCompleted](#) ([TpDeviceHandle_t](#) hDevice, [HANDLE](#) hEvent)
- [void GenSetEventControllableChanged](#) ([TpDeviceHandle_t](#) hDevice, [HANDLE](#) hEvent)
- [void GenSetMessageBurstCompleted](#) ([TpDeviceHandle_t](#) hDevice, [HWND](#) hWnd, [WPARAM](#) wParam, [LPA-
RAM](#) lParam)
- [void GenSetMessageControllableChanged](#) ([TpDeviceHandle_t](#) hDevice, [HWND](#) hWnd, [WPARAM](#) wParam, [LPA-
RAM](#) lParam)
- [bool8_t GenGetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice)
- [bool8_t GenSetAutoRanging](#) ([TpDeviceHandle_t](#) hDevice, [bool8_t](#) bEnable)
- [uint32_t GenGetRanges](#) ([TpDeviceHandle_t](#) hDevice, [double](#) *pList, [uint32_t](#) dwLength)
- [double GenGetRange](#) ([TpDeviceHandle_t](#) hDevice)
- [double GenSetRange](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dRange)
- [uint64_t GenGetTriggerSources](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetTriggerSourceAND](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenSetTriggerSourceAND](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask)
- [uint64_t GenGetTriggerSourceOR](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenSetTriggerSourceOR](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwTriggerSourceMask)
- [uint64_t GenGetDataLengthMax](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenGetDataLength](#) ([TpDeviceHandle_t](#) hDevice)
- [uint64_t GenVerifyDataLength](#) ([TpDeviceHandle_t](#) hDevice, [uint64_t](#) qwDataLength)
- [uint32_t GenGetDataRawType](#) ([TpDeviceHandle_t](#) hDevice)
- [void GenSetData](#) ([TpDeviceHandle_t](#) hDevice, [float](#) *pBuffer, [uint64_t](#) qwSampleCount)
- [void GenSetDataRaw](#) ([TpDeviceHandle_t](#) hDevice, [void](#) *pBuffer, [uint64_t](#) qwSampleCount)

- [bool8_t I2CRead](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, void *pBuffer, [uint32_t](#) dwSize, [bool8_t](#) bStop)
- [bool8_t I2CReadByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) *pValue)
- [bool8_t I2CReadWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint16_t](#) *pValue)
- [bool8_t I2CWrite](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, void *pBuffer, [uint32_t](#) dwSize, [bool8_t](#) bStop)
- [bool8_t I2CWriteByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue)
- [bool8_t I2CWriteByteByte](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue1, [uint8_t](#) byValue2)
- [bool8_t I2CWriteByteWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint8_t](#) byValue1, [uint16_t](#) wValue2)
- [bool8_t I2CWriteWord](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress, [uint16_t](#) wValue)
- [double I2CGetSpeed](#) ([TpDeviceHandle_t](#) hDevice)
- [double I2CGetSpeedMax](#) ([TpDeviceHandle_t](#) hDevice)
- [double I2CSetSpeed](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSpeed)
- [double I2CVerifySpeed](#) ([TpDeviceHandle_t](#) hDevice, [double](#) dSpeed)
- [bool8_t I2CIsInternalAddress](#) ([TpDeviceHandle_t](#) hDevice, [uint16_t](#) wAddress)

5.1.1 Detailed Description

Header for libtiepie.

Definition in file [libtiepie.h](#).

Index

Arbitrary, [112](#)

- GenGetDataLength, [112](#)
- GenGetDataLengthMax, [112](#)
- GenGetDataRawType, [112](#)
- GenSetData, [112](#)
- GenSetDataRaw, [113](#)
- GenVerifyDataLength, [113](#)

bool8_t

- Types, [44](#)

CK_ACA

- Coupling types, [11](#)

CK_ACV

- Coupling types, [11](#)

CK_DCA

- Coupling types, [11](#)

CK_DCV

- Coupling types, [11](#)

CK_OHM

- Coupling types, [11](#)

CK_UNKNOWN

- Coupling types, [11](#)

CKB_ACA

- Coupling type bit numbers, [10](#)

CKB_ACV

- Coupling type bit numbers, [10](#)

CKB_DCA

- Coupling type bit numbers, [10](#)

CKB_DCV

- Coupling type bit numbers, [10](#)

CKB_OHM

- Coupling type bit numbers, [10](#)

CKM_A

- Coupling type masks, [13](#)

CKM_OHM

- Coupling type masks, [13](#)

CKM_V

- Coupling type masks, [13](#)

CKN_COUPLING_COUNT

- Constants, [8](#)

CM_EXTERNAL

- Clock modes, [9](#)

CM_INTERNAL

- Clock modes, [9](#)

CONNECTORTYPE_BANANA

- Connector types, [37](#)

CONNECTORTYPE_BNC

- Connector types, [37](#)

CONNECTORTYPE_MASK

Connector types, [37](#)

CONNECTORTYPE_UNKNOWN

Connector types, [37](#)

Clock modes, [9](#)

CM_EXTERNAL, [9](#)

CM_INTERNAL, [9](#)

Connector types, [37](#)

CONNECTORTYPE_BANANA, [37](#)

CONNECTORTYPE_BNC, [37](#)

CONNECTORTYPE_MASK, [37](#)

CONNECTORTYPE_UNKNOWN, [37](#)

Constants, [7](#)

CKN_COUPLING_COUNT, [8](#)

STN_SIGNALTYPE_COUNT, [8](#)

TKN_KIND_COUNT, [8](#)

TSN_CHANNEL_COUNT, [8](#)

Control, [97](#)

GenGetAmplitude, [97](#)

GenGetAmplitudeMax, [98](#)

GenGetAmplitudeMin, [98](#)

GenGetBurstCount, [98](#)

GenGetBurstCountMax, [98](#)

GenGetFrequency, [98](#)

GenGetFrequencyMax, [99](#)

GenGetFrequencyMin, [99](#)

GenGetFrequencyMinMax, [99](#)

GenGetMode, [99](#)

GenGetModes, [100](#)

GenGetModesEx, [100](#)

GenGetOffset, [100](#)

GenGetOffsetMax, [100](#)

GenGetOffsetMin, [101](#)

GenGetOutputOn, [101](#)

GenGetPhase, [101](#)

GenGetSignalType, [101](#)

GenGetSignalTypes, [102](#)

GenGetSymmetry, [102](#)

GenIsBurstActive, [102](#)

GenIsControllable, [102](#)

GenSetAmplitude, [102](#)

GenSetBurstCount, [103](#)

GenSetFrequency, [103](#)

GenSetMode, [103](#)

GenSetOffset, [103](#)

GenSetOutputOn, [104](#)

GenSetPhase, [104](#)

GenSetSignalType, [104](#)

GenSetSymmetry, [104](#)

GenStart, [105](#)

- GenStop, [105](#)
- GenVerifyAmplitude, [105](#)
- GenVerifyFrequency, [105](#)
- GenVerifyFrequencyEx, [105](#)
- GenVerifyOffset, [106](#)
- GenVerifyPhase, [106](#)
- GenVerifySymmetry, [106](#)
- Coupling type bit numbers, [10](#)
 - CKB_ACA, [10](#)
 - CKB_ACV, [10](#)
 - CKB_DCA, [10](#)
 - CKB_DCV, [10](#)
 - CKB_OHM, [10](#)
- Coupling type masks, [13](#)
 - CKM_A, [13](#)
 - CKM_OHM, [13](#)
 - CKM_V, [13](#)
- Coupling types, [11](#)
 - CK_ACA, [11](#)
 - CK_ACV, [11](#)
 - CK_DCA, [11](#)
 - CK_DCV, [11](#)
 - CK_OHM, [11](#)
 - CK_UNKNOWN, [11](#)
- DATARAWTYPE_INT16
 - Raw data types, [38](#)
- DATARAWTYPE_INT32
 - Raw data types, [38](#)
- DATARAWTYPE_INT64
 - Raw data types, [38](#)
- DATARAWTYPE_INT8
 - Raw data types, [38](#)
- DATARAWTYPE_UINT16
 - Raw data types, [38](#)
- DATARAWTYPE_UINT32
 - Raw data types, [38](#)
- DATARAWTYPE_UINT64
 - Raw data types, [38](#)
- DATARAWTYPE_UINT8
 - Raw data types, [38](#)
- DATARAWTYPE_UNKNOWN
 - Raw data types, [38](#)
- DEVICETYPE_GENERATOR
 - Device type, [34](#)
- DEVICETYPE_I2CHOST
 - Device type, [34](#)
- Data, [67](#)
 - ScpChGetDataRawType, [67](#)
 - ScpChGetDataRawValueMax, [67](#)
 - ScpChGetDataRawValueMin, [67](#)
 - ScpChGetDataRawValueRange, [68](#)
 - ScpChGetDataRawValueZero, [68](#)
 - ScpChGetDataValueMax, [68](#)
 - ScpChGetDataValueMin, [68](#)
 - ScpChGetDataValueRange, [68](#)
 - ScpChIsRangeMaxReachable, [68](#)
 - ScpGetData, [69](#)
 - ScpGetData1Ch, [69](#)
 - ScpGetData2Ch, [69](#)
 - ScpGetData3Ch, [69](#)
 - ScpGetData4Ch, [70](#)
 - ScpGetDataRaw, [70](#)
 - ScpGetDataRaw1Ch, [70](#)
 - ScpGetDataRaw2Ch, [71](#)
 - ScpGetDataRaw3Ch, [71](#)
 - ScpGetDataRaw4Ch, [71](#)
 - ScpGetValidPreSampleCount, [71](#)
- DevClose
 - Device, [55](#)
- DevGetCalibrationDate
 - Device, [55](#)
- DevGetDriverVersion
 - Device, [56](#)
- DevGetFirmwareVersion
 - Device, [56](#)
- DevGetName
 - Device, [56](#)
- DevGetNameShort
 - Device, [57](#)
- DevGetProductId
 - Device, [57](#)
- DevGetSerialNumber
 - Device, [58](#)
- DevGetVendorId
 - Device, [58](#)
- DevIsRemoved
 - Device, [58](#)
- DevSetCallbackRemoved
 - Events, [59](#)
- DevSetEventRemoved
 - Events, [59](#)
- DevSetMessageRemoved
 - Events, [59](#)
- Device, [55](#)
 - DevClose, [55](#)
 - DevGetCalibrationDate, [55](#)
 - DevGetDriverVersion, [56](#)
 - DevGetFirmwareVersion, [56](#)
 - DevGetName, [56](#)
 - DevGetNameShort, [57](#)
 - DevGetProductId, [57](#)
 - DevGetSerialNumber, [58](#)
 - DevGetVendorId, [58](#)
 - DevIsRemoved, [58](#)
- Device list, [50](#)
 - LstGetCount, [50](#)
 - LstGetDeviceCanOpen, [50](#)
 - LstGetDeviceName, [50](#)
 - LstGetDeviceNameShort, [51](#)
 - LstGetDeviceProductId, [51](#)
 - LstGetDeviceSerialNumber, [51](#)
 - LstGetDeviceVendorId, [51](#)
 - LstOpenDevice, [52](#)
 - LstRemoveDevice, [52](#)
 - LstUpdate, [52](#)
- Device type, [34](#)

- DEVICETYPE_GENERATOR, [34](#)
- DEVICETYPE_I2CHOST, [34](#)
- DeviceID bits, [41](#)
 - IDB_HL0516, [41](#)
 - IDB_HP3, [41](#)
 - IDB_HS3, [41](#)
 - IDB_HS4, [41](#)
 - IDB_HS4D, [41](#)
 - IDB_HS5, [41](#)
 - IDB_HS805, [41](#)
 - IDB_PA1, [41](#)
- DeviceID masks, [40](#)
 - IDM_ALL, [40](#)
 - IDM_DEVICEID, [40](#)
- DeviceID's, [42](#)
 - ID_HL0516, [42](#)
 - ID_HP3, [42](#)
 - ID_HS3, [42](#)
 - ID_HS4, [42](#)
 - ID_HS4D, [42](#)
 - ID_HS5, [42](#)
 - ID_HS805, [43](#)
 - ID_PA1, [43](#)
- Events, [53](#), [59](#), [72](#), [107](#)
 - DevSetCallbackRemoved, [59](#)
 - DevSetEventRemoved, [59](#)
 - DevSetMessageRemoved, [59](#)
 - GenSetCallbackBurstCompleted, [107](#)
 - GenSetCallbackControllableChanged, [107](#)
 - GenSetEventBurstCompleted, [107](#)
 - GenSetEventControllableChanged, [107](#)
 - GenSetMessageBurstCompleted, [107](#)
 - GenSetMessageControllableChanged, [108](#)
 - LstSetCallbackDeviceAdded, [53](#)
 - LstSetCallbackDeviceRemoved, [53](#)
 - LstSetEventDeviceAdded, [53](#)
 - LstSetEventDeviceRemoved, [53](#)
 - LstSetMessageDeviceAdded, [53](#)
 - LstSetMessageDeviceRemoved, [53](#)
 - ScpSetCallbackDataOverflow, [72](#)
 - ScpSetCallbackDataReady, [72](#)
 - ScpSetEventDataOverflow, [72](#)
 - ScpSetEventDataReady, [72](#)
 - ScpSetMessageDataOverflow, [72](#)
 - ScpSetMessageDataReady, [73](#)
- FM_SAMPLEFREQUENCY
 - Generator modes, [15](#)
- FM_SIGNALFREQUENCY
 - Generator modes, [15](#)
- FM_UNKNOWN
 - Generator modes, [15](#)
- FMB_SAMPLEFREQUENCY
 - Generator mode bit numbers, [14](#)
- FMB_SIGNALFREQUENCY
 - Generator mode bit numbers, [14](#)
- Functions, [47](#)
 - GenGetAmplitude
 - Control, [97](#)
 - GenGetAmplitudeMax
 - Control, [98](#)
 - GenGetAmplitudeMin
 - Control, [98](#)
 - GenGetAutoRanging
 - Range, [109](#)
 - GenGetBurstCount
 - Control, [98](#)
 - GenGetBurstCountMax
 - Control, [98](#)
 - GenGetConnectorType
 - Info, [96](#)
 - GenGetDataLength
 - Arbitrary, [112](#)
 - GenGetDataLengthMax
 - Arbitrary, [112](#)
 - GenGetDataRawType
 - Arbitrary, [112](#)
 - GenGetFrequency
 - Control, [98](#)
 - GenGetFrequencyMax
 - Control, [99](#)
 - GenGetFrequencyMin
 - Control, [99](#)
 - GenGetFrequencyMinMax
 - Control, [99](#)
 - GenGetImpedance
 - Info, [96](#)
 - GenGetMode
 - Control, [99](#)
 - GenGetModes
 - Control, [100](#)
 - GenGetModesEx
 - Control, [100](#)
 - GenGetOffset
 - Control, [100](#)
 - GenGetOffsetMax
 - Control, [100](#)
 - GenGetOffsetMin
 - Control, [101](#)
 - GenGetOutputOn
 - Control, [101](#)
 - GenGetPhase
 - Control, [101](#)
 - GenGetRange
 - Range, [109](#)
 - GenGetRanges
 - Range, [109](#)
 - GenGetSignalType
 - Control, [101](#)
 - GenGetSignalTypes
 - Control, [102](#)
 - GenGetSymmetry
 - Control, [102](#)
 - GenGetTriggerSourceAND
 - Trigger system, [110](#)

- GenGetTriggerSourceOR
 - Trigger system, [110](#)
- GenGetTriggerSources
 - Trigger system, [110](#)
- GenIsBurstActive
 - Control, [102](#)
- GenIsControllable
 - Control, [102](#)
- GenIsDifferential
 - Info, [96](#)
- GenSetAmplitude
 - Control, [102](#)
- GenSetAutoRanging
 - Range, [109](#)
- GenSetBurstCount
 - Control, [103](#)
- GenSetCallbackBurstCompleted
 - Events, [107](#)
- GenSetCallbackControllableChanged
 - Events, [107](#)
- GenSetData
 - Arbitrary, [112](#)
- GenSetDataRaw
 - Arbitrary, [113](#)
- GenSetEventBurstCompleted
 - Events, [107](#)
- GenSetEventControllableChanged
 - Events, [107](#)
- GenSetFrequency
 - Control, [103](#)
- GenSetMessageBurstCompleted
 - Events, [107](#)
- GenSetMessageControllableChanged
 - Events, [108](#)
- GenSetMode
 - Control, [103](#)
- GenSetOffset
 - Control, [103](#)
- GenSetOutputOn
 - Control, [104](#)
- GenSetPhase
 - Control, [104](#)
- GenSetRange
 - Range, [109](#)
- GenSetSignalType
 - Control, [104](#)
- GenSetSymmetry
 - Control, [104](#)
- GenSetTriggerSourceAND
 - Trigger system, [111](#)
- GenSetTriggerSourceOR
 - Trigger system, [111](#)
- GenStart
 - Control, [105](#)
- GenStop
 - Control, [105](#)
- GenVerifyAmplitude
 - Control, [105](#)
- GenVerifyDataLength
 - Arbitrary, [113](#)
- GenVerifyFrequency
 - Control, [105](#)
- GenVerifyFrequencyEx
 - Control, [105](#)
- GenVerifyOffset
 - Control, [106](#)
- GenVerifyPhase
 - Control, [106](#)
- GenVerifySymmetry
 - Control, [106](#)
- Generator, [95](#)
- Generator mode bit numbers, [14](#)
 - FMB_SAMPLEFREQUENCY, [14](#)
 - FMB_SIGNALFREQUENCY, [14](#)
- Generator modes, [15](#)
 - FM_SAMPLEFREQUENCY, [15](#)
 - FM_SIGNALFREQUENCY, [15](#)
 - FM_UNKNOWN, [15](#)
- I2CGetSpeed
 - I2CHost, [114](#)
- I2CGetSpeedMax
 - I2CHost, [114](#)
- I2CHost, [114](#)
 - I2CGetSpeed, [114](#)
 - I2CGetSpeedMax, [114](#)
 - I2CIsInternalAddress, [114](#)
 - I2CRead, [114](#)
 - I2CReadByte, [114](#)
 - I2CReadWord, [114](#)
 - I2CSetSpeed, [114](#)
 - I2CVerifySpeed, [114](#)
 - I2CWrite, [114](#)
 - I2CWriteByte, [114](#)
 - I2CWriteByteByte, [114](#)
 - I2CWriteByteWord, [114](#)
 - I2CWriteWord, [114](#)
- I2CIsInternalAddress
 - I2CHost, [114](#)
- I2CRead
 - I2CHost, [114](#)
- I2CReadByte
 - I2CHost, [114](#)
- I2CReadWord
 - I2CHost, [114](#)
- I2CSetSpeed
 - I2CHost, [114](#)
- I2CVerifySpeed
 - I2CHost, [114](#)
- I2CWrite
 - I2CHost, [114](#)
- I2CWriteByte
 - I2CHost, [114](#)
- I2CWriteByteByte
 - I2CHost, [114](#)
- I2CWriteByteWord
 - I2CHost, [114](#)

- I2CWriteWord
 - I2CHost, [114](#)
- ID_HL0516
 - DeviceID's, [42](#)
- ID_HP3
 - DeviceID's, [42](#)
- ID_HS3
 - DeviceID's, [42](#)
- ID_HS4
 - DeviceID's, [42](#)
- ID_HS4D
 - DeviceID's, [42](#)
- ID_HS5
 - DeviceID's, [42](#)
- ID_HS805
 - DeviceID's, [43](#)
- ID_PA1
 - DeviceID's, [43](#)
- IDB_HL0516
 - DeviceID bits, [41](#)
- IDB_HP3
 - DeviceID bits, [41](#)
- IDB_HS3
 - DeviceID bits, [41](#)
- IDB_HS4
 - DeviceID bits, [41](#)
- IDB_HS4D
 - DeviceID bits, [41](#)
- IDB_HS5
 - DeviceID bits, [41](#)
- IDB_HS805
 - DeviceID bits, [41](#)
- IDB_PA1
 - DeviceID bits, [41](#)
- IDM_ALL
 - DeviceID masks, [40](#)
- IDM_DEVICEID
 - DeviceID masks, [40](#)
- Info, [96](#)
 - GenGetConnectorType, [96](#)
 - GenGetImpedance, [96](#)
 - GenIsDifferential, [96](#)
- Input channels, [61](#)
 - ScpChGetAutoRanging, [61](#)
 - ScpChGetConnectorType, [61](#)
 - ScpChGetCoupling, [62](#)
 - ScpChGetCouplings, [62](#)
 - ScpChGetEnabled, [62](#)
 - ScpChGetProbeGain, [63](#)
 - ScpChGetRange, [63](#)
 - ScpChGetRanges, [63](#)
 - ScpChGetRangesEx, [64](#)
 - ScpChIsDifferential, [64](#)
 - ScpChIsDualRateCapable, [64](#)
 - ScpChSetAutoRanging, [64](#)
 - ScpChSetCoupling, [65](#)
 - ScpChSetEnabled, [65](#)
 - ScpChSetProbeGain, [65](#)
 - ScpChSetRange, [65](#)
 - ScpGetChannelCount, [66](#)
 - ScpGetSharedChannelGroup, [66](#)
 - ScpGetSharedChannelGroupCount, [66](#)
- LibGetConfig
 - Library, [48](#)
- LibGetLastStatus
 - Library, [48](#)
- LibGetLastStatusStr
 - Library, [48](#)
- LibGetVersion
 - Library, [49](#)
- LibTiePieStatus_t
 - Types, [44](#)
- Library, [48](#)
 - LibGetConfig, [48](#)
 - LibGetLastStatus, [48](#)
 - LibGetLastStatusStr, [48](#)
 - LibGetVersion, [49](#)
- libtiepie.h, [117](#)
- LstGetCount
 - Device list, [50](#)
- LstGetDeviceCanOpen
 - Device list, [50](#)
- LstGetDeviceName
 - Device list, [50](#)
- LstGetDeviceNameShort
 - Device list, [51](#)
- LstGetDeviceProductId
 - Device list, [51](#)
- LstGetDeviceSerialNumber
 - Device list, [51](#)
- LstGetDeviceVendorId
 - Device list, [51](#)
- LstOpenDevice
 - Device list, [52](#)
- LstRemoveDevice
 - Device list, [52](#)
- LstSetCallbackDeviceAdded
 - Events, [53](#)
- LstSetCallbackDeviceRemoved
 - Events, [53](#)
- LstSetEventDeviceAdded
 - Events, [53](#)
- LstSetEventDeviceRemoved
 - Events, [53](#)
- LstSetMessageDeviceAdded
 - Events, [53](#)
- LstSetMessageDeviceRemoved
 - Events, [53](#)
- LstUpdate
 - Device list, [52](#)
- MM_BLOCK
 - Measure modes, [17](#)
- MM_STREAM
 - Measure modes, [17](#)
- MM_UNKNOWN

- Measure modes, [17](#)
- MMB_BLOCK
 - Measure modes bit numbers, [16](#)
- MMB_STREAM
 - Measure modes bit numbers, [16](#)
- Macro's, [45](#)
 - TPDATE_DAY, [45](#)
 - TPDATE_MONTH, [45](#)
 - TPDATE_YEAR, [45](#)
 - TPVERSION_BUILD, [45](#)
 - TPVERSION_MAJOR, [45](#)
 - TPVERSION_MINOR, [45](#)
 - TPVERSION_RELEASE, [45](#)
- Measure modes, [17](#)
 - MM_BLOCK, [17](#)
 - MM_STREAM, [17](#)
 - MM_UNKNOWN, [17](#)
- Measure modes bit numbers, [16](#)
 - MMB_BLOCK, [16](#)
 - MMB_STREAM, [16](#)
- Measurements, [74](#)
 - ScpForceTrigger, [74](#)
 - ScpGetMeasureMode, [74](#)
 - ScpGetMeasureModes, [74](#)
 - ScplsDataOverflow, [74](#)
 - ScplsDataReady, [75](#)
 - ScplsRunning, [75](#)
 - ScplsTriggered, [75](#)
 - ScpSetMeasureMode, [75](#)
 - ScpStart, [76](#)
 - ScpStop, [76](#)
- Messages, [46](#)
 - WM_LIBTIEPIE, [46](#)
- Oscilloscope, [60](#)
- Range, [109](#)
 - GenGetAutoRanging, [109](#)
 - GenGetRange, [109](#)
 - GenGetRanges, [109](#)
 - GenSetAutoRanging, [109](#)
 - GenSetRange, [109](#)
- Raw data types, [38](#)
 - DATARAWTYPE_INT16, [38](#)
 - DATARAWTYPE_INT32, [38](#)
 - DATARAWTYPE_INT64, [38](#)
 - DATARAWTYPE_INT8, [38](#)
 - DATARAWTYPE_UINT16, [38](#)
 - DATARAWTYPE_UINT32, [38](#)
 - DATARAWTYPE_UINT64, [38](#)
 - DATARAWTYPE_UINT8, [38](#)
 - DATARAWTYPE_UNKNOWN, [38](#)
- Resolution, [77](#)
 - ScpGetResolution, [77](#)
 - ScpGetResolutions, [77](#)
 - ScplsResolutionEnhanced, [77](#)
 - ScplsResolutionEnhancedEx, [77](#)
 - ScpSetResolution, [78](#)
- ST_ARBITRARY
 - Signal types, [19](#)
- ST_DC
 - Signal types, [19](#)
- ST_NOISE
 - Signal types, [19](#)
- ST_SINE
 - Signal types, [19](#)
- ST_SQUARE
 - Signal types, [19](#)
- ST_TRIANGLE
 - Signal types, [19](#)
- ST_UNKNOWN
 - Signal types, [19](#)
- STB_ARBITRARY
 - Signal type bit numbers, [18](#)
- STB_DC
 - Signal type bit numbers, [18](#)
- STB_NOISE
 - Signal type bit numbers, [18](#)
- STB_SINE
 - Signal type bit numbers, [18](#)
- STB_SQUARE
 - Signal type bit numbers, [18](#)
- STB_TRIANGLE
 - Signal type bit numbers, [18](#)
- STM_AMPLITUDE
 - Signal type masks, [20](#)
- STM_FREQUENCY
 - Signal type masks, [20](#)
- STM_OFFSET
 - Signal type masks, [20](#)
- STM_PHASE
 - Signal type masks, [20](#)
- STM_SYMMETRY
 - Signal type masks, [20](#)
- STN_SIGNALTYPE_COUNT
 - Constants, [8](#)
- ScpChGetAutoRanging
 - Input channels, [61](#)
- ScpChGetConnectorType
 - Input channels, [61](#)
- ScpChGetCoupling
 - Input channels, [62](#)
- ScpChGetCouplings
 - Input channels, [62](#)
- ScpChGetDataRawType
 - Data, [67](#)
- ScpChGetDataRawValueMax
 - Data, [67](#)
- ScpChGetDataRawValueMin
 - Data, [67](#)
- ScpChGetDataRawValueRange
 - Data, [68](#)
- ScpChGetDataRawValueZero
 - Data, [68](#)
- ScpChGetDataValueMax
 - Data, [68](#)

- ScpChGetDataValueMin
 - Data, [68](#)
- ScpChGetDataValueRange
 - Data, [68](#)
- ScpChGetEnabled
 - Input channels, [62](#)
- ScpChGetProbeGain
 - Input channels, [63](#)
- ScpChGetRange
 - Input channels, [63](#)
- ScpChGetRanges
 - Input channels, [63](#)
- ScpChGetRangesEx
 - Input channels, [64](#)
- ScpChGetTriggerHysteresis
 - Trigger system, [85](#)
- ScpChGetTriggerKind
 - Trigger system, [85](#)
- ScpChGetTriggerKinds
 - Trigger system, [86](#)
- ScpChGetTriggerKindsEx
 - Trigger system, [86](#)
- ScpChGetTriggerLevel
 - Trigger system, [86](#)
- ScpChGetTriggerPulseTime
 - Trigger system, [87](#)
- ScpChIsDifferential
 - Input channels, [64](#)
- ScpChIsDualRateCapable
 - Input channels, [64](#)
- ScpChIsRangeMaxReachable
 - Data, [68](#)
- ScpChSetAutoRanging
 - Input channels, [64](#)
- ScpChSetCoupling
 - Input channels, [65](#)
- ScpChSetEnabled
 - Input channels, [65](#)
- ScpChSetProbeGain
 - Input channels, [65](#)
- ScpChSetRange
 - Input channels, [65](#)
- ScpChSetTriggerHysteresis
 - Trigger system, [87](#)
- ScpChSetTriggerKind
 - Trigger system, [87](#)
- ScpChSetTriggerLevel
 - Trigger system, [88](#)
- ScpChSetTriggerPulseTime
 - Trigger system, [88](#)
- ScpForceTrigger
 - Measurements, [74](#)
- ScpGetChannelCount
 - Input channels, [66](#)
- ScpGetClockSource
 - Timebase, [79](#)
- ScpGetData
 - Data, [69](#)
- ScpGetData1Ch
 - Data, [69](#)
- ScpGetData2Ch
 - Data, [69](#)
- ScpGetData3Ch
 - Data, [69](#)
- ScpGetData4Ch
 - Data, [70](#)
- ScpGetDataRaw
 - Data, [70](#)
- ScpGetDataRaw1Ch
 - Data, [70](#)
- ScpGetDataRaw2Ch
 - Data, [71](#)
- ScpGetDataRaw3Ch
 - Data, [71](#)
- ScpGetDataRaw4Ch
 - Data, [71](#)
- ScpGetMeasureMode
 - Measurements, [74](#)
- ScpGetMeasureModes
 - Measurements, [74](#)
- ScpGetPreSampleRatio
 - Timebase, [79](#)
- ScpGetRecordLength
 - Timebase, [80](#)
- ScpGetRecordLengthMax
 - Timebase, [80](#)
- ScpGetRecordLengthMaxEx
 - Timebase, [80](#)
- ScpGetResolution
 - Resolution, [77](#)
- ScpGetResolutions
 - Resolution, [77](#)
- ScpGetSampleFrequency
 - Timebase, [80](#)
- ScpGetSampleFrequencyMax
 - Timebase, [80](#)
- ScpGetSharedChannelGroup
 - Input channels, [66](#)
- ScpGetSharedChannelGroupCount
 - Input channels, [66](#)
- ScpGetTriggerHoldOffCount
 - Timebase, [81](#)
- ScpGetTriggerHoldOffCountMax
 - Timebase, [81](#)
- ScpGetTriggerHoldOffCountMaxEx
 - Timebase, [81](#)
- ScpGetTriggerHysteresis
 - Trigger system, [88](#)
- ScpGetTriggerKind
 - Trigger system, [88](#)
- ScpGetTriggerKinds
 - Trigger system, [89](#)
- ScpGetTriggerKindsEx
 - Trigger system, [89](#)
- ScpGetTriggerLevel
 - Trigger system, [90](#)

- ScpGetTriggerSourceAND
 - Trigger system, [90](#)
- ScpGetTriggerSourceOR
 - Trigger system, [90](#)
- ScpGetTriggerSources
 - Trigger system, [90](#)
- ScpGetTriggerSourcesEx
 - Trigger system, [91](#)
- ScpGetTriggerTimeOut
 - Trigger system, [91](#)
- ScpGetValidPreSampleCount
 - Data, [71](#)
- ScplsDataOverflow
 - Measurements, [74](#)
- ScplsDataReady
 - Measurements, [75](#)
- ScplsResolutionEnhanced
 - Resolution, [77](#)
- ScplsResolutionEnhancedEx
 - Resolution, [77](#)
- ScplsRunning
 - Measurements, [75](#)
- ScplsTriggered
 - Measurements, [75](#)
- ScpSetCallbackDataOverflow
 - Events, [72](#)
- ScpSetCallbackDataReady
 - Events, [72](#)
- ScpSetClockSource
 - Timebase, [81](#)
- ScpSetEventDataOverflow
 - Events, [72](#)
- ScpSetEventDataReady
 - Events, [72](#)
- ScpSetMeasureMode
 - Measurements, [75](#)
- ScpSetMessageDataOverflow
 - Events, [72](#)
- ScpSetMessageDataReady
 - Events, [73](#)
- ScpSetPreSampleRatio
 - Timebase, [82](#)
- ScpSetRecordLength
 - Timebase, [82](#)
- ScpSetResolution
 - Resolution, [78](#)
- ScpSetSampleFrequency
 - Timebase, [82](#)
- ScpSetTriggerHoldOffCount
 - Timebase, [82](#)
- ScpSetTriggerHysteresis
 - Trigger system, [91](#)
- ScpSetTriggerKind
 - Trigger system, [92](#)
- ScpSetTriggerLevel
 - Trigger system, [92](#)
- ScpSetTriggerSourceAND
 - Trigger system, [92](#)
- ScpSetTriggerSourceOR
 - Trigger system, [93](#)
- ScpSetTriggerTimeOut
 - Trigger system, [93](#)
- ScpStart
 - Measurements, [76](#)
- ScpStop
 - Measurements, [76](#)
- ScpVerifyRecordLength
 - Timebase, [83](#)
- ScpVerifyRecordLengthEx
 - Timebase, [83](#)
- ScpVerifySampleFrequency
 - Timebase, [83](#)
- ScpVerifySampleFrequencyEx
 - Timebase, [84](#)
- ScpVerifyTriggerTimeOut
 - Trigger system, [93](#)
- Signal type bit numbers, [18](#)
 - STB_ARBITRARY, [18](#)
 - STB_DC, [18](#)
 - STB_NOISE, [18](#)
 - STB_SINE, [18](#)
 - STB_SQUARE, [18](#)
 - STB_TRIANGLE, [18](#)
- Signal type masks, [20](#)
 - STM_AMPLITUDE, [20](#)
 - STM_FREQUENCY, [20](#)
 - STM_OFFSET, [20](#)
 - STM_PHASE, [20](#)
 - STM_SYMMETRY, [20](#)
- Signal types, [19](#)
 - ST_ARBITRARY, [19](#)
 - ST_DC, [19](#)
 - ST_NOISE, [19](#)
 - ST_SINE, [19](#)
 - ST_SQUARE, [19](#)
 - ST_TRIANGLE, [19](#)
 - ST_UNKNOWN, [19](#)
- Status return codes, [35](#)
- TH_ALLPRESAMPLES
 - Trigger holdoff, [21](#)
- TK_DROPINWINDOW
 - Trigger kinds, [23](#)
- TK_DROPOUTWINDOW
 - Trigger kinds, [23](#)
- TK_EDGE
 - Trigger kinds, [23](#)
- TK_FALLING
 - Trigger kinds, [23](#)
- TK_INWINDOW
 - Trigger kinds, [23](#)
- TK_OUTWINDOW
 - Trigger kinds, [23](#)
- TK_RISING
 - Trigger kinds, [23](#)
- TK_UNKNOWN
 - Trigger kinds, [23](#)

- TKB_DROPINWINDOW
 - Trigger kinds bit numbers, [22](#)
- TKB_DROPOUTWINDOW
 - Trigger kinds bit numbers, [22](#)
- TKB_EDGE
 - Trigger kinds bit numbers, [22](#)
- TKB_FALLING
 - Trigger kinds bit numbers, [22](#)
- TKB_INWINDOW
 - Trigger kinds bit numbers, [22](#)
- TKB_OUTWINDOW
 - Trigger kinds bit numbers, [22](#)
- TKB_RISING
 - Trigger kinds bit numbers, [22](#)
- TKM_ALL
 - Trigger kind masks, [24](#)
- TKM_EDGE
 - Trigger kind masks, [24](#)
- TKM_NONE
 - Trigger kind masks, [24](#)
- TKM_WINDOW
 - Trigger kind masks, [24](#)
- TKN_KIND_COUNT
 - Constants, [8](#)
- TO_INFINITY
 - Time outs, [25](#)
- TPDATE_DAY
 - Macro's, [45](#)
- TPDATE_MONTH
 - Macro's, [45](#)
- TPDATE_YEAR
 - Macro's, [45](#)
- TPVERSION_BUILD
 - Macro's, [45](#)
- TPVERSION_MAJOR
 - Macro's, [45](#)
- TPVERSION_MINOR
 - Macro's, [45](#)
- TPVERSION_RELEASE
 - Macro's, [45](#)
- TS_CH1
 - Trigger sources, [27](#)
- TS_CH10
 - Trigger sources, [27](#)
- TS_CH11
 - Trigger sources, [27](#)
- TS_CH12
 - Trigger sources, [27](#)
- TS_CH13
 - Trigger sources, [28](#)
- TS_CH14
 - Trigger sources, [28](#)
- TS_CH15
 - Trigger sources, [28](#)
- TS_CH16
 - Trigger sources, [28](#)
- TS_CH17
 - Trigger sources, [28](#)
- TS_CH18
 - Trigger sources, [28](#)
- TS_CH19
 - Trigger sources, [28](#)
- TS_CH2
 - Trigger sources, [28](#)
- TS_CH20
 - Trigger sources, [28](#)
- TS_CH21
 - Trigger sources, [29](#)
- TS_CH22
 - Trigger sources, [29](#)
- TS_CH23
 - Trigger sources, [29](#)
- TS_CH24
 - Trigger sources, [29](#)
- TS_CH25
 - Trigger sources, [29](#)
- TS_CH26
 - Trigger sources, [29](#)
- TS_CH27
 - Trigger sources, [29](#)
- TS_CH28
 - Trigger sources, [29](#)
- TS_CH29
 - Trigger sources, [29](#)
- TS_CH3
 - Trigger sources, [30](#)
- TS_CH30
 - Trigger sources, [30](#)
- TS_CH31
 - Trigger sources, [30](#)
- TS_CH32
 - Trigger sources, [30](#)
- TS_CH4
 - Trigger sources, [30](#)
- TS_CH5
 - Trigger sources, [30](#)
- TS_CH6
 - Trigger sources, [30](#)
- TS_CH7
 - Trigger sources, [30](#)
- TS_CH8
 - Trigger sources, [30](#)
- TS_CH9
 - Trigger sources, [31](#)
- TS_EXT
 - Trigger sources, [31](#)
- TS_EXT2
 - Trigger sources, [31](#)
- TS_EXTANALOG
 - Trigger sources, [31](#)
- TS_GENNEW
 - Trigger sources, [31](#)
- TS_GENSTART
 - Trigger sources, [31](#)
- TS_GENSTOP
 - Trigger sources, [31](#)

- TS_NONE
 - Trigger sources, [31](#)
- TSM_ALL
 - Trigger source masks, [32](#)
- TSM_CHANNELS
 - Trigger source masks, [32](#)
- TSM_GENALL
 - Trigger source masks, [32](#)
- TSM_NONCHANNELS
 - Trigger source masks, [32](#)
- TSM_NONE
 - Trigger source masks, [32](#)
- TSN_CHANNEL_COUNT
 - Constants, [8](#)
- TiePie device handles, [33](#)
- Time outs, [25](#)
 - TO_INFINITY, [25](#)
- Timebase, [79](#)
 - ScpGetClockSource, [79](#)
 - ScpGetPreSampleRatio, [79](#)
 - ScpGetRecordLength, [80](#)
 - ScpGetRecordLengthMax, [80](#)
 - ScpGetRecordLengthMaxEx, [80](#)
 - ScpGetSampleFrequency, [80](#)
 - ScpGetSampleFrequencyMax, [80](#)
 - ScpGetTriggerHoldOffCount, [81](#)
 - ScpGetTriggerHoldOffCountMax, [81](#)
 - ScpGetTriggerHoldOffCountMaxEx, [81](#)
 - ScpSetClockSource, [81](#)
 - ScpSetPreSampleRatio, [82](#)
 - ScpSetRecordLength, [82](#)
 - ScpSetSampleFrequency, [82](#)
 - ScpSetTriggerHoldOffCount, [82](#)
 - ScpVerifyRecordLength, [83](#)
 - ScpVerifyRecordLengthEx, [83](#)
 - ScpVerifySampleFrequency, [83](#)
 - ScpVerifySampleFrequencyEx, [84](#)
- TpCallback_t
 - Types, [115](#)
- TpDate_t
 - Types, [44](#)
- TpDeviceHandle_t
 - Types, [44](#)
- TpVersion_t
 - Types, [44](#)
- Trigger holdoff, [21](#)
 - TH_ALLPRESAMPLES, [21](#)
- Trigger kind masks, [24](#)
 - TKM_ALL, [24](#)
 - TKM_EDGE, [24](#)
 - TKM_NONE, [24](#)
 - TKM_WINDOW, [24](#)
- Trigger kinds, [23](#)
 - TK_DROPINWINDOW, [23](#)
 - TK_DROPOUTWINDOW, [23](#)
 - TK_EDGE, [23](#)
 - TK_FALLING, [23](#)
 - TK_INWINDOW, [23](#)
 - TK_OUTWINDOW, [23](#)
 - TK_RISING, [23](#)
 - TK_UNKNOWN, [23](#)
- Trigger kinds bit numbers, [22](#)
 - TKB_DROPINWINDOW, [22](#)
 - TKB_DROPOUTWINDOW, [22](#)
 - TKB_EDGE, [22](#)
 - TKB_FALLING, [22](#)
 - TKB_INWINDOW, [22](#)
 - TKB_OUTWINDOW, [22](#)
 - TKB_RISING, [22](#)
- Trigger source masks, [32](#)
 - TSM_ALL, [32](#)
 - TSM_CHANNELS, [32](#)
 - TSM_GENALL, [32](#)
 - TSM_NONCHANNELS, [32](#)
 - TSM_NONE, [32](#)
- Trigger sources, [26](#)
 - TS_CH1, [27](#)
 - TS_CH10, [27](#)
 - TS_CH11, [27](#)
 - TS_CH12, [27](#)
 - TS_CH13, [28](#)
 - TS_CH14, [28](#)
 - TS_CH15, [28](#)
 - TS_CH16, [28](#)
 - TS_CH17, [28](#)
 - TS_CH18, [28](#)
 - TS_CH19, [28](#)
 - TS_CH2, [28](#)
 - TS_CH20, [28](#)
 - TS_CH21, [29](#)
 - TS_CH22, [29](#)
 - TS_CH23, [29](#)
 - TS_CH24, [29](#)
 - TS_CH25, [29](#)
 - TS_CH26, [29](#)
 - TS_CH27, [29](#)
 - TS_CH28, [29](#)
 - TS_CH29, [29](#)
 - TS_CH3, [30](#)
 - TS_CH30, [30](#)
 - TS_CH31, [30](#)
 - TS_CH32, [30](#)
 - TS_CH4, [30](#)
 - TS_CH5, [30](#)
 - TS_CH6, [30](#)
 - TS_CH7, [30](#)
 - TS_CH8, [30](#)
 - TS_CH9, [31](#)
 - TS_EXT, [31](#)
 - TS_EXT2, [31](#)
 - TS_EXTANALOG, [31](#)
 - TS_GENNEW, [31](#)
 - TS_GENSTART, [31](#)
 - TS_GENSTOP, [31](#)
 - TS_NONE, [31](#)
- Trigger system, [85](#), [110](#)

- GenGetTriggerSourceAND, [110](#)
- GenGetTriggerSourceOR, [110](#)
- GenGetTriggerSources, [110](#)
- GenSetTriggerSourceAND, [111](#)
- GenSetTriggerSourceOR, [111](#)
- ScpChGetTriggerHysteresis, [85](#)
- ScpChGetTriggerKind, [85](#)
- ScpChGetTriggerKinds, [86](#)
- ScpChGetTriggerKindsEx, [86](#)
- ScpChGetTriggerLevel, [86](#)
- ScpChGetTriggerPulseTime, [87](#)
- ScpChSetTriggerHysteresis, [87](#)
- ScpChSetTriggerKind, [87](#)
- ScpChSetTriggerLevel, [88](#)
- ScpChSetTriggerPulseTime, [88](#)
- ScpGetTriggerHysteresis, [88](#)
- ScpGetTriggerKind, [88](#)
- ScpGetTriggerKinds, [89](#)
- ScpGetTriggerKindsEx, [89](#)
- ScpGetTriggerLevel, [90](#)
- ScpGetTriggerSourceAND, [90](#)
- ScpGetTriggerSourceOR, [90](#)
- ScpGetTriggerSources, [90](#)
- ScpGetTriggerSourcesEx, [91](#)
- ScpGetTriggerTimeOut, [91](#)
- ScpSetTriggerHysteresis, [91](#)
- ScpSetTriggerKind, [92](#)
- ScpSetTriggerLevel, [92](#)
- ScpSetTriggerSourceAND, [92](#)
- ScpSetTriggerSourceOR, [93](#)
- ScpSetTriggerTimeOut, [93](#)
- ScpVerifyTriggerTimeOut, [93](#)
- Types, [44](#), [115](#)
 - bool8_t, [44](#)
 - LibTiePieStatus_t, [44](#)
 - TpCallback_t, [115](#)
 - TpDate_t, [44](#)
 - TpDeviceHandle_t, [44](#)
 - TpVersion_t, [44](#)
- WM_LIBTIEPIE
 - Messages, [46](#)