LibTiePie

0.3.1

Generated by Doxygen 1.8.1.1

Thu Jul 5 2012 11:04:56

Contents

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:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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 0xffffffffffffffff

  All presamples.

4.13.1 Detailed Description

4.13.2 Macro Definition Documentation

4.13.2.1 #define TH_ALLPRESAMPLES 0xffffffffffffffff

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 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.

4.19.1 Detailed Description

4.19.2 Macro Definition Documentation

4.19.2.1 #define TSM_ALL 0xFFFFFFFFFFFFFFFFULL

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

pBuffer	
dwBufferLength	

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 librarty 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

dwDeviceType	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

dwDeviceType	a device type
dwId	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

dwDeviceType	a device type
dwId	device index or a Device ID or a serial number
pBuffer	
dwBufferLength	

Returns

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

Parameters

dwDeviceType	a device type
dwId	device index or a Device ID or a serial number
pBuffer	
dwBufferLength	

Returns

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

Parameters

dwDeviceType	a device type
dwId	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

dwDeviceType	a device type
dwId	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

dwDeviceType	a device type
dwId	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

dwDeviceType	a device type
dwId	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

dwSerialNumber	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

dwDeviceIdMask	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

dwDeviceType	
pCallback	
pData	

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

Parameters

dwDeviceType	
pCallback	
pData	

4.34.2.3 void LstSetEventDeviceAdded (uint32_t *dwDeviceType,* HANDLE *hEvent*)

Parameters

dwDeviceType	
hEvent	

4.34.2.4 void LstSetEventDeviceRemoved (uint32_t *dwDeviceType,* HANDLE *hEvent*)

Parameters

dwDeviceType	
hEvent	

4.34.2.5 void LstSetMessageDeviceAdded (uint32_t *dwDeviceType,* HWND *hWnd*)

Parameters

dwDeviceType	
hWnd	

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

Parameters

dwDeviceType	
hWnd	

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

hDevice	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

hDevice	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

hDevice	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

hDevice	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

hDevice	a device handle
pBuffer	pointer to buffer to write to
dwBufferLength	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

hDevice	a device handle
pBuffer	pointer to buffer to write to
dwBufferLength	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

hDevice	a device handle

Returns

Instruments Product ID

4.35.2.8 uint32_t DevGetSerialNumber (TpDeviceHandle_t *hDevice*)

Get the instruments serial number.

Parameters

hDevice	a device handle

Returns

Instruments serial number

4.35.2.9 uint32_t DevGetVendorId (TpDeviceHandle_t *hDevice*)

Get the instruments Vendor ID.

Parameters

hDevice	a device handle

Returns

Instruments Vendor ID

4.35.2.10 bool8_t DevIsRemoved (TpDeviceHandle_t *hDevice*)

Check whether an instrument is removed.

Parameters

hDevice	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

hDevice	a device handle
pCallback	pointer to callback function
pData	optional user data

4.36.2.2 void DevSetEventRemoved (TpDeviceHandle_t *hDevice,* HANDLE *hEvent*)

Parameters

hDevice	
hEvent	

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

Parameters

hDevice	
hWnd	
wParam	
lParam	

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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 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

hDevice	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
pList	pointer to array
dwLength	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
qwCoupling	coupling: a CK_* value
pList	pointer to array
dwLength	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

hDevice	a device handle

Returns

`1` if differential, else `0`.

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

Parameters

hDevice	a device handle

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

Set auto ranging.

Parameters

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
bEnable	?

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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
qwCoupling	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
bEnable	?

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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dProbeGain	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dRange	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

hDevice	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

hDevice	a device handle

4.38.2.19 uint32_t ScpGetSharedChannelGroupCount (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`

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

Parameters

hDevice	a device handle

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

Parameters

hDevice	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

hDevice	a device handle

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

Parameters

hDevice	a device handle

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

Get possible maximum value for current available data

Parameters

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle

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

Parameters

hDevice	a device handle
wCh	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	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

hDevice	a device handle
qwSampleCount	number of samples to write into buffer

Returns

number of samples ready

4.39.2.20 uint64_t ScpGetValidPreSampleCount (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	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

hDevice	a device handle

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

Parameters

hDevice	a device handle

4.40.2.3 void ScpSetEventDataOverflow (TpDeviceHandle_t *hDevice,* HANDLE *hEvent*)

Note

Windows only

Parameters

hDevice	a device handle

4.40.2.4 void ScpSetEventDataReady (TpDeviceHandle_t *hDevice,* HANDLE *hEvent*)

Note

Windows only

Parameters

hDevice	a device handle

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

Note

Windows only

Parameters

hDevice	a device handle

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

Note

Windows only

Parameters

hDevice	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 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)

4.41.1 Detailed Description

4.41.2 Function Documentation

4.41.2.1 void ScpForceTrigger (TpDeviceHandle_t *hDevice*)

Force trigger

Parameters

hDevice	a device handle

4.41.2.2 uint32_t ScpGetMeasureMode (TpDeviceHandle_t *hDevice*)

Get oscilloscope measure mode

Parameters

hDevice	a device handle

Returns

measure mode: a CM_* value

4.41.2.3 uint32_t ScpGetMeasureModes (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	a device handle

4.41.2.4 bool8_t ScpIsDataOverflow (TpDeviceHandle_t *hDevice*)

Check whether overflow occurred

Parameters

hDevice	a device handle

Returns

1 if overflow occurred, 0 otherwise

4.41.2.5 bool8_t ScpIsDataReady (TpDeviceHandle_t *hDevice*)

Check whether data is ready

Parameters

hDevice	a device handle

Returns

1 if measuring done, 0 otherwise

4.41.2.6 bool8_t ScpIsRunning (TpDeviceHandle_t *hDevice*)

Check whether instrument is measuring

Parameters

hDevice	a device handle

Returns

1 if instrument is measuring, 0 otherwise

4.41.2.7 bool8_t ScpIsTriggered (TpDeviceHandle_t *hDevice*)

Check is instrument has triggered

Parameters

hDevice	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

hDevice	a device handle
dwMeasure-Mode	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

hDevice	a device handle

Returns

see ScpIsRunning

4.41.2.10 void ScpStop (TpDeviceHandle_t *hDevice*)

Stop measuring

Parameters

hDevice	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 ScpIsResolutionEnhanced (TpDeviceHandle_t hDevice)
- bool8_t ScpIsResolutionEnhancedEx (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

hDevice	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

hDevice	a device handle
pList	pointer to array
dwLength	number of elements in array

Returns

total number of resolutions

4.42.2.3 bool8_t ScpIsResolutionEnhanced (TpDeviceHandle_t *hDevice*)

Check whether current resolution is enhanced

Parameters

hDevice	a device handle

Returns

`1` if current resolution is enhanced, `0` otherwise

4.42.2.4 bool8_t ScpIsResolutionEnhancedEx (TpDeviceHandle_t *hDevice,* uint8_t *byResolution*)

Check whether resolution is enhanced

Parameters

hDevice	a device handle
byResolution	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

hDevice	a device handle
byResolution	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 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)

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

hDevice	a device handle

Returns

current clock source, a CM_* value

4.43.2.2 double ScpGetPreSampleRatio (TpDeviceHandle_t *hDevice*)

Get pre sample ratio

Parameters

hDevice	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

hDevice	a device handle

Returns

current record length

4.43.2.4 uint64_t ScpGetRecordLengthMax (TpDeviceHandle_t *hDevice*)

Get instruments maximum record length

Parameters

hDevice	a device handle

Returns

maximum record length

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

Parameters

hDevice	a device handle

4.43.2.6 double ScpGetSampleFrequency (TpDeviceHandle_t *hDevice*)

Get current sample frequency

Parameters

hDevice	a device handle

Returns

current sample frequency in Hz

4.43.2.7 double ScpGetSampleFrequencyMax (TpDeviceHandle_t *hDevice*)

Get instruments maximum sample frequency

Parameters

hDevice	a device handle

Returns

maximum sample frequency

4.43.2.8 uint64_t ScpGetTriggerHoldOffCount (TpDeviceHandle_t *hDevice*)

Get ...

Parameters

hDevice	a device handle

Returns

...

4.43.2.9 uint64_t ScpGetTriggerHoldOffCountMax (TpDeviceHandle_t *hDevice*)

Get ...

Parameters

hDevice	a device handle

Returns

...

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

Get ...

Parameters

hDevice	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

hDevice	a device handle
dwClockSource	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

hDevice	a device handle
dPreSample-Ratio	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

hDevice	a device handle
qwRecordLength	record length

Returns

current record length

4.43.2.14 double ScpSetSampleFrequency (TpDeviceHandle_t *hDevice*, double *dSampleFrequency*)

Set sample frequency

Parameters

hDevice	a device handle
dSample-Frequency	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

hDevice	a device handle
qwTriggerHold-OffCount	...

Returns

...

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

Parameters

hDevice	a device handle
qwRecordLength	

Returns

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

Parameters

hDevice	a device handle
qwRecordLength	
dwMeasure-Mode	
byResolution	
qwActive-ChannelMask	

Returns

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

Verify sample frequency

Parameters

hDevice	a device handle
dSample-Frequency	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

hDevice	a device handle
dSample-Frequency	sample frequency in Hz
dwMeasure-Mode	...
byResolution	...

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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dwIndex	...

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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dwMeasure-Mode	...

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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dwIndex	...

Returns

relative trigger level: `0..1`

4.44.2.6 double ScpChGetTriggerPulseTime (TpDeviceHandle_t *hDevice*, uint16_t *wCh*)

Get current trigger pulse time

Parameters

hDevice	a device handle
wCh	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dwIndex	...
dHysteresis	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
qwTriggerKind	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dwIndex	...
dLevel	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

hDevice	a device handle
wCh	channel number: `0..ScpGetChannelCount() - 1`
dPulseTime	

Returns

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

Get current trigger hysteresis

Parameters

hDevice	a device handle
qwTriggerSource	TS_* value
dwIndex	...

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

hDevice	a device handle
qwTriggerSource	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

hDevice	a device handle
qwTrigger-SourceMask	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

hDevice	a device handle
qwTrigger-SourceMask	TS_∗ value or multiple ORed TS_∗ values
dwMeasure-Mode	...

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

hDevice	a device handle
qwTriggerSource	TS_∗ value
dwIndex	...

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

hDevice	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

hDevice	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

hDevice	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

hDevice	a device handle
dwMeasure-Mode	...

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

hDevice	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

hDevice	a device handle
qwTriggerSource	TS_∗ value
dwIndex	...
dHysteresis	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

hDevice	a device handle
qwTriggerSource	TS_∗ value
qwTriggerKind	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

hDevice	a device handle
qwTriggerSource	TS_∗ value
dwIndex	...
dLevel	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

hDevice	a device handle
qwTrigger-SourceMask	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

hDevice	a device handle
qwTrigger-SourceMask	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

hDevice	a device handle
dTimeout	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

hDevice	a device handle
dTimeout	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

hDevice	a device handle

Returns

Output connector type.

4.46.2.2 double GenGetImpedance (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	a device handle

Returns

Output impedance in Ohm.

4.46.2.3 bool8_t GenIsDifferential (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	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

hDevice	a device handle

Returns

generator amplitude

4.47.2.2 double GenGetAmplitudeMax (TpDeviceHandle_t *hDevice*)

Get maximum signal amplitude

Parameters

hDevice	a device handle

Returns

maximum generator amplitude

4.47.2.3 double GenGetAmplitudeMin (TpDeviceHandle_t *hDevice*)

Get minimum signal amplitude

Parameters

hDevice	a device handle

Returns

minimum generator amplitude

4.47.2.4 uint64_t GenGetBurstCount (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	a device handle

Returns

4.47.2.5 uint64_t GenGetBurstCountMax (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	

Returns

4.47.2.6 double GenGetFrequency (TpDeviceHandle_t *hDevice*)

Get current signal/sample frequency

Parameters

hDevice	a device handle

Returns

current signal/sample frequency

4.47.2.7 double GenGetFrequencyMax (TpDeviceHandle_t *hDevice*)

Get maximum signal/sample frequency

Parameters

hDevice	a device handle

Returns

minimum signal/sample frequency

4.47.2.8 double GenGetFrequencyMin (TpDeviceHandle_t *hDevice*)

Get minimum signal/sample frequency

Parameters

hDevice	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

hDevice	
dwMode	
pMin	
pMax	

4.47.2.10 uint32_t GenGetMode (TpDeviceHandle_t *hDevice*)

Get generator mode

See also

Generator modes

Parameters

hDevice	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

hDevice	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

hDevice	a device handle
dwSignalType	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

hDevice	a device handle

Returns

current generator offset in Volts

4.47.2.14 double GenGetOffsetMax (TpDeviceHandle_t *hDevice*)

Get maximum signal offset

Parameters

hDevice	a device handle

Returns

maximum generator offset in Volts

4.47.2.15 double GenGetOffsetMin (TpDeviceHandle_t *hDevice*)

Get minimum signal offset

Parameters

hDevice	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

hDevice	a device handle

Returns

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

4.47.2.17 double GenGetPhase (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	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

hDevice	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

hDevice	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

hDevice	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

hDevice	

Returns

`1` if active `0` otherwise

4.47.2.22 bool8_t GenIsControllable (TpDeviceHandle_t *hDevice*)

Parameters

hDevice	a device handle

Returns

`1` if controllable, `0` otherwise.

4.47.2.23 double GenSetAmplitude (TpDeviceHandle_t *hDevice,* double *dAmplitude*)

Set signal amplitude

Parameters

hDevice	a device handle
dAmplitude	generator amplitude

Returns

generator amplitude

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

Parameters

hDevice	
qwBurstCount	

Returns

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

Set signal/sample frequency

Parameters

hDevice	a device handle
dFrequency	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

hDevice	a device handle
dwMode	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

hDevice	a device handle
dOffset	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

hDevice	a device handle
bOutputOn	?

Returns

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

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

Parameters

hDevice	a device handle
dPhase	

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

hDevice	a device handle
dwSignalType	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

hDevice	a device handle
dSymmetry	signal symmetry `0...1`

Returns

current signal symmetry `0...1`

4.47.2.32 void GenStart (TpDeviceHandle_t *hDevice*)

Start generator

Parameters

hDevice	a device handle

4.47.2.33 void GenStop (TpDeviceHandle_t *hDevice*)

Stop generator

Parameters

hDevice	a device handle

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

Parameters

hDevice	
dAmplitude	

Returns

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

Parameters

hDevice	
dFrequency	

Returns

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

Parameters

hDevice	
dFrequency	
dwMode	

Returns

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

Parameters

hDevice	
dOffset	

Returns

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

Parameters

hDevice	
dPhase	

Returns

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

Parameters

hDevice	
dSymmetry	

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

hDevice	a device handle
pCallback	
pData	

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

Parameters

hDevice	a device handle
pCallback	
pData	

4.48.2.3 void GenSetEventBurstCompleted (TpDeviceHandle_t *hDevice,* HANDLE *hEvent*)

Parameters

hDevice	a device handle
hEvent	

4.48.2.4 void GenSetEventControllableChanged (TpDeviceHandle_t *hDevice,* HANDLE *hEvent*)

Parameters

hDevice	a device handle
hEvent	

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

Parameters

hDevice	a device handle
hWnd	
wParam	
lParam	

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

Parameters

hDevice	a device handle
hWnd	
wParam	
lParam	

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

hDevice	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

hDevice	a device handle

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

Parameters

hDevice	a device handle

4.49.2.4 bool8_t GenSetAutoRanging (TpDeviceHandle_t *hDevice,* bool8_t *bEnable*)

Parameters

hDevice	a device handle

4.49.2.5 double GenSetRange (TpDeviceHandle_t *hDevice,* double *dRange*)

Parameters

hDevice	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

hDevice	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

hDevice	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

hDevice	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

hDevice	a device handle
qwTrigger-SourceMask	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

hDevice	a device handle
qwTrigger-SourceMask	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

hDevice	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

hDevice	a device handle

Returns

maximum datalength in samples

4.51.2.3 uint32_t GenGetDataRawType (TpDeviceHandle_t *hDevice*)

Get arbitrary raw data type

Parameters

hDevice	a device handle

Returns

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

Load arbitrary data into memory

Parameters

hDevice	a device handle
pBuffer	pointer to buffer with sample data
qwSampleCount	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

hDevice	a device handle
pBuffer	pointer to buffer with sample data
qwSampleCount	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

hDevice	a device handle
qwDataLength	

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 0xffffffffffffffff

  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_DROPOUTWINDOW)
- #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 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)
- #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 dwBuffer-Length)

- 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 *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)
- 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