libcaer

2.0.0-r6c20a7bfdf0555c72a946f044ad1d0a9d6c04374

Generated by Doxygen 1.8.13

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 caer_bias_coarsefine Struct Reference

`#include <davis.h>`

Data Fields

- uint8_t coarseValue

  Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t fineValue

  Fine current, from 0 to 255, creates small variations in output current.
- bool enabled

  Whether this bias is enabled or not.
- bool sexN

  Bias sex: true for 'N' type, false for 'P' type.
- bool typeNormal

  Bias type: true for 'Normal', false for 'Cascode'.
- bool currentLevelNormal

  Bias current level: true for 'Normal, false for 'Low'.

3.1.1 Detailed Description

On-chip coarse-fine bias current configuration. See 'http://inilabs.com/support/biasing/' for more details.

The documentation for this struct was generated from the following file:

- devices/davis.h

3.2 caer_bias_dynapse Struct Reference

`#include <dynapse.h>`

Data Fields

- uint8_t coarseValue

  Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t fineValue

  Fine current, from 0 to 255, creates small variations in output current.
- bool BiasLowHi

  Bias current level: true for 'HighBias, false for 'LowBias'.
- bool currentLevel

  Bias type: true for 'Normal', false for 'Cascode'.
- bool sex

  Bias sex: true for 'NBias' type, false for 'PBias' type.
- bool enabled

  Whether this bias is enabled or not.
- bool special

  whether this is a special bias.

3.2.1 Detailed Description

On-chip coarse-fine bias current configuration. See 'http://inilabs.com/support/biasing/' for more details.

The documentation for this struct was generated from the following file:

- devices/dynapse.h

3.3 caer_bias_shiftedsource Struct Reference

```
#include <davis.h>
```

Data Fields

- uint8_t refValue

  Shifted-source bias level, from 0 to 63.
- uint8_t regValue

  Shifted-source bias current for buffer amplifier, from 0 to 63.
- enum caer_bias_shiftedsource_operating_mode operatingMode

  Shifted-source operating mode (see 'enum caer_bias_shiftedsource_operating_mode').
- enum caer_bias_shiftedsource_voltage_level voltageLevel

  Shifted-source voltage level (see 'enum caer_bias_shiftedsource_voltage_level').

3.3.1 Detailed Description

On-chip shifted-source bias current configuration. See 'http://inilabs.com/support/biasing/' for more details.

The documentation for this struct was generated from the following file:

- devices/davis.h

3.4 caer_bias_vdac Struct Reference

`#include <davis.h>`

Data Fields

- uint8_t voltageValue
  Voltage, between 0 and 63, as a fraction of 1/64th of VDD=3.3V.
- uint8_t currentValue
  Current, between 0 and 7, that drives the voltage.

3.4.1 Detailed Description

On-chip voltage digital-to-analog converter configuration. See 'http://inilabs.com/support/biasing/' for more details.

The documentation for this struct was generated from the following file:

- devices/davis.h

3.5 caer_davis_info Struct Reference

`#include <davis.h>`

Data Fields

- int16_t deviceID
  Unique device identifier. Also 'source' for events.
- char deviceSerialNumber [8+1]
  Device serial number.
- uint8_t deviceUSBBusNumber
  Device USB bus number.
- uint8_t deviceUSBDeviceAddress
  Device USB device address.
- char * deviceString
  Device information string, for logging purposes.
- int16_t logicVersion
  Logic (FPGA/CPLD) version.
- bool deviceIsMaster
  Whether the device is a time-stamp master or slave.
- int16_t logicClock
  Clock in MHz for main logic (FPGA/CPLD).
- int16_t adcClock
  Clock in MHz for ADC/APS logic (FPGA/CPLD).
- int16_t chipID
  Chip identifier/type.

- int16_t dvsSizeX

  DVS X axis resolution.
- int16_t dvsSizeY

  DVS Y axis resolution.
- bool dvsHasPixelFilter

  Feature test: DVS pixel-level filtering.
- bool dvsHasBackgroundActivityFilter

  Feature test: DVS Background Activity filter.
- bool dvsHasTestEventGenerator

  Feature test: fake event generator (testing/debug).
- int16_t apsSizeX

  APS X axis resolution.
- int16_t apsSizeY

  APS Y axis resolution.
- enum caer_frame_event_color_filter apsColorFilter

  APS color filter type.
- bool apsHasGlobalShutter

  Feature test: APS supports Global Shutter.
- bool apsHasQuadROI

  Feature test: APS supports Quadruple Region-of-Interest readout.
- bool apsHasExternalADC

  Feature test: APS supports External ADC for getting the image.
- bool apsHasInternalADC

  Feature test: APS supports Internal (on-chip) ADC for getting the image.
- bool extInputHasGenerator

  Feature test: External Input module supports Signal-Generation.
- bool extInputHasExtraDetectors

  Feature test: External Input module supports extra detectors (1 & 2).

3.5.1 Detailed Description

DAVIS device-related information.

The documentation for this struct was generated from the following file:

- devices/davis.h

3.6 caer_dvs128_info Struct Reference

```
#include <dvs128.h>
```

Data Fields

- int16_t deviceID
  Unique device identifier. Also 'source' for events.
- char deviceSerialNumber [8+1]
  Device serial number.
- uint8_t deviceUSBBusNumber
  Device USB bus number.
- uint8_t deviceUSBDeviceAddress
  Device USB device address.
- char * deviceString
  Device information string, for logging purposes.
- int16_t logicVersion
  Logic (FPGA/CPLD) version.
- bool deviceIsMaster
  Whether the device is a time-stamp master or slave.
- int16_t dvsSizeX
  DVS X axis resolution.
- int16_t dvsSizeY
  DVS Y axis resolution.

3.6.1 Detailed Description

DVS128 device-related information.

The documentation for this struct was generated from the following file:

- devices/dvs128.h

3.7 caer_dynapse_info Struct Reference

```
#include <dynapse.h>
```

Data Fields

- int16_t deviceID
  Unique device identifier. Also 'source' for events.
- char deviceSerialNumber [8+1]
  Device serial number.
- uint8_t deviceUSBBusNumber
  Device USB bus number.
- uint8_t deviceUSBDeviceAddress
  Device USB device address.
- char * deviceString
  Device information string, for logging purposes.
- int16_t logicVersion
  Logic (FPGA/CPLD) version.
- bool deviceIsMaster
  Whether the device is a time-stamp master or slave.
- int16_t logicClock
  Clock in MHz for main logic (FPGA/CPLD).
- int16_t chipID
  Chip identifier/type.

3.7.1 Detailed Description

Dynap-se device-related information.

The documentation for this struct was generated from the following file:

- devices/dynapse.h

Chapter 4

File Documentation

4.1 devices/davis.h File Reference

```
#include "usb.h"
#include "../events/polarity.h"
#include "../events/special.h"
#include "../events/frame.h"
#include "../events/imu6.h"
#include "../events/sample.h"
```

Data Structures

- struct caer_davis_info
- struct caer_bias_vdac
- struct caer_bias_coarsefine
- struct caer_bias_shiftedsource

Macros

- #define CAER_DEVICE_DAVIS_FX2 1
- #define CAER_DEVICE_DAVIS_FX3 2
- #define DAVIS_CHIP_DAVIS240A 0
- #define DAVIS_CHIP_DAVIS240B 1
- #define DAVIS_CHIP_DAVIS240C 2
- #define DAVIS_CHIP_DAVIS128 3
- #define DAVIS_CHIP_DAVIS346A 4
- #define DAVIS_CHIP_DAVIS346B 5
- #define DAVIS_CHIP_DAVIS640 6
- #define DAVIS_CHIP_DAVISRGB 7
- #define DAVIS_CHIP_DAVIS208 8
- #define DAVIS_CHIP_DAVIS346C 9
- #define DAVIS_CONFIG_MUX 0
- #define DAVIS_CONFIG_DVS 1
- #define DAVIS_CONFIG_APS 2
- #define DAVIS_CONFIG_IMU 3

- #define DAVIS_CONFIG_EXTINPUT 4
- #define DAVIS_CONFIG_BIAS 5
- #define DAVIS_CONFIG_CHIP 5
- #define DAVIS_CONFIG_SYSINFO 6
- #define DAVIS_CONFIG_MICROPHONE 7
- #define DAVIS_CONFIG_USB 9
- #define DAVIS_CONFIG_MUX_RUN 0
- #define DAVIS_CONFIG_MUX_TIMESTAMP_RUN 1
- #define DAVIS_CONFIG_MUX_TIMESTAMP_RESET 2
- #define DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
- #define DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 4
- #define DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL 5
- #define DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL 6
- #define DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 7
- #define DAVIS_CONFIG_MUX_DROP_MIC_ON_TRANSFER_STALL 8
- #define DAVIS_CONFIG_DVS_SIZE_COLUMNS 0
- #define DAVIS_CONFIG_DVS_SIZE_ROWS 1
- #define DAVIS_CONFIG_DVS_ORIENTATION_INFO 2
- #define DAVIS_CONFIG_DVS_RUN 3
- #define DAVIS_CONFIG_DVS_ACK_DELAY_ROW 4
- #define DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN 5
- #define DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW 6
- #define DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN 7
- #define DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL 8
- #define DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS 9
- #define DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 10
- #define DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 11
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 12
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 13
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 14
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 15
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 16
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 17
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 18
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 19
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 20
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 21
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 22
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 23
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 24
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 25
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 26
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 27
- #define DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 28
- #define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 29
- #define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT 30
- #define DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR 31
- #define DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE 32
- #define DAVIS_CONFIG_APS_SIZE_COLUMNS 0
- #define DAVIS_CONFIG_APS_SIZE_ROWS 1
- #define DAVIS_CONFIG_APS_ORIENTATION_INFO 2
- #define DAVIS_CONFIG_APS_COLOR_FILTER 3
- #define DAVIS_CONFIG_APS_RUN 4
- #define DAVIS_CONFIG_APS_RESET_READ 5
- #define DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL 6

- #define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 7
- #define DAVIS_CONFIG_APS_GLOBAL_SHUTTER 8
- #define DAVIS_CONFIG_APS_START_COLUMN_0 9
- #define DAVIS_CONFIG_APS_START_ROW_0 10
- #define DAVIS_CONFIG_APS_END_COLUMN_0 11
- #define DAVIS_CONFIG_APS_END_ROW_0 12
- #define DAVIS_CONFIG_APS_EXPOSURE 13
- #define DAVIS_CONFIG_APS_FRAME_DELAY 14
- #define DAVIS_CONFIG_APS_RESET_SETTLE 15
- #define DAVIS_CONFIG_APS_COLUMN_SETTLE 16
- #define DAVIS_CONFIG_APS_ROW_SETTLE 17
- #define DAVIS_CONFIG_APS_NULL_SETTLE 18
- #define DAVIS_CONFIG_APS_HAS_QUAD_ROI 19
- #define DAVIS_CONFIG_APS_START_COLUMN_1 20
- #define DAVIS_CONFIG_APS_START_ROW_1 21
- #define DAVIS_CONFIG_APS_END_COLUMN_1 22
- #define DAVIS_CONFIG_APS_END_ROW_1 23
- #define DAVIS_CONFIG_APS_START_COLUMN_2 24
- #define DAVIS_CONFIG_APS_START_ROW_2 25
- #define DAVIS_CONFIG_APS_END_COLUMN_2 26
- #define DAVIS_CONFIG_APS_END_ROW_2 27
- #define DAVIS_CONFIG_APS_START_COLUMN_3 28
- #define DAVIS_CONFIG_APS_START_ROW_3 29
- #define DAVIS_CONFIG_APS_END_COLUMN_3 30
- #define DAVIS_CONFIG_APS_END_ROW_3 31
- #define DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC 32
- #define DAVIS_CONFIG_APS_HAS_INTERNAL_ADC 33
- #define DAVIS_CONFIG_APS_USE_INTERNAL_ADC 34
- #define DAVIS_CONFIG_APS_SAMPLE_ENABLE 35
- #define DAVIS_CONFIG_APS_SAMPLE_SETTLE 36
- #define DAVIS_CONFIG_APS_RAMP_RESET 37
- #define DAVIS_CONFIG_APS_RAMP_SHORT_RESET 38
- #define DAVIS_CONFIG_APS_ADC_TEST_MODE 39
- #define DAVISRGB_CONFIG_APS_TRANSFER 50
- #define DAVISRGB_CONFIG_APS_RSFDSETTLE 51
- #define DAVISRGB_CONFIG_APS_GSPDRESET 52
- #define DAVISRGB_CONFIG_APS_GSRESETFALL 53
- #define DAVISRGB_CONFIG_APS_GSTXFALL 54
- #define DAVISRGB_CONFIG_APS_GSFDRESET 55
- #define DAVIS_CONFIG_APS_SNAPSHOT 80
- #define DAVIS_CONFIG_IMU_RUN 0
- #define DAVIS_CONFIG_IMU_TEMP_STANDBY 1
- #define DAVIS_CONFIG_IMU_ACCEL_STANDBY 2
- #define DAVIS_CONFIG_IMU_GYRO_STANDBY 3
- #define DAVIS_CONFIG_IMU_LP_CYCLE 4
- #define DAVIS_CONFIG_IMU_LP_WAKEUP 5
- #define DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 6
- #define DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER 7
- #define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 8
- #define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 9
- #define DAVIS_CONFIG_IMU_ORIENTATION_INFO 10
- #define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR 0
- #define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES 1
- #define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES 2
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES 3

- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY 4
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH 5
- #define DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR 6
- #define DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 7
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL 8
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 9
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 10
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 11
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 12
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE 13
- #define DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS 14
- #define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1 15
- #define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1 16
- #define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1 17
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1 18
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1 19
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1 20
- #define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2 21
- #define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2 22
- #define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2 23
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2 24
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2 25
- #define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2 26
- #define DAVIS_CONFIG_SYSINFO_LOGIC_VERSION 0
- #define DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
- #define DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
- #define DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK 3
- #define DAVIS_CONFIG_SYSINFO_ADC_CLOCK 4
- #define DAVIS_CONFIG_MICROPHONE_RUN 0
- #define DAVIS_CONFIG_MICROPHONE_SAMPLE_FREQUENCY 1
- #define DAVIS_CONFIG_USB_RUN 0
- #define DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 1

- #define IS_DAVIS128(chipID) ((chipID) == DAVIS_CHIP_DAVIS128)
- #define IS_DAVIS208(chipID) ((chipID) == DAVIS_CHIP_DAVIS208)
- #define IS_DAVIS240A(chipID) ((chipID) == DAVIS_CHIP_DAVIS240A)
- #define IS_DAVIS240B(chipID) ((chipID) == DAVIS_CHIP_DAVIS240B)
- #define IS_DAVIS240C(chipID) ((chipID) == DAVIS_CHIP_DAVIS240C)
- #define IS_DAVIS240(chipID) (IS_DAVIS240A(chipID) || IS_DAVIS240B(chipID) || IS_DAVIS240C(chipID))
- #define IS_DAVIS346A(chipID) ((chipID) == DAVIS_CHIP_DAVIS346A)
- #define IS_DAVIS346B(chipID) ((chipID) == DAVIS_CHIP_DAVIS346B)
- #define IS_DAVIS346C(chipID) ((chipID) == DAVIS_CHIP_DAVIS346C)
- #define IS_DAVIS346(chipID) (IS_DAVIS346A(chipID) || IS_DAVIS346B(chipID) || IS_DAVIS346C(chipID))
- #define IS_DAVIS640(chipID) ((chipID) == DAVIS_CHIP_DAVIS640)
- #define IS_DAVISRGB(chipID) ((chipID) == DAVIS_CHIP_DAVISRGB)

- #define DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL 0
- #define DAVIS128_CONFIG_BIAS_APSCAS 1

- #define DAVIS128_CONFIG_BIAS_ADCREFHIGH 2
- #define DAVIS128_CONFIG_BIAS_ADCREFLOW 3
- #define DAVIS128_CONFIG_BIAS_LOCALBUFBN 8
- #define DAVIS128_CONFIG_BIAS_PADFOLLBN 9
- #define DAVIS128_CONFIG_BIAS_DIFFBN 10
- #define DAVIS128_CONFIG_BIAS_ONBN 11
- #define DAVIS128_CONFIG_BIAS_OFFBN 12
- #define DAVIS128_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS128_CONFIG_BIAS_PRBP 14
- #define DAVIS128_CONFIG_BIAS_PRSFBP 15
- #define DAVIS128_CONFIG_BIAS_REFRBP 16
- #define DAVIS128_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS128_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS128_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS128_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS128_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS128_CONFIG_BIAS_AEPDBN 23
- #define DAVIS128_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS128_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS128_CONFIG_BIAS_IFREFRBN 26
- #define DAVIS128_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS128_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS128_CONFIG_BIAS_SSP 35
- #define DAVIS128_CONFIG_BIAS_SSN 36

- #define DAVIS128_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS128_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS128_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS128_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS128_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS128_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS128_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS128_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS128_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS128_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS128_CONFIG_CHIP_AERNAROW 140
- #define DAVIS128_CONFIG_CHIP_USEAOUT 141
- #define DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER 142
- #define DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER 143

- #define DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 0
- #define DAVIS208_CONFIG_BIAS_APSCAS 1
- #define DAVIS208_CONFIG_BIAS_ADCREFHIGH 2
- #define DAVIS208_CONFIG_BIAS_ADCREFLOW 3
- #define DAVIS208_CONFIG_BIAS_RESETHIGHPASS 6
- #define DAVIS208_CONFIG_BIAS_REFSS 7

- #define DAVIS208_CONFIG_BIAS_LOCALBUFBN 8
- #define DAVIS208_CONFIG_BIAS_PADFOLLBN 9
- #define DAVIS208_CONFIG_BIAS_DIFFBN 10
- #define DAVIS208_CONFIG_BIAS_ONBN 11
- #define DAVIS208_CONFIG_BIAS_OFFBN 12
- #define DAVIS208_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS208_CONFIG_BIAS_PRBP 14
- #define DAVIS208_CONFIG_BIAS_PRSFBP 15
- #define DAVIS208_CONFIG_BIAS_REFRBP 16
- #define DAVIS208_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS208_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS208_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS208_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS208_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS208_CONFIG_BIAS_AEPDBN 23
- #define DAVIS208_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS208_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS208_CONFIG_BIAS_IFREFRBN 26
- #define DAVIS208_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS208_CONFIG_BIAS_REGBIASBP 28
- #define DAVIS208_CONFIG_BIAS_REFSSBN 30
- #define DAVIS208_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS208_CONFIG_BIAS_SSP 35
- #define DAVIS208_CONFIG_BIAS_SSN 36

- #define DAVIS208_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS208_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS208_CONFIG_CHIP_AERNAROW 140
- #define DAVIS208_CONFIG_CHIP_USEAOUT 141
- #define DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 142
- #define DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG 145
- #define DAVIS208_CONFIG_CHIP_SELECTBIASREFSS 146
- #define DAVIS208_CONFIG_CHIP_SELECTSENSE 147
- #define DAVIS208_CONFIG_CHIP_SELECTPOSFB 148
- #define DAVIS208_CONFIG_CHIP_SELECTHIGHPASS 149

- #define DAVIS240_CONFIG_BIAS_DIFFBN 0

- #define DAVIS240_CONFIG_BIAS_ONBN 1
- #define DAVIS240_CONFIG_BIAS_OFFBN 2
- #define DAVIS240_CONFIG_BIAS_APSCASEPC 3
- #define DAVIS240_CONFIG_BIAS_DIFFCASBNC 4
- #define DAVIS240_CONFIG_BIAS_APSROSFBN 5
- #define DAVIS240_CONFIG_BIAS_LOCALBUFBN 6
- #define DAVIS240_CONFIG_BIAS_PIXINVBN 7
- #define DAVIS240_CONFIG_BIAS_PRBP 8
- #define DAVIS240_CONFIG_BIAS_PRSFBP 9
- #define DAVIS240_CONFIG_BIAS_REFRBP 10
- #define DAVIS240_CONFIG_BIAS_AEPDBN 11
- #define DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN 12
- #define DAVIS240_CONFIG_BIAS_AEPUXBP 13
- #define DAVIS240_CONFIG_BIAS_AEPUYBP 14
- #define DAVIS240_CONFIG_BIAS_IFTHRBN 15
- #define DAVIS240_CONFIG_BIAS_IFREFRBN 16
- #define DAVIS240_CONFIG_BIAS_PADFOLLBN 17
- #define DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN 18
- #define DAVIS240_CONFIG_BIAS_BIASBUFFER 19
- #define DAVIS240_CONFIG_BIAS_SSP 20
- #define DAVIS240_CONFIG_BIAS_SSN 21

- #define DAVIS240_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS240_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS240_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS240_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS240_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS240_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS240_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS240_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS240_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS240_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL 139
- #define DAVIS240_CONFIG_CHIP_AERNAROW 140
- #define DAVIS240_CONFIG_CHIP_USEAOUT 141
- #define DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER 142

- #define DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL 0
- #define DAVIS346_CONFIG_BIAS_APSCAS 1
- #define DAVIS346_CONFIG_BIAS_ADCREFHIGH 2
- #define DAVIS346_CONFIG_BIAS_ADCREFLOW 3
- #define DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE 4
- #define DAVIS346_CONFIG_BIAS_LOCALBUFBN 8
- #define DAVIS346_CONFIG_BIAS_PADFOLLBN 9
- #define DAVIS346_CONFIG_BIAS_DIFFBN 10
- #define DAVIS346_CONFIG_BIAS_ONBN 11
- #define DAVIS346_CONFIG_BIAS_OFFBN 12

- #define DAVIS346_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS346_CONFIG_BIAS_PRBP 14
- #define DAVIS346_CONFIG_BIAS_PRSFBP 15
- #define DAVIS346_CONFIG_BIAS_REFRBP 16
- #define DAVIS346_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS346_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS346_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS346_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS346_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS346_CONFIG_BIAS_AEPDBN 23
- #define DAVIS346_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS346_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS346_CONFIG_BIAS_IFREFRBN 26
- #define DAVIS346_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS346_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS346_CONFIG_BIAS_SSP 35
- #define DAVIS346_CONFIG_BIAS_SSN 36

- #define DAVIS346_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS346_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS346_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS346_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS346_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS346_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS346_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS346_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS346_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS346_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS346_CONFIG_CHIP_AERNAROW 140
- #define DAVIS346_CONFIG_CHIP_USEAOUT 141
- #define DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER 142
- #define DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVIS346_CONFIG_CHIP_TESTADC 144

- #define DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL 0
- #define DAVIS640_CONFIG_BIAS_APSCAS 1
- #define DAVIS640_CONFIG_BIAS_ADCREFHIGH 2
- #define DAVIS640_CONFIG_BIAS_ADCREFLOW 3
- #define DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE 4
- #define DAVIS640_CONFIG_BIAS_LOCALBUFBN 8
- #define DAVIS640_CONFIG_BIAS_PADFOLLBN 9
- #define DAVIS640_CONFIG_BIAS_DIFFBN 10
- #define DAVIS640_CONFIG_BIAS_ONBN 11
- #define DAVIS640_CONFIG_BIAS_OFFBN 12
- #define DAVIS640_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS640_CONFIG_BIAS_PRBP 14

- #define DAVIS640_CONFIG_BIAS_PRSFBP 15
- #define DAVIS640_CONFIG_BIAS_REFRBP 16
- #define DAVIS640_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS640_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS640_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS640_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS640_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS640_CONFIG_BIAS_AEPDBN 23
- #define DAVIS640_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS640_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS640_CONFIG_BIAS_IFREFRBN 26
- #define DAVIS640_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS640_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS640_CONFIG_BIAS_SSP 35
- #define DAVIS640_CONFIG_BIAS_SSN 36

- #define DAVIS640_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS640_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS640_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS640_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS640_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS640_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS640_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS640_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS640_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS640_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS640_CONFIG_CHIP_AERNAROW 140
- #define DAVIS640_CONFIG_CHIP_USEAOUT 141
- #define DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER 142
- #define DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVIS640_CONFIG_CHIP_TESTADC 144

- #define DAVISRGB_CONFIG_BIAS_APSCAS 0
- #define DAVISRGB_CONFIG_BIAS_OVG1LO 1
- #define DAVISRGB_CONFIG_BIAS_OVG2LO 2
- #define DAVISRGB_CONFIG_BIAS_TX2OVG2HI 3
- #define DAVISRGB_CONFIG_BIAS_GND07 4
- #define DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE 5
- #define DAVISRGB_CONFIG_BIAS_ADCREFHIGH 6
- #define DAVISRGB_CONFIG_BIAS_ADCREFLOW 7
- #define DAVISRGB_CONFIG_BIAS_IFREFRBN 8
- #define DAVISRGB_CONFIG_BIAS_IFTHRBN 9
- #define DAVISRGB_CONFIG_BIAS_LOCALBUFBN 10
- #define DAVISRGB_CONFIG_BIAS_PADFOLLBN 11
- #define DAVISRGB_CONFIG_BIAS_PIXINVBN 13
- #define DAVISRGB_CONFIG_BIAS_DIFFBN 14

- #define DAVISRGB_CONFIG_BIAS_ONBN 15
- #define DAVISRGB_CONFIG_BIAS_OFFBN 16
- #define DAVISRGB_CONFIG_BIAS_PRBP 17
- #define DAVISRGB_CONFIG_BIAS_PRSFBP 18
- #define DAVISRGB_CONFIG_BIAS_REFRBP 19
- #define DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN 20
- #define DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN 22
- #define DAVISRGB_CONFIG_BIAS_FALLTIMEBN 23
- #define DAVISRGB_CONFIG_BIAS_RISETIMEBP 24
- #define DAVISRGB_CONFIG_BIAS_READOUTBUFBP 25
- #define DAVISRGB_CONFIG_BIAS_APSROSFBN 26
- #define DAVISRGB_CONFIG_BIAS_ADCCOMPBP 27
- #define DAVISRGB_CONFIG_BIAS_DACBUFBP 28
- #define DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN 30
- #define DAVISRGB_CONFIG_BIAS_AEPDBN 31
- #define DAVISRGB_CONFIG_BIAS_AEPUXBP 32
- #define DAVISRGB_CONFIG_BIAS_AEPUYBP 33
- #define DAVISRGB_CONFIG_BIAS_BIASBUFFER 34
- #define DAVISRGB_CONFIG_BIAS_SSP 35
- #define DAVISRGB_CONFIG_BIAS_SSN 36

- #define DAVISRGB_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVISRGB_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVISRGB_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVISRGB_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVISRGB_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVISRGB_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVISRGB_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVISRGB_CONFIG_CHIP_BIASMUX0 135
- #define DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVISRGB_CONFIG_CHIP_AERNAROW 140
- #define DAVISRGB_CONFIG_CHIP_USEAOUT 141
- #define DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVISRGB_CONFIG_CHIP_TESTADC 144
- #define DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO 145
- #define DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO 146
- #define DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI 147

Enumerations

- enum caer_bias_shiftedsource_operating_mode { SHIFTED_SOURCE = 0, HI_Z = 1, TIED_TO_RAIL = 2 }
- enum caer_bias_shiftedsource_voltage_level { SPLIT_GATE = 0, SINGLE_DIODE = 1, DOUBLE_DIODE = 2 }

Functions

- struct caer_davis_info caerDavisInfoGet (caerDeviceHandle handle)
- uint16_t caerBiasVDACGenerate (const struct caer_bias_vdac vdacBias)
- struct caer_bias_vdac caerBiasVDACParse (const uint16_t vdacBias)
- uint16_t caerBiasCoarseFineGenerate (const struct caer_bias_coarsefine coarseFineBias)
- struct caer_bias_coarsefine caerBiasCoarseFineParse (const uint16_t coarseFineBias)
- uint16_t caerBiasShiftedSourceGenerate (const struct caer_bias_shiftedsource shiftedSourceBias)
- struct caer_bias_shiftedsource caerBiasShiftedSourceParse (const uint16_t shiftedSourceBias)

4.1.1 Detailed Description

DAVIS specific configuration defines and information structures.

4.1.2 Macro Definition Documentation

4.1.2.1 CAER_DEVICE_DAVIS_FX2

```
#define CAER_DEVICE_DAVIS_FX2 1
```

Device type definition for iniLabs DAVIS FX2-based boards, like DAVIS240a/b/c.

4.1.2.2 CAER_DEVICE_DAVIS_FX3

```
#define CAER_DEVICE_DAVIS_FX3 2
```

Device type definition for iniLabs DAVIS FX3-based boards, like DAVIS640.

4.1.2.3 DAVIS128_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS128_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.4 DAVIS128_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS128_CONFIG_BIAS_ADCREFHIGH 2
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.5 DAVIS128_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS128_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.6 DAVIS128_CONFIG_BIAS_AEPDBN

```
#define DAVIS128_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.7 DAVIS128_CONFIG_BIAS_AEPUXBP

```
#define DAVIS128_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.8 DAVIS128_CONFIG_BIAS_AEPUYBP

```
#define DAVIS128_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.9 DAVIS128_CONFIG_BIAS_APSCAS

```
#define DAVIS128_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.10 DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.11 DAVIS128_CONFIG_BIAS_APSROSFBN

```
#define DAVIS128_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.12 DAVIS128_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS128_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.13 DAVIS128_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS128_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.14 DAVIS128_CONFIG_BIAS_DACBUFBP

```
#define DAVIS128_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.15 DAVIS128_CONFIG_BIAS_DIFFBN

```
#define DAVIS128_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.16 DAVIS128_CONFIG_BIAS_IFREFRBN

```
#define DAVIS128_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.17 DAVIS128_CONFIG_BIAS_IFTHRBN

```
#define DAVIS128_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.18 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.19 DAVIS128_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS128_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.20 DAVIS128_CONFIG_BIAS_OFFBN

```
#define DAVIS128_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.21 DAVIS128_CONFIG_BIAS_ONBN

```
#define DAVIS128_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.22 DAVIS128_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS128_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.23 DAVIS128_CONFIG_BIAS_PIXINVBN

```
#define DAVIS128_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.24 DAVIS128_CONFIG_BIAS_PRBP

```
#define DAVIS128_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.25 DAVIS128_CONFIG_BIAS_PRSFBP

```
#define DAVIS128_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.26 DAVIS128_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS128_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.27 DAVIS128_CONFIG_BIAS_REFRBP

```
#define DAVIS128_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.28 DAVIS128_CONFIG_BIAS_SSN

```
#define DAVIS128_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.29 DAVIS128_CONFIG_BIAS_SSP

```
#define DAVIS128_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.30 DAVIS128_CONFIG_CHIP_AERNAROW

```
#define DAVIS128_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.31 DAVIS128_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.32 DAVIS128_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.33 DAVIS128_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.34 DAVIS128_CONFIG_CHIP_BIASMUX0

```
#define DAVIS128_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.35 DAVIS128_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.36 DAVIS128_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.37 DAVIS128_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.38 DAVIS128_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.39 DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.40 DAVIS128_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.41 DAVIS128_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS128_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.42 DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.43 DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.44 DAVIS128_CONFIG_CHIP_USEAOUT

```
#define DAVIS128_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.45 DAVIS208_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS208_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.46 DAVIS208_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS208_CONFIG_BIAS_ADCREFHIGH 2
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.47 DAVIS208_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS208_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.48 DAVIS208_CONFIG_BIAS_AEPDBN

```
#define DAVIS208_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.49 DAVIS208_CONFIG_BIAS_AEPUXBP

```
#define DAVIS208_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.50 DAVIS208_CONFIG_BIAS_AEPUYBP

```
#define DAVIS208_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.51 DAVIS208_CONFIG_BIAS_APSCAS

```
#define DAVIS208_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.52 DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.53 DAVIS208_CONFIG_BIAS_APSROSFBN

```
#define DAVIS208_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.54 DAVIS208_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS208_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.55 DAVIS208_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS208_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.56 DAVIS208_CONFIG_BIAS_DACBUFBP

```
#define DAVIS208_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.57 DAVIS208_CONFIG_BIAS_DIFFBN

```
#define DAVIS208_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.58 DAVIS208_CONFIG_BIAS_IFREFRBN

```
#define DAVIS208_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.59 DAVIS208_CONFIG_BIAS_IFTHRBN

```
#define DAVIS208_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.60 DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.61 DAVIS208_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS208_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.62 DAVIS208_CONFIG_BIAS_OFFBN

```
#define DAVIS208_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.63 DAVIS208_CONFIG_BIAS_ONBN

```
#define DAVIS208_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.64 DAVIS208_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS208_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.65 DAVIS208_CONFIG_BIAS_PIXINVBN

```
#define DAVIS208_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.66 DAVIS208_CONFIG_BIAS_PRBP

```
#define DAVIS208_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.67 DAVIS208_CONFIG_BIAS_PRSFBP

```
#define DAVIS208_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.68 DAVIS208_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS208_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.69 DAVIS208_CONFIG_BIAS_REFRBP

```
#define DAVIS208_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.70 DAVIS208_CONFIG_BIAS_REFSS

```
#define DAVIS208_CONFIG_BIAS_REFSS 7
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.71 DAVIS208_CONFIG_BIAS_REFSSBN

```
#define DAVIS208_CONFIG_BIAS_REFSSBN 30
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.72 DAVIS208_CONFIG_BIAS_REGBIASBP

```
#define DAVIS208_CONFIG_BIAS_REGBIASBP 28
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.73 DAVIS208_CONFIG_BIAS_RESETHIGHPASS

```
#define DAVIS208_CONFIG_BIAS_RESETHIGHPASS 6
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.74 DAVIS208_CONFIG_BIAS_SSN

```
#define DAVIS208_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.75 DAVIS208_CONFIG_BIAS_SSP

```
#define DAVIS208_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.76 DAVIS208_CONFIG_CHIP_AERNAROW

```
#define DAVIS208_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.77 DAVIS208_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.78 DAVIS208_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.79 DAVIS208_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.80 DAVIS208_CONFIG_CHIP_BIASMUX0

```
#define DAVIS208_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.81 DAVIS208_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.82 DAVIS208_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.83 DAVIS208_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.84 DAVIS208_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.85 DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.86 DAVIS208_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.87 DAVIS208_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.88 DAVIS208_CONFIG_CHIP_SELECTBIASREFSS

```
#define DAVIS208_CONFIG_CHIP_SELECTBIASREFSS 146
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.89 DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.90 DAVIS208_CONFIG_CHIP_SELECTHIGHPASS

```
#define DAVIS208_CONFIG_CHIP_SELECTHIGHPASS 149
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.91 DAVIS208_CONFIG_CHIP_SELECTPOSFB

```
#define DAVIS208_CONFIG_CHIP_SELECTPOSFB 148
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.92 DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG

```
#define DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG 145
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.93 DAVIS208_CONFIG_CHIP_SELECTSENSE

```
#define DAVIS208_CONFIG_CHIP_SELECTSENSE 147
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.94 DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.95 DAVIS208_CONFIG_CHIP_USEAOUT

```
#define DAVIS208_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.96 DAVIS240_CONFIG_BIAS_AEPDBN

```
#define DAVIS240_CONFIG_BIAS_AEPDBN 11
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.97 DAVIS240_CONFIG_BIAS_AEPUXBP

```
#define DAVIS240_CONFIG_BIAS_AEPUXBP 13
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.98 DAVIS240_CONFIG_BIAS_AEPUYBP

```
#define DAVIS240_CONFIG_BIAS_AEPUYBP 14
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.99 DAVIS240_CONFIG_BIAS_APSCASEPC

```
#define DAVIS240_CONFIG_BIAS_APSCASEPC 3
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.100 DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN

```
#define DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN 18
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.101 DAVIS240_CONFIG_BIAS_APSROSFBN

```
#define DAVIS240_CONFIG_BIAS_APSROSFBN 5
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.102 DAVIS240_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS240_CONFIG_BIAS_BIASBUFFER 19
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.103 DAVIS240_CONFIG_BIAS_DIFFBN

```
#define DAVIS240_CONFIG_BIAS_DIFFBN 0
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.104 DAVIS240_CONFIG_BIAS_DIFFCASBNC

```
#define DAVIS240_CONFIG_BIAS_DIFFCASBNC 4
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.105 DAVIS240_CONFIG_BIAS_IFREFRBN

```
#define DAVIS240_CONFIG_BIAS_IFREFRBN 16
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.106 DAVIS240_CONFIG_BIAS_IFTHRBN

```
#define DAVIS240_CONFIG_BIAS_IFTHRBN 15
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.107 DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN 12
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.108 DAVIS240_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS240_CONFIG_BIAS_LOCALBUFBN 6
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.109 DAVIS240_CONFIG_BIAS_OFFBN

```
#define DAVIS240_CONFIG_BIAS_OFFBN 2
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.110 DAVIS240_CONFIG_BIAS_ONBN

```
#define DAVIS240_CONFIG_BIAS_ONBN 1
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.111 DAVIS240_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS240_CONFIG_BIAS_PADFOLLBN 17
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.112 DAVIS240_CONFIG_BIAS_PIXINVBN

```
#define DAVIS240_CONFIG_BIAS_PIXINVBN 7
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.113 DAVIS240_CONFIG_BIAS_PRBP

```
#define DAVIS240_CONFIG_BIAS_PRBP 8
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.114 DAVIS240_CONFIG_BIAS_PRSFBP

```
#define DAVIS240_CONFIG_BIAS_PRSFBP 9
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.115 DAVIS240_CONFIG_BIAS_REFRBP

```
#define DAVIS240_CONFIG_BIAS_REFRBP 10
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.116 DAVIS240_CONFIG_BIAS_SSN

```
#define DAVIS240_CONFIG_BIAS_SSN 21
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.117 DAVIS240_CONFIG_BIAS_SSP

```
#define DAVIS240_CONFIG_BIAS_SSP 20
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.118 DAVIS240_CONFIG_CHIP_AERNAROW

```
#define DAVIS240_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.119 DAVIS240_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.120 DAVIS240_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.121 DAVIS240_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.122 DAVIS240_CONFIG_CHIP_BIASMUX0

```
#define DAVIS240_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.123 DAVIS240_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.124 DAVIS240_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.125 DAVIS240_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.126 DAVIS240_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.127 DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.128 DAVIS240_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.129 DAVIS240_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS240_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.130 DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL

```
#define DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL 139
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.131 DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.132 DAVIS240_CONFIG_CHIP_USEAOUT

```
#define DAVIS240_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.133 DAVIS346_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS346_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.134 DAVIS346_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS346_CONFIG_BIAS_ADCREFHIGH 2
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.135 DAVIS346_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS346_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.136 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.137 DAVIS346_CONFIG_BIAS_AEPDBN

```
#define DAVIS346_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.138 DAVIS346_CONFIG_BIAS_AEPUXBP

```
#define DAVIS346_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.139 DAVIS346_CONFIG_BIAS_AEPUYBP

```
#define DAVIS346_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.140 DAVIS346_CONFIG_BIAS_APSCAS

```
#define DAVIS346_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.141 DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.142 DAVIS346_CONFIG_BIAS_APSROSFBN

```
#define DAVIS346_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.143 DAVIS346_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS346_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.144 DAVIS346_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS346_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.145 DAVIS346_CONFIG_BIAS_DACBUFBP

```
#define DAVIS346_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.146 DAVIS346_CONFIG_BIAS_DIFFBN

```
#define DAVIS346_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.147 DAVIS346_CONFIG_BIAS_IFREFRBN

```
#define DAVIS346_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.148 DAVIS346_CONFIG_BIAS_IFTHRBN

```
#define DAVIS346_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.149 DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.150 DAVIS346_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS346_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.151 DAVIS346_CONFIG_BIAS_OFFBN

```
#define DAVIS346_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.152 DAVIS346_CONFIG_BIAS_ONBN

```
#define DAVIS346_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.153 DAVIS346_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS346_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.154 DAVIS346_CONFIG_BIAS_PIXINVBN

```
#define DAVIS346_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.155 DAVIS346_CONFIG_BIAS_PRBP

```
#define DAVIS346_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.156 DAVIS346_CONFIG_BIAS_PRSFBP

```
#define DAVIS346_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.157 DAVIS346_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS346_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.158 DAVIS346_CONFIG_BIAS_REFRBP

```
#define DAVIS346_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.159 DAVIS346_CONFIG_BIAS_SSN

```
#define DAVIS346_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.160 DAVIS346_CONFIG_BIAS_SSP

```
#define DAVIS346_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.161 DAVIS346_CONFIG_CHIP_AERNAROW

```
#define DAVIS346_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.162 DAVIS346_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.163 DAVIS346_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.164 DAVIS346_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.165 DAVIS346_CONFIG_CHIP_BIASMUX0

```
#define DAVIS346_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.166 DAVIS346_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.167 DAVIS346_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.168 DAVIS346_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.169 DAVIS346_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.170 DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.171 DAVIS346_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.172 DAVIS346_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS346_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.173 DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.174 DAVIS346_CONFIG_CHIP_TESTADC

```
#define DAVIS346_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.175 DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.176 DAVIS346_CONFIG_CHIP_USEAOUT

```
#define DAVIS346_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.177 DAVIS640_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS640_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.178 DAVIS640_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS640_CONFIG_BIAS_ADCREFHIGH 2
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.179 DAVIS640_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS640_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.180 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.181 DAVIS640_CONFIG_BIAS_AEPDBN

```
#define DAVIS640_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.182 DAVIS640_CONFIG_BIAS_AEPUXBP

```
#define DAVIS640_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.183 DAVIS640_CONFIG_BIAS_AEPUYBP

```
#define DAVIS640_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.184 DAVIS640_CONFIG_BIAS_APSCAS

```
#define DAVIS640_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.185 DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.186 DAVIS640_CONFIG_BIAS_APSROSFBN

```
#define DAVIS640_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.187 DAVIS640_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS640_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.188 DAVIS640_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS640_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.189 DAVIS640_CONFIG_BIAS_DACBUFBP

```
#define DAVIS640_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.190 DAVIS640_CONFIG_BIAS_DIFFBN

```
#define DAVIS640_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.191 DAVIS640_CONFIG_BIAS_IFREFRBN

```
#define DAVIS640_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.192 DAVIS640_CONFIG_BIAS_IFTHRBN

```
#define DAVIS640_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.193 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.194 DAVIS640_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS640_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.195 DAVIS640_CONFIG_BIAS_OFFBN

```
#define DAVIS640_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.196 DAVIS640_CONFIG_BIAS_ONBN

```
#define DAVIS640_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.197 DAVIS640_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS640_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.198 DAVIS640_CONFIG_BIAS_PIXINVBN

```
#define DAVIS640_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.199 DAVIS640_CONFIG_BIAS_PRBP

```
#define DAVIS640_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.200 DAVIS640_CONFIG_BIAS_PRSFBP

```
#define DAVIS640_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.201 DAVIS640_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS640_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.202 DAVIS640_CONFIG_BIAS_REFRBP

```
#define DAVIS640_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.203 DAVIS640_CONFIG_BIAS_SSN

```
#define DAVIS640_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.204 DAVIS640_CONFIG_BIAS_SSP

```
#define DAVIS640_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.205 DAVIS640_CONFIG_CHIP_AERNAROW

```
#define DAVIS640_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.206 DAVIS640_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.207 DAVIS640_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.209 DAVIS640_CONFIG_CHIP_BIASMUX0

```
#define DAVIS640_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.210 DAVIS640_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.211 DAVIS640_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.214 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.215 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.216 DAVIS640_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS640_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.217 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.218 DAVIS640_CONFIG_CHIP_TESTADC

```
#define DAVIS640_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.219 DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.220 DAVIS640_CONFIG_CHIP_USEAOUT

```
#define DAVIS640_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.221 DAVIS_CHIP_DAVIS128

```
#define DAVIS_CHIP_DAVIS128 3
```

DAVIS128 chip identifier. 128x128, color possible, internal ADC.

4.1.2.222 DAVIS_CHIP_DAVIS208

```
#define DAVIS_CHIP_DAVIS208 8
```

DAVIS208 chip identifier. 208x192, special sensitive test pixels, color possible, internal ADC.

4.1.2.223 DAVIS_CHIP_DAVIS240A

```
#define DAVIS_CHIP_DAVIS240A 0
```

DAVIS240A chip identifier. 240x180, no color, no global shutter.

4.1.2.224 DAVIS_CHIP_DAVIS240B

```
#define DAVIS_CHIP_DAVIS240B 1
```

DAVIS240B chip identifier. 240x180, no color, 50 test columns left-side.

4.1.2.225 DAVIS_CHIP_DAVIS240C

```
#define DAVIS_CHIP_DAVIS240C 2
```

DAVIS240C chip identifier. 240x180, no color.

4.1.2.226 DAVIS_CHIP_DAVIS346A

```
#define DAVIS_CHIP_DAVIS346A 4
```

DAVIS346A chip identifier. 346x260, color possible, internal ADC.

4.1.2.227 DAVIS_CHIP_DAVIS346B

```
#define DAVIS_CHIP_DAVIS346B 5
```

DAVIS346B chip identifier. 346x260, color possible, internal ADC.

4.1.2.228 DAVIS_CHIP_DAVIS346C

```
#define DAVIS_CHIP_DAVIS346C 9
```

DAVIS346C chip identifier. 346x260, BSI, color possible, internal ADC.

4.1.2.229 DAVIS_CHIP_DAVIS640

```
#define DAVIS_CHIP_DAVIS640 6
```

DAVIS640 chip identifier. 640x480, color possible, internal ADC.

4.1.2.230 DAVIS_CHIP_DAVISRGB

```
#define DAVIS_CHIP_DAVISRGB 7
```

DAVISRGB chip identifier. 640x480 APS, 320x240 DVS, color possible, internal ADC.

4.1.2.231 DAVIS_CONFIG_APS

```
#define DAVIS_CONFIG_APS 2
```

Module address: device-side APS (Frame) configuration. The APS (Active-Pixel-Sensor) is responsible for getting the normal, synchronous frame from the camera chip. It supports various options for very precise timing control, as well as Region of Interest imaging.

4.1.2.232 DAVIS_CONFIG_APS_ADC_TEST_MODE

```
#define DAVIS_CONFIG_APS_ADC_TEST_MODE 39
```

Parameter address for module DAVIS_CONFIG_APS: put all APS pixels into reset, while keeping everything else running. This is only useful for testing and characterizing the internal ADC, to minimize noise.

4.1.2.233 DAVIS_CONFIG_APS_COLOR_FILTER

```
#define DAVIS_CONFIG_APS_COLOR_FILTER 3
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains information on the type of color filter present on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper color filter information.

4.1.2.234 DAVIS_CONFIG_APS_COLUMN_SETTLE

```
#define DAVIS_CONFIG_APS_COLUMN_SETTLE 16
```

Parameter address for module DAVIS_CONFIG_APS: column settle time in ADCClock cycles.

4.1.2.235 DAVIS_CONFIG_APS_END_COLUMN_0

```
#define DAVIS_CONFIG_APS_END_COLUMN_0 11
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 0. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_0.

4.1.2.236 DAVIS_CONFIG_APS_END_COLUMN_1

```
#define DAVIS_CONFIG_APS_END_COLUMN_1 22
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 1. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_1.

4.1.2.237 DAVIS_CONFIG_APS_END_COLUMN_2

```
#define DAVIS_CONFIG_APS_END_COLUMN_2 26
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 2. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_2.

4.1.2.238 DAVIS_CONFIG_APS_END_COLUMN_3

```
#define DAVIS_CONFIG_APS_END_COLUMN_3 30
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 3. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_3.

4.1.2.239 DAVIS_CONFIG_APS_END_ROW_0

```
#define DAVIS_CONFIG_APS_END_ROW_0 12
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 0. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_0.

4.1.2.240 DAVIS_CONFIG_APS_END_ROW_1

```
#define DAVIS_CONFIG_APS_END_ROW_1 23
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 1. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_1.

4.1.2.241 DAVIS_CONFIG_APS_END_ROW_2

```
#define DAVIS_CONFIG_APS_END_ROW_2 27
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 2. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_2.

4.1.2.242 DAVIS_CONFIG_APS_END_ROW_3

```
#define DAVIS_CONFIG_APS_END_ROW_3 31
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 3. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_3.

4.1.2.243 DAVIS_CONFIG_APS_EXPOSURE

```
#define DAVIS_CONFIG_APS_EXPOSURE 13
```

Parameter address for module DAVIS_CONFIG_APS: frame exposure time in microseconds, up to about one second maximum. Very precise for Global Shutter, slightly less exact for Rolling Shutter due to column-based timing constraints.

4.1.2.244 DAVIS_CONFIG_APS_FRAME_DELAY

```
#define DAVIS_CONFIG_APS_FRAME_DELAY 14
```

Parameter address for module DAVIS_CONFIG_APS: delay between consecutive frames in microseconds, up to about one second maximum. This can be used to achieve slower frame-rates, down to about 1 Hertz.

4.1.2.245 DAVIS_CONFIG_APS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_GLOBAL_SHUTTER 8
```

Parameter address for module DAVIS_CONFIG_APS: enable Global Shutter mode instead of Rolling Shutter. The Global Shutter eliminates motion artifacts, but is noisier than the Rolling Shutter (worse quality).

4.1.2.246 DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC

```
#define DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC 32
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of an external ADC to read the pixel values. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.247 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 7
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of the global shutter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.248 DAVIS_CONFIG_APS_HAS_INTERNAL_ADC

```
#define DAVIS_CONFIG_APS_HAS_INTERNAL_ADC 33
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of an internal, on-chip ADC to read the pixel values. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.249 DAVIS_CONFIG_APS_HAS_QUAD_ROI

```
#define DAVIS_CONFIG_APS_HAS_QUAD_ROI 19
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of the Quadruple Region of Interest feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.250 DAVIS_CONFIG_APS_NULL_SETTLE

```
#define DAVIS_CONFIG_APS_NULL_SETTLE 18
```

Parameter address for module DAVIS_CONFIG_APS: null (between states) settle time in ADCClock cycles.

4.1.2.251 DAVIS_CONFIG_APS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_APS_ORIENTATION_INFO 2
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains information on the orientation of the X/Y axes, whether they should be inverted or not on the host when parsing incoming pixels, as well as if the X or Y axes need to be flipped when reading the pixels. Bit 2: apsInvertXY Bit 1: apsFlipX Bit 0: apsFlipY This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_↩ davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.252 DAVIS_CONFIG_APS_RAMP_RESET

```
#define DAVIS_CONFIG_APS_RAMP_RESET 37
```

Parameter address for module DAVIS_CONFIG_APS: ramp reset time in ADCClock cycles.

4.1.2.253 DAVIS_CONFIG_APS_RAMP_SHORT_RESET

```
#define DAVIS_CONFIG_APS_RAMP_SHORT_RESET 38
```

Parameter address for module DAVIS_CONFIG_APS: only perform a short ramp (half length) during reset reads, given that the voltage should always be close to the top of the range. This increases the frame-rate, but may have impacts on image quality, especially in very bright regions.

4.1.2.254 DAVIS_CONFIG_APS_RESET_READ

```
#define DAVIS_CONFIG_APS_RESET_READ 5
```

Parameter address for module DAVIS_CONFIG_APS: enable the reset read phase in addition to the signal read, to allow for correlated double sampling schemes. This heavily improves image quality and should always be turned on. In special cases, especially when the camera is perfectly stationary, this can be turned off for longer periods of time to achieve a higher frame-rate and significantly faster frame capture.

4.1.2.255 DAVIS_CONFIG_APS_RESET_SETTLE

```
#define DAVIS_CONFIG_APS_RESET_SETTLE 15
```

Parameter address for module DAVIS_CONFIG_APS: column reset settle time in ADCClock cycles.

4.1.2.256 DAVIS_CONFIG_APS_ROW_SETTLE

```
#define DAVIS_CONFIG_APS_ROW_SETTLE 17
```

Parameter address for module DAVIS_CONFIG_APS: row settle time in ADCClock cycles.

4.1.2.257 DAVIS_CONFIG_APS_RUN

```
#define DAVIS_CONFIG_APS_RUN 4
```

Parameter address for module DAVIS_CONFIG_APS: enable the APS module and take intensity images of the scene. While this parameter is enabled, frames will be taken continuously. To slow down the frame-rate, see DAVIS_CONFIG_APS_FRAME_DELAY. To only take snapshots, see DAVIS_CONFIG_APS_SNAPSHOT.

4.1.2.258 DAVIS_CONFIG_APS_SAMPLE_ENABLE

```
#define DAVIS_CONFIG_APS_SAMPLE_ENABLE 35
```

Parameter address for module DAVIS_CONFIG_APS: enable sampling of pixel voltage by the internal ADC circuitry. Must always be enabled to get proper frame values.

4.1.2.259 DAVIS_CONFIG_APS_SAMPLE_SETTLE

```
#define DAVIS_CONFIG_APS_SAMPLE_SETTLE 36
```

Parameter address for module DAVIS_CONFIG_APS: sample settle time in ADCClock cycles.

4.1.2.260 DAVIS_CONFIG_APS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_APS_SIZE_COLUMNS 0
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains the X axis resolution of the APS frames returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.261 DAVIS_CONFIG_APS_SIZE_ROWS

```
#define DAVIS_CONFIG_APS_SIZE_ROWS 1
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains the Y axis resolution of the APS frames returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.262 DAVIS_CONFIG_APS_SNAPSHOT

```
#define DAVIS_CONFIG_APS_SNAPSHOT 80
```

Parameter address for module DAVIS_CONFIG_APS: takes a snapshot (one frame), like a photo-camera. More efficient implementation that just toggling the DAVIS_CONFIG_APS_RUN parameter. The APS module should not be running prior to calling this, as it only makes sense if frames are not being generated at the time. Also, DAVI↩ S_CONFIG_APS_FRAME_DELAY should be set to zero if only doing snapshots, to ensure a quicker readiness for the next one, since the delay is always observed after taking a frame.

4.1.2.263 DAVIS_CONFIG_APS_START_COLUMN_0

```
#define DAVIS_CONFIG_APS_START_COLUMN_0 9
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 0. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_0 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.264 DAVIS_CONFIG_APS_START_COLUMN_1

```
#define DAVIS_CONFIG_APS_START_COLUMN_1 20
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 1. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_1 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.265 DAVIS_CONFIG_APS_START_COLUMN_2

```
#define DAVIS_CONFIG_APS_START_COLUMN_2 24
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 2. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_2 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.266 DAVIS_CONFIG_APS_START_COLUMN_3

```
#define DAVIS_CONFIG_APS_START_COLUMN_3 28
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 3. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_3 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.267 DAVIS_CONFIG_APS_START_ROW_0

```
#define DAVIS_CONFIG_APS_START_ROW_0 10
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 0. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_0.

4.1.2.268 DAVIS_CONFIG_APS_START_ROW_1

```
#define DAVIS_CONFIG_APS_START_ROW_1 21
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 1. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_1.

4.1.2.269 DAVIS_CONFIG_APS_START_ROW_2

```
#define DAVIS_CONFIG_APS_START_ROW_2 25
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 2. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_2.

4.1.2.270 DAVIS_CONFIG_APS_START_ROW_3

```
#define DAVIS_CONFIG_APS_START_ROW_3 29
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 3. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_3.

4.1.2.271 DAVIS_CONFIG_APS_USE_INTERNAL_ADC

```
#define DAVIS_CONFIG_APS_USE_INTERNAL_ADC 34
```

Parameter address for module DAVIS_CONFIG_APS: use the internal, on-chip ADC instead of the external one. This enables a much faster and more power-efficient readout for the frames, and should as such always be preferred.

4.1.2.272 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL 6
```

Parameter address for module DAVIS_CONFIG_APS: if the output FIFO for this module is full, stall the APS state machine and wait until it's free again, instead of just dropping the pixels as they are being read out. This guarantees a complete frame readout, at the possible cost of slight timing differences between pixels. If disabled, incomplete frames may be transmitted and will then be dropped on the host, resulting in lower frame-rates, especially during high DVS traffic.

4.1.2.273 DAVIS_CONFIG_BIAS

```
#define DAVIS_CONFIG_BIAS 5
```

Module address: device-side chip bias configuration. Shared with DAVIS_CONFIG_CHIP. This state machine is responsible for configuring the chip's bias generator.

4.1.2.274 DAVIS_CONFIG_CHIP

```
#define DAVIS_CONFIG_CHIP 5
```

Module address: device-side chip control configuration. Shared with DAVIS_CONFIG_BIAS. This state machine is responsible for configuring the chip's internal control shift registers, to set special options.

4.1.2.275 DAVIS_CONFIG_DVS

```
#define DAVIS_CONFIG_DVS 1
```

Module address: device-side DVS configuration. The DVS state machine handshakes with the chip's AER bus and gets the polarity events from it. It supports various configurable delays, as well as advanced filtering capabilities on the polarity events.

4.1.2.276 DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN

```
#define DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN 5
```

Parameter address for module DAVIS_CONFIG_DVS: delay capturing the data and acknowledging it on the AER bus for the column events (serial AER protocol) by this many LogicClock cycles.

4.1.2.277 DAVIS_CONFIG_DVS_ACK_DELAY_ROW

```
#define DAVIS_CONFIG_DVS_ACK_DELAY_ROW 4
```

Parameter address for module DAVIS_CONFIG_DVS: delay capturing the data and acknowledging it on the AER bus for the row events (serial AER protocol) by this many LogicClock cycles.

4.1.2.278 DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN

```
#define DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN 7
```

Parameter address for module DAVIS_CONFIG_DVS: extend the length of the acknowledge on the AER bus for the column events (serial AER protocol) by this many LogicClock cycles.

4.1.2.279 DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW

```
#define DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW 6
```

Parameter address for module DAVIS_CONFIG_DVS: extend the length of the acknowledge on the AER bus for the row events (serial AER protocol) by this many LogicClock cycles.

4.1.2.280 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL

```
#define DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 10
```

Parameter address for module DAVIS_CONFIG_DVS: enable external AER control. This ensures the chip and the DVS pixel array are running, but doesn't do the handshake and leaves the ACK pin in high-impedance, to allow for an external system to take over the AER communication with the chip. DAVIS_CONFIG_DVS_RUN has to be turned off for this to work.

4.1.2.281 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 29
```

Parameter address for module DAVIS_CONFIG_DVS: enable the background-activity filter, which tries to remove events caused by transistor leakage, by rejecting uncorrelated events.

4.1.2.282 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT 30
```

Parameter address for module DAVIS_CONFIG_DVS: specify the time difference constant for the background-activity filter in microseconds. Events that do correlated within this time-frame are let through, while others are filtered out.

4.1.2.283 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 13
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 0, X axis setting.

4.1.2.284 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 12
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 0, Y axis setting.

4.1.2.285 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 15
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 1, X axis setting.

4.1.2.286 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 14
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 1, Y axis setting.

4.1.2.287 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 17
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 2, X axis setting.

4.1.2.288 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 16
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 2, Y axis setting.

4.1.2.289 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 19
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 3, X axis setting.

4.1.2.290 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 18
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 3, Y axis setting.

4.1.2.291 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 21
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 4, X axis setting.

4.1.2.292 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 20
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 4, Y axis setting.

4.1.2.293 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 23
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 5, X axis setting.

4.1.2.294 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 22
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 5, Y axis setting.

4.1.2.295 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 25
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 6, X axis setting.

4.1.2.296 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 24
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 6, Y axis setting.

4.1.2.297 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 27
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 7, X axis setting.

4.1.2.298 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 26
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 7, Y axis setting.

4.1.2.299 DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS

```
#define DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS 9
```

Parameter address for module DAVIS_CONFIG_DVS: enable row-only event filter, to eliminate spurious row events with no following columns events. This can happen on DAVIS240 chips, or following the various pixel and background-activity filtering stages, which drop column events to achieve their effect. This should always be enabled!

4.1.2.300 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 28
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the background-activity filter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.301 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 11
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the pixel filter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.302 DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR

```
#define DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR 31
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the test event generator feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.303 DAVIS_CONFIG_DVS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_DVS_ORIENTATION_INFO 2
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains information on the orientation of the X/Y axes, whether they should be inverted or not on the host when parsing incoming events. Bit 2: dvsInvert↩ XY Bit 1: reserved Bit 0: reserved This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.304 DAVIS_CONFIG_DVS_RUN

```
#define DAVIS_CONFIG_DVS_RUN 3
```

Parameter address for module DAVIS_CONFIG_DVS: run the DVS state machine and get polarity events from the chip by handshaking with its AER bus.

4.1.2.305 DAVIS_CONFIG_DVS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_DVS_SIZE_COLUMNS 0
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains the X axis resolution of the DVS events returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.306 DAVIS_CONFIG_DVS_SIZE_ROWS

```
#define DAVIS_CONFIG_DVS_SIZE_ROWS 1
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains the Y axis resolution of the DVS events returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.307 DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE

```
#define DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE 32
```

Parameter address for module DAVIS_CONFIG_DVS: enable the test event generator for debugging purposes. This generates fake events that appear to originate from all rows sequentially, and for each row going through all its columns, first with an ON polarity and then with an OFF polarity. Both DAVIS_CONFIG_DVS_RUN and DAVIS_↩ CONFIG_DVS_EXTERNAL_AER_CONTROL have to be turned off for this to work.

4.1.2.308 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL 8
```

Parameter address for module DAVIS_CONFIG_DVS: if the output FIFO for this module is full, stall the AER handshake with the chip and wait until it's free again, instead of just continuing the handshake and dropping the resulting events.

4.1.2.309 DAVIS_CONFIG_EXTINPUT

```
#define DAVIS_CONFIG_EXTINPUT 4
```

Module address: device-side External Input (signal detector/generator) configuration. The External Input module is used to detect external signals on the external input jack and inject an event into the event stream when this happens. It can detect pulses of a specific length or rising and falling edges. On some systems, a signal generator module is also present, which can generate PWM-like pulsed signals with configurable timing.

4.1.2.310 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES 2
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_FALLING_EDGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.311 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1 17
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_FALLING_E↩ DGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.312 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2 23
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_FALLING_E↩ DGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.313 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH 5
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct caer_davis_info' for details on how to get the frequency).

4.1.2.314 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1 20
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct caer_davis_info' for details on how to get the frequency).

4.1.2.315 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2 26
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct caer_davis_info' for details on how to get the frequency).

4.1.2.316 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY 4
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.317 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1 19
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.318 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2 25
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.319 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES 3
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT↩ _DETECT_PULSE_POLARITY and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH for more details.

4.1.2.320 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1 18
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT_↩ DETECT_PULSE_POLARITY1 and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1 for more details.

4.1.2.321 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2 24
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT_↩ DETECT_PULSE_POLARITY2 and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2 for more details.

4.1.2.322 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES 1
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.323 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1 16
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.324 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2 22
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.325 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE 13
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enables event injection when a falling edge occurs in the generated signal; a special event EXTERNAL_GENERATOR_FALLING_EDGE is emitted into the event stream.

4.1.2.326 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 12
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enables event injection when a rising edge occurs in the generated signal; a special event EXTERNAL_GENERATOR_RISING_EDGE is emitted into the event stream.

4.1.2.327 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 10
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the interval between the start of two consecutive pulses, expressed in cycles at LogicClock frequency (see 'struct caer_davis_info' for details on how to get the frequency). This must be bigger or equal to DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH. To generate a signal with 50% duty cycle, this would have to be exactly double of DAVIS_CONFIG_EXTINPUT_GENE↩ RATE_PULSE_LENGTH.

4.1.2.328 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 11
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the length a pulse stays active, expressed in cycles at LogicClock frequency (see 'struct caer_davis_info' for details on how to get the frequency). This must be smaller or equal to DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL. To generate a signal with 50% duty cycle, this would have to be exactly half of DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL.

4.1.2.329 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 9
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: polarity of the PWM-like signal to be generated. '1' means active high, '0' means active low.

4.1.2.330 DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL 8
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: instead of generating a PWM-like signal by using the configured parameters, use a signal on the FPGA/CPLD that's passed as an input to the External Input module. By default this is disabled and tied to ground, but it can be useful for customized logic designs.

4.1.2.331 DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS

```
#define DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS 14
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: read-only parameter, information about the presence of the extra detectors feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.332 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR 6
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: read-only parameter, information about the presence of the signal generator feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.333 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR 0
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the IN JACK signal. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.334 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1 15
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the B1P20 input pin. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.335 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2 21
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the B1P21 input pin. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.336 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 7
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal generator module. It generates a PWM-like signal based on configurable parameters and outputs it on the OUT JACK signal.

4.1.2.337 DAVIS_CONFIG_IMU

```
#define DAVIS_CONFIG_IMU 3
```

Module address: device-side IMU (Inertial Measurement Unit) configuration. The IMU module connects to the external IMU chip and sends data on the device's movement in space. It can configure various options on the external chip, such as accelerometer range or gyroscope refresh rate.

4.1.2.338 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 8
```

Parameter address for module DAVIS_CONFIG_IMU: select the full scale range of the accelerometer outputs. Valid values are: 0 - +- 2 g 1 - +- 4 g 2 - +- 8 g 3 - +- 16 g

4.1.2.339 DAVIS_CONFIG_IMU_ACCEL_STANDBY

```
#define DAVIS_CONFIG_IMU_ACCEL_STANDBY 2
```

Parameter address for module DAVIS_CONFIG_IMU: put the accelerometer sensor in standby, disabling it.

4.1.2.340 DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER

```
#define DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER 7
```

Parameter address for module DAVIS_CONFIG_IMU: this configures the digital low-pass filter for both the accelerometer and the gyroscope. Valid values are from 0 to 7 and have the following meaning: 0 - Accel: BW=260Hz, Delay=0ms, FS=1kHz - Gyro: BW=256Hz, Delay=0.98ms, FS=8kHz 1 - Accel: BW=184Hz, Delay=2.0ms, FS=1k↩ Hz - Gyro: BW=188Hz, Delay=1.9ms, FS=1kHz 2 - Accel: BW=94Hz, Delay=3.0ms, FS=1kHz - Gyro: BW=98Hz, Delay=2.8ms, FS=1kHz 3 - Accel: BW=44Hz, Delay=4.9ms, FS=1kHz - Gyro: BW=42Hz, Delay=4.8ms, FS=1k↩ Hz 4 - Accel: BW=21Hz, Delay=8.5ms, FS=1kHz - Gyro: BW=20Hz, Delay=8.3ms, FS=1kHz 5 - Accel: BW=10Hz, Delay=13.8ms, FS=1kHz - Gyro: BW=10Hz, Delay=13.4ms, FS=1kHz 6 - Accel: BW=5Hz, Delay=19.0ms, FS=1k↩ Hz - Gyro: BW=5Hz, Delay=18.6ms, FS=1kHz 7 - Accel: RESERVED, FS=1kHz - Gyro: RESERVED, FS=8kHz

4.1.2.341 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 9
```

Parameter address for module DAVIS_CONFIG_IMU: select the full scale range of the gyroscope outputs. Valid values are: 0 - +- 250 °/s 1 - +- 500 °/s 2 - +- 1000 °/s 3 - +- 2000 °/s

4.1.2.342 DAVIS_CONFIG_IMU_GYRO_STANDBY

```
#define DAVIS_CONFIG_IMU_GYRO_STANDBY 3
```

Parameter address for module DAVIS_CONFIG_IMU: put the gyroscope sensor in standby, disabling it.

4.1.2.343 DAVIS_CONFIG_IMU_LP_CYCLE

```
#define DAVIS_CONFIG_IMU_LP_CYCLE 4
```

Parameter address for module DAVIS_CONFIG_IMU: put the IMU into Cycle Mode. In Cycle Mode, the device cycles between sleep mode and waking up to take a single sample of data from the accelerometer at a rate determined by DAVIS_CONFIG_IMU_LP_WAKEUP.

4.1.2.344 DAVIS_CONFIG_IMU_LP_WAKEUP

```
#define DAVIS_CONFIG_IMU_LP_WAKEUP 5
```

Parameter address for module DAVIS_CONFIG_IMU: rate at which the IMU takes an accelerometer sample while in Cycle Mode (see DAVIS_CONFIG_IMU_LP_CYCLE). Valid values are: 0 - 1.25 Hz wake-up frequency 1 - 5 Hz wake-up frequency 2 - 20 Hz wake-up frequency 3 - 40 Hz wake-up frequency

4.1.2.345 DAVIS_CONFIG_IMU_ORIENTATION_INFO

```
#define DAVIS_CONFIG_IMU_ORIENTATION_INFO 10
```

Parameter address for module DAVIS_CONFIG_IMU: read-only parameter, contains information on the orientation of the X/Y/Z axes, whether they should be flipped or not on the host when parsing incoming IMU data samples. Bit 2: imuFlipX Bit 1: imuFlipY Bit 0: imuFlipZ This is reserved for internal use and should not be used by anything other than libcaer. Generated IMU events are already properly flipped when returned to the user.

4.1.2.346 DAVIS_CONFIG_IMU_RUN

```
#define DAVIS_CONFIG_IMU_RUN 0
```

Parameter address for module DAVIS_CONFIG_IMU: run the IMU state machine to get information about the movement and position of the device. This takes the IMU chip out of sleep.

4.1.2.347 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER

```
#define DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 6
```

Parameter address for module DAVIS_CONFIG_IMU: this specifies the divider from the Gyroscope Output Rate used to generate the Sample Rate for the IMU. Valid values are from 0 to 255. The Sample Rate is generated like this: Sample Rate = Gyroscope Output Rate / (1 + DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER) where Gyroscope Output Rate = 8 kHz when DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER is disabled (set to 0 or 7), and 1 kHz when enabled. Note: the accelerometer output rate is 1 kHz. This means that for a Sample Rate greater than 1 kHz, the same accelerometer sample may be output multiple times.

4.1.2.348 DAVIS_CONFIG_IMU_TEMP_STANDBY

```
#define DAVIS_CONFIG_IMU_TEMP_STANDBY 1
```

Parameter address for module DAVIS_CONFIG_IMU: put the temperature sensor in standby, disabling it.

4.1.2.349 DAVIS_CONFIG_MICROPHONE

```
#define DAVIS_CONFIG_MICROPHONE 7
```

Module address: device-side microphone configuration. The Microphone module enables the use of InvenSense stereo microphones to capture samples of sound from devices that support is, such as the miniDAVIS346.

4.1.2.350 DAVIS_CONFIG_MICROPHONE_RUN

```
#define DAVIS_CONFIG_MICROPHONE_RUN 0
```

Parameter address for module DAVIS_CONFIG_MICROPHONE: enable the Microphone module, which provides stereo samples of sound recorded by on-board InvenSense microphones.

4.1.2.351 DAVIS_CONFIG_MICROPHONE_SAMPLE_FREQUENCY

```
#define DAVIS_CONFIG_MICROPHONE_SAMPLE_FREQUENCY 1
```

Parameter address for module DAVIS_CONFIG_MICROPHONE: allows setting the sample frequency of the stereo microphones, by specifying the length of an SCK clock cycle in LogicClock cycles. Value can be between 30 and 215 inclusive. The desired value can be calculated in the following way: floor(100'000'000/64/<desired freq="" in="" hz>="">) For example for 48 KHz sampling frequency, this would be 32. For 44.1 KHz it would be 35, and for 16 KHz it would be 97.

4.1.2.352 DAVIS_CONFIG_MUX

```
#define DAVIS_CONFIG_MUX 0
```

Module address: device-side Multiplexer configuration. The Multiplexer is responsible for mixing, timestamping and outputting (via USB) the various event types generated by the device. It is also responsible for timestamp generation and synchronization.

4.1.2.353 DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL 5
```

Parameter address for module DAVIS_CONFIG_MUX: drop APS events if the USB output FIFO is full, instead of having them pile up at the input FIFOs. This normally should not be enabled to guarantee complete, coherent frame events, though small timing differences may cause a reduction in observed image quality.

4.1.2.354 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 4
```

Parameter address for module DAVIS_CONFIG_MUX: drop DVS events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.1.2.355 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 7
```

Parameter address for module DAVIS_CONFIG_MUX: drop External Input events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.1.2.356 DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL 6
```

Parameter address for module DAVIS_CONFIG_MUX: drop IMU events if the USB output FIFO is full, instead of having them pile up at the input FIFOs. This normally should not be enabled to guarantee complete, coherent IMU events, and not get incomplete or wrong IMU information.

4.1.2.357 DAVIS_CONFIG_MUX_DROP_MIC_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_MIC_ON_TRANSFER_STALL 8
```

Parameter address for module DAVIS_CONFIG_MUX: drop Microphone sample events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.1.2.358 DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE

```
#define DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
```

Parameter address for module DAVIS_CONFIG_MUX: under normal circumstances, the chip's bias generator is only powered up when either the DVS or the APS state machines are running, to save power. This flag forces the bias generator to be powered up all the time, which may be useful when one wants to shut-down both APS and DVS temporarily, but still have a quick and well-defined resume behavior.

4.1.2.359 DAVIS_CONFIG_MUX_RUN

```
#define DAVIS_CONFIG_MUX_RUN 0
```

Parameter address for module DAVIS_CONFIG_MUX: run the Multiplexer state machine, which is responsible for mixing the various event types at the device level, timestamping them and outputting them via USB or other connectors.

4.1.2.360 DAVIS_CONFIG_MUX_TIMESTAMP_RESET

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RESET 2
```

Parameter address for module DAVIS_CONFIG_MUX: reset the Timestamp Generator to zero. This also sends a reset pulse to all connected slave devices, resetting their timestamp too.

4.1.2.361 DAVIS_CONFIG_MUX_TIMESTAMP_RUN

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RUN 1
```

Parameter address for module DAVIS_CONFIG_MUX: run the Timestamp Generator inside the Multiplexer state machine, which will provide microsecond accurate timestamps to the events passing through.

4.1.2.362 DAVIS_CONFIG_SYSINFO

```
#define DAVIS_CONFIG_SYSINFO 6
```

Module address: device-side system information. The system information module provides various details on the device, such as currently installed logic revision or clock speeds. All its parameters are read-only. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation for more details on what information is available.

4.1.2.363 DAVIS_CONFIG_SYSINFO_ADC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_ADC_CLOCK 4
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the FPGA/CPLD logic related to APS frame grabbing is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.364 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, an integer used to identify the different types of sensor chips used on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.365 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, whether the device is currently a timestamp master or slave when synchronizing multiple devices together. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.366 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the main FPGA/CPLD logic is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.367 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_VERSION 0
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the version of the logic currently running on the device's FPGA/CPLD. It usually represents a specific SVN revision, at which the logic code was synthesized. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_davis_info' documentation to get this information.

4.1.2.368 DAVIS_CONFIG_USB

```
#define DAVIS_CONFIG_USB 9
```

Module address: device-side USB output configuration. The USB output module forwards the data from the device and the FPGA/CPLD to the USB chip, usually a Cypress FX2 or FX3.

4.1.2.369 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 1
```

Parameter address for module DAVIS_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 125µs time-slices, corresponding to how USB schedules its operations (a value of 4 for example would mean waiting at most 0.5ms until sending a short USB packet to the host).

4.1.2.370 DAVIS_CONFIG_USB_RUN

```
#define DAVIS_CONFIG_USB_RUN 0
```

Parameter address for module DAVIS_CONFIG_USB: enable the USB FIFO module, which transfers the data from the FPGA/CPLD to the USB chip, to be then sent to the host. Turning this off will suppress any USB data communication!

4.1.2.371 DAVISRGB_CONFIG_APS_GSFDRESET

```
#define DAVISRGB_CONFIG_APS_GSFDRESET 55
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter FD reset time in ADCClock cycles.

4.1.2.372 DAVISRGB_CONFIG_APS_GSPDRESET

```
#define DAVISRGB_CONFIG_APS_GSPDRESET 52
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter PD reset time in ADCClock cycles.

4.1.2.373 DAVISRGB_CONFIG_APS_GSRESETFALL

```
#define DAVISRGB_CONFIG_APS_GSRESETFALL 53
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter Reset Fall time in ADCClock cycles.

4.1.2.374 DAVISRGB_CONFIG_APS_GSTXFALL

```
#define DAVISRGB_CONFIG_APS_GSTXFALL 54
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter Transfer Fall time in ADCClock cycles.

4.1.2.375 DAVISRGB_CONFIG_APS_RSFDSETTLE

```
#define DAVISRGB_CONFIG_APS_RSFDSETTLE 51
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Rolling Shutter FD settle time in ADCClock cycles.

4.1.2.376 DAVISRGB_CONFIG_APS_TRANSFER

```
#define DAVISRGB_CONFIG_APS_TRANSFER 50
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): charge transfer time in ADCClock cycles.

4.1.2.377 DAVISRGB_CONFIG_BIAS_ADCCOMPBP

```
#define DAVISRGB_CONFIG_BIAS_ADCCOMPBP 27
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.378 DAVISRGB_CONFIG_BIAS_ADCREFHIGH

```
#define DAVISRGB_CONFIG_BIAS_ADCREFHIGH 6
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.379 DAVISRGB_CONFIG_BIAS_ADCREFLOW

```
#define DAVISRGB_CONFIG_BIAS_ADCREFLOW 7
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.380 DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE 5
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.381 DAVISRGB_CONFIG_BIAS_AEPDBN

```
#define DAVISRGB_CONFIG_BIAS_AEPDBN 31
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.382 DAVISRGB_CONFIG_BIAS_AEPUXBP

```
#define DAVISRGB_CONFIG_BIAS_AEPUXBP 32
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.383 DAVISRGB_CONFIG_BIAS_AEPUYBP

```
#define DAVISRGB_CONFIG_BIAS_AEPUYBP 33
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.384 DAVISRGB_CONFIG_BIAS_APSCAS

```
#define DAVISRGB_CONFIG_BIAS_APSCAS 0
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.385 DAVISRGB_CONFIG_BIAS_APSROSFBN

```
#define DAVISRGB_CONFIG_BIAS_APSROSFBN 26
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.386 DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN

```
#define DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN 20
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.387 DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN

```
#define DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN 22
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.388 DAVISRGB_CONFIG_BIAS_BIASBUFFER

```
#define DAVISRGB_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.389 DAVISRGB_CONFIG_BIAS_DACBUFBP

```
#define DAVISRGB_CONFIG_BIAS_DACBUFBP 28
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.390 DAVISRGB_CONFIG_BIAS_DIFFBN

```
#define DAVISRGB_CONFIG_BIAS_DIFFBN 14
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.391 DAVISRGB_CONFIG_BIAS_FALLTIMEBN

```
#define DAVISRGB_CONFIG_BIAS_FALLTIMEBN 23
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.392 DAVISRGB_CONFIG_BIAS_GND07

```
#define DAVISRGB_CONFIG_BIAS_GND07 4
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.393 DAVISRGB_CONFIG_BIAS_IFREFRBN

```
#define DAVISRGB_CONFIG_BIAS_IFREFRBN 8
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.394 DAVISRGB_CONFIG_BIAS_IFTHRBN

```
#define DAVISRGB_CONFIG_BIAS_IFTHRBN 9
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.395 DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN 30
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.396 DAVISRGB_CONFIG_BIAS_LOCALBUFBN

```
#define DAVISRGB_CONFIG_BIAS_LOCALBUFBN 10
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.397 DAVISRGB_CONFIG_BIAS_OFFBN

```
#define DAVISRGB_CONFIG_BIAS_OFFBN 16
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.398 DAVISRGB_CONFIG_BIAS_ONBN

```
#define DAVISRGB_CONFIG_BIAS_ONBN 15
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.399 DAVISRGB_CONFIG_BIAS_OVG1LO

```
#define DAVISRGB_CONFIG_BIAS_OVG1LO 1
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.400 DAVISRGB_CONFIG_BIAS_OVG2LO

```
#define DAVISRGB_CONFIG_BIAS_OVG2LO 2
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.401 DAVISRGB_CONFIG_BIAS_PADFOLLBN

```
#define DAVISRGB_CONFIG_BIAS_PADFOLLBN 11
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.402 DAVISRGB_CONFIG_BIAS_PIXINVBN

```
#define DAVISRGB_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.403 DAVISRGB_CONFIG_BIAS_PRBP

```
#define DAVISRGB_CONFIG_BIAS_PRBP 17
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.404 DAVISRGB_CONFIG_BIAS_PRSFBP

```
#define DAVISRGB_CONFIG_BIAS_PRSFBP 18
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.405 DAVISRGB_CONFIG_BIAS_READOUTBUFBP

```
#define DAVISRGB_CONFIG_BIAS_READOUTBUFBP 25
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.406 DAVISRGB_CONFIG_BIAS_REFRBP

```
#define DAVISRGB_CONFIG_BIAS_REFRBP 19
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.407 DAVISRGB_CONFIG_BIAS_RISETIMEBP

```
#define DAVISRGB_CONFIG_BIAS_RISETIMEBP 24
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.408 DAVISRGB_CONFIG_BIAS_SSN

```
#define DAVISRGB_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.409 DAVISRGB_CONFIG_BIAS_SSP

```
#define DAVISRGB_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.410 DAVISRGB_CONFIG_BIAS_TX2OVG2HI

```
#define DAVISRGB_CONFIG_BIAS_TX2OVG2HI 3
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- caerBiasVDACGenerate() for VDAC (voltage) biases.
- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See 'http://inilabs.com/support/biasing/' for more details.

4.1.2.411 DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO 145
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.412 DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO 146
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.413 DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI 147
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.414 DAVISRGB_CONFIG_CHIP_AERNAROW

```
#define DAVISRGB_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.415 DAVISRGB_CONFIG_CHIP_ANALOGMUX0

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.416 DAVISRGB_CONFIG_CHIP_ANALOGMUX1

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.417 DAVISRGB_CONFIG_CHIP_ANALOGMUX2

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.418 DAVISRGB_CONFIG_CHIP_BIASMUX0

```
#define DAVISRGB_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.419 DAVISRGB_CONFIG_CHIP_DIGITALMUX0

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.420 DAVISRGB_CONFIG_CHIP_DIGITALMUX1

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.421 DAVISRGB_CONFIG_CHIP_DIGITALMUX2

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.422 DAVISRGB_CONFIG_CHIP_DIGITALMUX3

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.423 DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.424 DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.425 DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.426 DAVISRGB_CONFIG_CHIP_TESTADC

```
#define DAVISRGB_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.427 DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.428 DAVISRGB_CONFIG_CHIP_USEAOUT

```
#define DAVISRGB_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.429 IS_DAVIS128

```
#define IS_DAVIS128(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS128)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.430 IS_DAVIS208

```
#define IS_DAVIS208(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS208)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.431 IS_DAVIS240

```
#define IS_DAVIS240(
              chipID ) (IS_DAVIS240A(chipID) || IS_DAVIS240B(chipID) || IS_DAVIS240C(chipID))
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.432 IS_DAVIS240A

```
#define IS_DAVIS240A(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS240A)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.433 IS_DAVIS240B

```
#define IS_DAVIS240B(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS240B)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.434 IS_DAVIS240C

```
#define IS_DAVIS240C(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS240C)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.435 IS_DAVIS346

```
#define IS_DAVIS346(
              chipID ) (IS_DAVIS346A(chipID) || IS_DAVIS346B(chipID) || IS_DAVIS346C(chipID))
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.436 IS_DAVIS346A

```
#define IS_DAVIS346A(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS346A)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.437 IS_DAVIS346B

```
#define IS_DAVIS346B(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS346B)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.438 IS_DAVIS346C

```
#define IS_DAVIS346C(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS346C)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.439 IS_DAVIS640

```
#define IS_DAVIS640(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS640)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.440 IS_DAVISRGB

```
#define IS_DAVISRGB(
              chipID ) ((chipID) == DAVIS_CHIP_DAVISRGB)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.3 Enumeration Type Documentation

4.1.3.1 caer_bias_shiftedsource_operating_mode

```
enum caer_bias_shiftedsource_operating_mode
```

Shifted-source bias operating mode.

Enumerator

SHIFTED_SOURCE	Standard mode.
HI_Z	High impedance (driven from outside).
TIED_TO_RAIL	Tied to ground (SSN) or VDD (SSP).

4.1.3.2 caer_bias_shiftedsource_voltage_level

```
enum caer_bias_shiftedsource_voltage_level
```

Shifted-source bias voltage level.

Enumerator

SPLIT_GATE	Standard mode (200-400mV).
SINGLE_DIODE	Higher shifted-source voltage (one cascode).
DOUBLE_DIODE	Even higher shifted-source voltage (two cascodes).

4.1.4 Function Documentation

4.1.4.1 caerBiasCoarseFineGenerate()

```
uint16_t caerBiasCoarseFineGenerate (
            const struct caer_bias_coarsefine coarseFineBias )
```

Transform coarse-fine bias structure into internal integer representation, suited for sending directly to the device via caerDeviceConfigSet().

Parameters

coarseFineBias	coarse-fine bias structure.

Returns

internal integer representation for device configuration.

4.1.4.2 caerBiasCoarseFineParse()

```
struct caer_bias_coarsefine caerBiasCoarseFineParse (
            const uint16_t coarseFineBias )
```

Transform internal integer representation, as received by calls to caerDeviceConfigGet(), into a coarse-fine bias structure, for easier handling and understanding of the various parameters.

Parameters

coarseFineBias	internal integer representation from device.

Returns

coarse-fine bias structure.

4.1.4.3 caerBiasShiftedSourceGenerate()

```
uint16_t caerBiasShiftedSourceGenerate (
            const struct caer_bias_shiftedsource shiftedSourceBias )
```

Transform shifted-source bias structure into internal integer representation, suited for sending directly to the device via caerDeviceConfigSet().

Parameters

shiftedSourceBias	shifted-source bias structure.

Returns

internal integer representation for device configuration.

4.1.4.4 caerBiasShiftedSourceParse()

```
struct caer_bias_shiftedsource caerBiasShiftedSourceParse (
            const uint16_t shiftedSourceBias )
```

Transform internal integer representation, as received by calls to caerDeviceConfigGet(), into a shifted-source bias structure, for easier handling and understanding of the various parameters.

Parameters

shiftedSourceBias	internal integer representation from device.

Returns

shifted-source bias structure.

4.1.4.5 caerBiasVDACGenerate()

```
uint16_t caerBiasVDACGenerate (
            const struct caer_bias_vdac vdacBias )
```

Transform VDAC bias structure into internal integer representation, suited for sending directly to the device via caerDeviceConfigSet().

Parameters

vdacBias	VDAC bias structure.

Returns

internal integer representation for device configuration.

4.1.4.6 caerBiasVDACParse()

```
struct caer_bias_vdac caerBiasVDACParse (
            const uint16_t vdacBias )
```

Transform internal integer representation, as received by calls to caerDeviceConfigGet(), into a VDAC bias structure, for easier handling and understanding of the various parameters.

Parameters

vdacBias	internal integer representation from device.

Returns

VDAC bias structure.

4.1.4.7 caerDavisInfoGet()

```
struct caer_davis_info caerDavisInfoGet (
            caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct caer_davis_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.2 devices/dvs128.h File Reference

```
#include "usb.h"
#include "../events/polarity.h"
#include "../events/special.h"
```

Data Structures

- struct caer_dvs128_info

Macros

- #define CAER_DEVICE_DVS128 0
- #define DVS128_CONFIG_DVS 0
- #define DVS128_CONFIG_BIAS 1
- #define DVS128_CONFIG_DVS_RUN 0
- #define DVS128_CONFIG_DVS_TIMESTAMP_RESET 1
- #define DVS128_CONFIG_DVS_ARRAY_RESET 2
- #define DVS128_CONFIG_DVS_TS_MASTER 3
- #define DVS128_CONFIG_BIAS_CAS 0
- #define DVS128_CONFIG_BIAS_INJGND 1
- #define DVS128_CONFIG_BIAS_REQPD 2
- #define DVS128_CONFIG_BIAS_PUX 3
- #define DVS128_CONFIG_BIAS_DIFFOFF 4
- #define DVS128_CONFIG_BIAS_REQ 5
- #define DVS128_CONFIG_BIAS_REFR 6
- #define DVS128_CONFIG_BIAS_PUY 7
- #define DVS128_CONFIG_BIAS_DIFFON 8
- #define DVS128_CONFIG_BIAS_DIFF 9
- #define DVS128_CONFIG_BIAS_FOLL 10
- #define DVS128_CONFIG_BIAS_PR 11

Functions

- struct caer_dvs128_info caerDVS128InfoGet (caerDeviceHandle handle)

4.2.1 Detailed Description

DVS128 specific configuration defines and information structures.

4.2.2 Macro Definition Documentation

4.2.2.1 CAER_DEVICE_DVS128

```
#define CAER_DEVICE_DVS128 0
```

Device type definition for iniLabs DVS128.

4.2.2.2 DVS128_CONFIG_BIAS

```
#define DVS128_CONFIG_BIAS 1
```

Module address: device-side chip bias generator configuration.

4.2.2.3 DVS128_CONFIG_BIAS_CAS

```
#define DVS128_CONFIG_BIAS_CAS 0
```

Parameter address for module DVS128_CONFIG_BIAS: First stage amplifier cascode bias. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.4 DVS128_CONFIG_BIAS_DIFF

```
#define DVS128_CONFIG_BIAS_DIFF 9
```

Parameter address for module DVS128_CONFIG_BIAS: Differential (second stage amplifier) bias. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.5 DVS128_CONFIG_BIAS_DIFFOFF

```
#define DVS128_CONFIG_BIAS_DIFFOFF 4
```

Parameter address for module DVS128_CONFIG_BIAS: Off events threshold bias. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.6 DVS128_CONFIG_BIAS_DIFFON

```
#define DVS128_CONFIG_BIAS_DIFFON 8
```

Parameter address for module DVS128_CONFIG_BIAS: On events threshold bias. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.7 DVS128_CONFIG_BIAS_FOLL

```
#define DVS128_CONFIG_BIAS_FOLL 10
```

Parameter address for module DVS128_CONFIG_BIAS: Source follower bias. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.8 DVS128_CONFIG_BIAS_INJGND

```
#define DVS128_CONFIG_BIAS_INJGND 1
```

Parameter address for module DVS128_CONFIG_BIAS: Injected ground bias. See 'http://inilabs.↩com/support/biasing/' for more details.

4.2.2.9 DVS128_CONFIG_BIAS_PR

```
#define DVS128_CONFIG_BIAS_PR 11
```

Parameter address for module DVS128_CONFIG_BIAS: Photoreceptor bias. See 'http://inilabs.↩com/support/biasing/' for more details.

4.2.2.10 DVS128_CONFIG_BIAS_PUX

```
#define DVS128_CONFIG_BIAS_PUX 3
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from X arbiter (AER). See 'http↩://inilabs.com/support/biasing/' for more details.

4.2.2.11 DVS128_CONFIG_BIAS_PUY

```
#define DVS128_CONFIG_BIAS_PUY 7
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See 'http↩://inilabs.com/support/biasing/' for more details.

4.2.2.12 DVS128_CONFIG_BIAS_REFR

```
#define DVS128_CONFIG_BIAS_REFR 6
```

Parameter address for module DVS128_CONFIG_BIAS: Refractory period bias. See 'http://inilabs.↩com/support/biasing/' for more details.

4.2.2.13 DVS128_CONFIG_BIAS_REQ

```
#define DVS128_CONFIG_BIAS_REQ 5
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down for passive load inverters in digital AER pixel circuitry. See 'http://inilabs.com/support/biasing/' for more details.

4.2.2.14 DVS128_CONFIG_BIAS_REQPD

```
#define DVS128_CONFIG_BIAS_REQPD 2
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down on chip request (AER). See 'http↩://inilabs.com/support/biasing/' for more details.

4.2.2.15 DVS128_CONFIG_DVS

```
#define DVS128_CONFIG_DVS 0
```

Module address: device-side DVS configuration.

4.2.2.16 DVS128_CONFIG_DVS_ARRAY_RESET

```
#define DVS128_CONFIG_DVS_ARRAY_RESET 2
```

Parameter address for module DVS128_CONFIG_DVS: reset the whole DVS pixel array. This is a temporary configuration switch and will reset itself right away.

4.2.2.17 DVS128_CONFIG_DVS_RUN

```
#define DVS128_CONFIG_DVS_RUN 0
```

Parameter address for module DVS128_CONFIG_DVS: run the DVS chip and generate polarity event data.

4.2.2.18 DVS128_CONFIG_DVS_TIMESTAMP_RESET

```
#define DVS128_CONFIG_DVS_TIMESTAMP_RESET 1
```

Parameter address for module DVS128_CONFIG_DVS: reset the time-stamp counter of the device. This is a temporary configuration switch and will reset itself right away.

4.2.2.19 DVS128_CONFIG_DVS_TS_MASTER

```
#define DVS128_CONFIG_DVS_TS_MASTER 3
```

Parameter address for module DVS128_CONFIG_DVS: control if this DVS is a timestamp master device. Default is enabled.

4.2.3 Function Documentation

4.2.3.1 caerDVS128InfoGet()

```
struct caer_dvs128_info caerDVS128InfoGet (
            caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct caer_dvs128_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.3 devices/dynapse.h File Reference

```
#include "usb.h"
#include "../events/spike.h"
#include "../events/special.h"
```

Data Structures

- struct caer_dynapse_info
- struct caer_bias_dynapse

Macros

- #define CAER_DEVICE_DYNAPSE 3
- #define DYNAPSE_CHIP_DYNAPSE 64
- #define DYNAPSE_CONFIG_MUX 0
- #define DYNAPSE_CONFIG_AER 1
- #define DYNAPSE_CONFIG_CHIP 5
- #define DYNAPSE_CONFIG_SYSINFO 6
- #define DYNAPSE_CONFIG_USB 9
- #define DYNAPSE_CONFIG_CLEAR_CAM 10
- #define DYNAPSE_CONFIG_DEFAULT_SRAM 11
- #define DYNAPSE_CONFIG_MONITOR_NEU 12
- #define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 13
- #define DYNAPSE_CONFIG_SRAM 14
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG 15
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 0
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL 1
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 2
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 3
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 4
- #define DYNAPSE_CONFIG_SRAM_ADDRESS 1
- #define DYNAPSE_CONFIG_SRAM_READDATA 2
- #define DYNAPSE_CONFIG_SRAM_WRITEDATA 3
- #define DYNAPSE_CONFIG_SRAM_RWCOMMAND 4
- #define DYNAPSE_CONFIG_SRAM_WRITE 1
- #define DYNAPSE_CONFIG_SRAM_READ 0
- #define DYNAPSE_CONFIG_MUX_RUN 0
- #define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 1
- #define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 2
- #define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3

- #define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 4
- #define DYNAPSE_CONFIG_AER_RUN 3
- #define DYNAPSE_CONFIG_AER_ACK_DELAY 4
- #define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
- #define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
- #define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
- #define DYNAPSE_CONFIG_CHIP_RUN 0
- #define DYNAPSE_CONFIG_CHIP_ID 1
- #define DYNAPSE_CONFIG_CHIP_CONTENT 2
- #define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
- #define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
- #define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
- #define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
- #define DYNAPSE_CONFIG_USB_RUN 0
- #define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_NEG** 1
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH** 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH** 1
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST** 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST** 1
- #define DYNAPSE_X4BOARD_NEUX 64
- #define DYNAPSE_X4BOARD_NEUY 64
- #define DYNAPSE_X4BOARD_COREX 4
- #define DYNAPSE_X4BOARD_COREY 4
- #define **DYNAPSE_CONFIG_DYNAPSE_U0** 0
- #define **DYNAPSE_CONFIG_DYNAPSE_U1** 8
- #define **DYNAPSE_CONFIG_DYNAPSE_U2** 4
- #define **DYNAPSE_CONFIG_DYNAPSE_U3** 12
- #define **DYNAPSE_CONFIG_NUMNEURONS** 1024
- #define **DYNAPSE_CONFIG_SRAMROW** 1024
- #define **DYNAPSE_CONFIG_CAMCOL** 16
- #define **DYNAPSE_CONFIG_NUMNEURONS_CORE** 256
- #define **DYNAPSE_CONFIG_NUMCORES** 4
- #define **DYNAPSE_CONFIG_NUMSRAM_NEU** 4
- #define **DYNAPSE_CONFIG_XCHIPSIZE** 32
- #define **DYNAPSE_CONFIG_YCHIPSIZE** 32
- #define **DYNAPSE_CONFIG_NEUROW** 16
- #define **DYNAPSE_CONFIG_NEUCOL** 16
- #define **DYNAPSE_CONFIG_NUMCAM** 64
- #define **DYNAPSE_CONFIG_CAMTYPE_F_EXC** 3
- #define **DYNAPSE_CONFIG_CAMTYPE_S_EXC** 2
- #define **DYNAPSE_CONFIG_CAMTYPE_F_INH** 1
- #define **DYNAPSE_CONFIG_CAMTYPE_S_INH** 0
- #define **DYNAPSE_MAX_USER_USB_PACKET_SIZE** 1024
- #define **DYNAPSE_CONFIG_MAX_USB_TRANSFER** 512
- #define **DYNAPSE_CONFIG_MAX_PARAM_SIZE** 85
- #define DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 0
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_S_N** 2
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_F_N** 4
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_S_N** 6
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_F_N** 8
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_RFR_N** 10

- #define **DYNAPSE_CONFIG_BIAS_C0_IF_TAU1_N** 12
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHTAU_N** 14
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_CASC_N** 16
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_TAU2_N** 18
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_BUF_P** 20
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHTHR_N** 22
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_THR_N** 24
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_S_P** 26
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_F_P** 28
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_F_P** 30
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_S_P** 32
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_NMDA_N** 34
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_DC_P** 36
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHW_P** 38
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_S_P** 40
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_F_P** 42
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_F_P** 44
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_S_P** 46
- #define **DYNAPSE_CONFIG_BIAS_C0_R2R_P** 48
- #define **DYNAPSE_CONFIG_BIAS_C1_PULSE_PWLK_P** 1
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_S_N** 3
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_F_N** 5
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_S_N** 7
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_F_N** 9
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_RFR_N** 11
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_TAU1_N** 13
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHTAU_N** 15
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_CASC_N** 17
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_TAU2_N** 19
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_BUF_P** 21
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHTHR_N** 23
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_THR_N** 25
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_S_P** 27
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_F_P** 29
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_F_P** 31
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_S_P** 33
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_NMDA_N** 35
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_DC_P** 37
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHW_P** 39
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_S_P** 41
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_F_P** 43
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_F_P** 45
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_S_P** 47
- #define **DYNAPSE_CONFIG_BIAS_C1_R2R_P** 49
- #define **DYNAPSE_CONFIG_BIAS_U_BUFFER** 50
- #define **DYNAPSE_CONFIG_BIAS_U_SSP** 51
- #define **DYNAPSE_CONFIG_BIAS_U_SSN** 52
- #define **DYNAPSE_CONFIG_BIAS_C2_PULSE_PWLK_P** 64
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_S_N** 66
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_F_N** 68
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_S_N** 70
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_F_N** 72
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_RFR_N** 74
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_TAU1_N** 76
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHTAU_N** 78

- #define **DYNAPSE_CONFIG_BIAS_C2_IF_CASC_N** 80
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_TAU2_N** 82
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_BUF_P** 84
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHTHR_N** 86
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_THR_N** 88
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_S_P** 90
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_F_P** 92
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_F_P** 94
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_S_P** 96
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_NMDA_N** 98
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_DC_P** 100
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHW_P** 102
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_S_P** 104
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_F_P** 106
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_F_P** 108
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_S_P** 110
- #define **DYNAPSE_CONFIG_BIAS_C2_R2R_P** 112
- #define **DYNAPSE_CONFIG_BIAS_C3_PULSE_PWLK_P** 65
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_S_N** 67
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_F_N** 69
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_S_N** 71
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_F_N** 73
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_RFR_N** 75
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_TAU1_N** 77
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHTAU_N** 79
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_CASC_N** 81
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_TAU2_N** 83
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_BUF_P** 85
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHTHR_N** 87
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_THR_N** 89
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_S_P** 91
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_F_P** 93
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_F_P** 95
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_S_P** 97
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_NMDA_N** 99
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_DC_P** 101
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHW_P** 103
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_S_P** 105
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_F_P** 107
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_F_P** 109
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_S_P** 111
- #define **DYNAPSE_CONFIG_BIAS_C3_R2R_P** 113
- #define **DYNAPSE_CONFIG_BIAS_D_BUFFER** 114
- #define **DYNAPSE_CONFIG_BIAS_D_SSP** 115
- #define **DYNAPSE_CONFIG_BIAS_D_SSN** 116

Functions

- struct caer_dynapse_info caerDynapseInfoGet (caerDeviceHandle handle)
- bool **caerDynapseWriteSramWords** (caerDeviceHandle handle, const uint16_t *data, uint32_t baseAddr, uint32_t numWords)
- bool **caerDynapseWriteSram** (caerDeviceHandle handle, uint16_t coreId, uint32_t neuronId, uint16_t virtualCoreId, bool sx, uint8_t dx, bool sy, uint8_t dy, uint16_t sramId, uint16_t destinationCore)
- bool **caerDynapseSendDataToUSB** (caerDeviceHandle handle, const uint32_t *data, int numConfig)
- bool **caerDynapseWriteCam** (caerDeviceHandle handle, uint32_t preNeuronAddr, uint32_t postNeuronAddr, uint32_t camId, int16_t synapseType)
- uint32_t **caerDynapseGenerateCamBits** (uint32_t preNeuronAddr, uint32_t postNeuronAddr, uint32_t camId, int16_t synapseType)

4.3.1 Detailed Description

Dynap-se specific configuration defines and information structures.

4.3.2 Macro Definition Documentation

4.3.2.1 CAER_DEVICE_DYNAPSE

```
#define CAER_DEVICE_DYNAPSE 3
```

Device type definition for iniLabs Dynap-se FX2-based boards.

4.3.2.2 DYNAPSE_CHIP_DYNAPSE

```
#define DYNAPSE_CHIP_DYNAPSE 64
```

Dynap-se chip identifier.

4.3.2.3 DYNAPSE_CONFIG_AER

```
#define DYNAPSE_CONFIG_AER 1
```

Module address: device-side AER configuration (from chip). The AER state machine handshakes with the chip's AER bus and gets the spike events from it. It supports various configurable delays.

4.3.2.4 DYNAPSE_CONFIG_AER_ACK_DELAY

```
#define DYNAPSE_CONFIG_AER_ACK_DELAY 4
```

Parameter address for module DYNAPSE_CONFIG_AER: delay capturing the data and acknowledging it on the AER bus for the events by this many LogicClock cycles.

4.3.2.5 DYNAPSE_CONFIG_AER_ACK_EXTENSION

```
#define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
```

Parameter address for module DYNAPSE_CONFIG_AER: extend the length of the acknowledge on the AER bus for the events by this many LogicClock cycles.

4.3.2.6 DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL

```
#define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
```

Parameter address for module DYNAPSE_CONFIG_AER: enable external AER control. This ensures the chip and the neuron array are running, but doesn't do the handshake and leaves the ACK pin in high-impedance, to allow for an external system to take over the AER communication with the chip. DYNAPSE_CONFIG_AER_RUN has to be turned off for this to work.

4.3.2.7 DYNAPSE_CONFIG_AER_RUN

```
#define DYNAPSE_CONFIG_AER_RUN 3
```

Parameter address for module DYNAPSE_CONFIG_AER: run the AER state machine and get spike events from the chip by handshaking with its AER bus.

4.3.2.8 DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
```

Parameter address for module DYNAPSE_CONFIG_AER: if the output FIFO for this module is full, stall the AER handshake with the chip and wait until it's free again, instead of just continuing the handshake and dropping the resulting events.

4.3.2.9 DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P

```
#define DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 0
```

Parameter address for module DYNAPSE_CONFIG_BIAS: DYNAPSE chip biases. Bias configuration values must be generated using the proper functions, which are:

- convertBias() for coarse-fine (current) biases. See 'http://inilabs.com/support/biasing/' for more details.

4.3.2.10 DYNAPSE_CONFIG_CHIP

```
#define DYNAPSE_CONFIG_CHIP 5
```

Module address: device-side chip control configuration. This state machine is responsible for configuring the chip's internal control registers, to set special options and biases.

4.3.2.11 DYNAPSE_CONFIG_CHIP_CONTENT

```
#define DYNAPSE_CONFIG_CHIP_CONTENT 2
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the configuration content to send to the chip. Every time this changes, the chip ID is appended and the configuration is sent out to the chip.

4.3.2.12 DYNAPSE_CONFIG_CHIP_ID

```
#define DYNAPSE_CONFIG_CHIP_ID 1
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the chip ID to which configuration content is being sent.

4.3.2.13 DYNAPSE_CONFIG_CHIP_REQ_DELAY

```
#define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
```

Parameter address for module DYNAPSE_CONFIG_CHIP: delay doing the request after putting out the data by this many LogicClock cycles.

4.3.2.14 DYNAPSE_CONFIG_CHIP_REQ_EXTENSION

```
#define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
```

Parameter address for module DYNAPSE_CONFIG_CHIP: extend the request after receiving the ACK by this many LogicClock cycles.

4.3.2.15 DYNAPSE_CONFIG_CHIP_RUN

```
#define DYNAPSE_CONFIG_CHIP_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_CHIP: enable the configuration AER state machine to send bias and control configuration to the chip.

4.3.2.16 DYNAPSE_CONFIG_CLEAR_CAM

```
#define DYNAPSE_CONFIG_CLEAR_CAM 10
```

Clear CAM content Output USB data packets in streams of 512 bytes using libusb es: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_CLEAR_CAM, 0, 0); //0,0 not used

4.3.2.17 DYNAPSE_CONFIG_DEFAULT_SRAM

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM 11
```

Clear SRAM content, use one SRAM cell to monitor neurons Output USB data packets in streams of 512 bytes using libusb es: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_DEFAULT_SRAM, DYNAPSE_CONFIG_DYNAPSE_U2, 0); // zero not used

4.3.2.18 DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 13
```

Clear SRAM content, route nothing outside Output USB data packets in streams of 512 bytes using libusb es: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_DEFAULT_SRAM, DYNAPSE_CONFIG_DYNAPSE_U2, 0); // zero not used

4.3.2.19 DYNAPSE_CONFIG_MONITOR_NEU

```
#define DYNAPSE_CONFIG_MONITOR_NEU 12
```

Used to monitor neurons , example usage: es: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG↩ _MONITOR_NEU, 1, 0); // core 1 neuron 0

4.3.2.20 DYNAPSE_CONFIG_MUX

```
#define DYNAPSE_CONFIG_MUX 0
```

Module address: device-side Multiplexer configuration. The Multiplexer is responsible for mixing, timestamping and outputting (via USB) the various event types generated by the device. It is also responsible for timestamp generation.

4.3.2.21 DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 4
```

Parameter address for module DYNAPSE_CONFIG_MUX: drop AER events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.3.2.22 DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE

```
#define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
```

Parameter address for module DYNAPSE_CONFIG_MUX: under normal circumstances, the chip's bias generator is only powered up when either the AER or the configuration state machines are running, to save power. This flag forces the bias generator to be powered up all the time.

4.3.2.23 DYNAPSE_CONFIG_MUX_RUN

```
#define DYNAPSE_CONFIG_MUX_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_MUX: run the Multiplexer state machine, which is responsible for mixing the various event types at the device level, timestamping them and outputting them via USB or other connectors.

4.3.2.24 DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 2
```

Parameter address for module DYNAPSE_CONFIG_MUX: reset the Timestamp Generator to zero. This also sends a reset pulse to all connected slave devices, resetting their timestamp too.

4.3.2.25 DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 1
```

Parameter address for module DYNAPSE_CONFIG_MUX: run the Timestamp Generator inside the Multiplexer state machine, which will provide microsecond accurate timestamps to the events passing through.

4.3.2.26 DYNAPSE_CONFIG_SRAM

```
#define DYNAPSE_CONFIG_SRAM 14
```

Module address: device side SRAM controller configuration. The module supports holds an address, a word to be written to SRAM the most recent word read using a read command, and a read/write command. Reads/writes are triggered when the address field is changed ex: caerDynapseWriteSRAM(moduleData->moduleState, SRAMData, baseAddr, numWords); Writes numWords words from array SRAMData to the SRAM, starting at baseAddr.

4.3.2.27 DYNAPSE_CONFIG_SRAM_ADDRESS

```
#define DYNAPSE_CONFIG_SRAM_ADDRESS 1
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the address that will be used for the next read/write. Writing or reading this field will trigger the command contained in the command register to be executed.

4.3.2.28 DYNAPSE_CONFIG_SRAM_DIRECTION_POS

```
#define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
```

Parameter address for module DYNAPSE_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 125µs time-slices, corresponding to how USB schedules its operations (a value of 4 for example would mean waiting at most 0.5ms until sending a short USB packet to the host).

4.3.2.29 DYNAPSE_CONFIG_SRAM_READ

```
#define DYNAPSE_CONFIG_SRAM_READ 0
```

Command for module DYNAPSE_CONFIG_SRAM: Read command for the RWCOMMAND field. ex: caerConfig↩Set(moduleData->moduleState, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, D↩YNAPSE_CONFIG_SRAM_READ); Sets the SRAM controller up for doing reads.

4.3.2.30 DYNAPSE_CONFIG_SRAM_READDATA

```
#define DYNAPSE_CONFIG_SRAM_READDATA 2
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the most recently read data from the SRAM. Read only parameter.

4.3.2.31 DYNAPSE_CONFIG_SRAM_RWCOMMAND

```
#define DYNAPSE_CONFIG_SRAM_RWCOMMAND 4
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the command that will be executed when the address field is written to. ex: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_SRAM, DYN↩APSE_CONFIG_SRAM_RWCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes.

4.3.2.32 DYNAPSE_CONFIG_SRAM_WRITE

```
#define DYNAPSE_CONFIG_SRAM_WRITE 1
```

Command for module DYNAPSE_CONFIG_SRAM: Write command for the RWCOMMAND field. ex: caerConfig↩Set(moduleData->moduleState, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, D↩YNAPSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes.

4.3.2.33 DYNAPSE_CONFIG_SRAM_WRITEDATA

```
#define DYNAPSE_CONFIG_SRAM_WRITEDATA 3
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the data that will be written on the next write. ex: caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_WRI↩TEDATA, wData); caerConfigSet(moduleData->moduleState, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFI↩G_SRAM_RWCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); caerConfigSet(moduleData->moduleState, D↩YNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_ADDRESS, wAddr); Writes wData to the address specified by wAddr.

4.3.2.34 DYNAPSE_CONFIG_SYNAPSERECONFIG

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG 15
```

Module address: Device side Synapse Reconfiguration module configuration. Provides run control, selection between using a single kernel for all neurons and reading per-neuron kernels from SRAM, programming of the global kernel, as well as target output chip ID selection and SRAM kernel table base address.

4.3.2.35 DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 3
```

Parameter address for moudle DYNAPSE_CONFIG_SYNAPSERECONFIG Output chip select using chip identifiers from this document

4.3.2.36 DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL 1
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG Bits 16 downto 12 select the address in the global kernel table and bits 11 downto 0 specify the data. The 12 data bits are split into 4∗3 synaptic weight bits which map onto positive/negative polarity events from 2 DVS pixels.

4.3.2.37 DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Run control. Starts and stops handshaking with DVS.

4.3.2.38 DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 4
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG SRAM base address configuration in increments of 32 Kib. Setting this to N will place the SRAM kernel LUT in the range [N∗2^15,(N+1)∗2^15-1]

4.3.2.39 DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 2
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG Boolean parameter for selecting between using kernels stored in SRAM or the global kernel table. 1 for SRAM, 0 for global kernel table

4.3.2.40 DYNAPSE_CONFIG_SYSINFO

```
#define DYNAPSE_CONFIG_SYSINFO 6
```

Module address: device-side system information. The system information module provides various details on the device, such as currently installed logic revision or clock speeds. All its parameters are read-only. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_dynapse_info' documentation for more details on what information is available.

4.3.2.41 DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, an integer used to identify the different types of sensor chips used on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_dynapse_info' documentation to get this information.

4.3.2.42 DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, whether the device is currently a timestamp master or slave when synchronizing multiple devices together. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_dynapse_info' documentation to get this information.

4.3.2.43 DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the main FPGA/CPLD logic is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_dynapse_info' documentation to get this information.

4.3.2.44 DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, the version of the logic currently running on the device's FPGA/CPLD. It usually represents a specific SVN revision, at which the logic code was synthesized. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct caer_dynapse_info' documentation to get this information.

4.3.2.45 DYNAPSE_CONFIG_USB

```
#define DYNAPSE_CONFIG_USB 9
```

Module address: device-side USB output configuration. The USB output module forwards the data from the device and the FPGA/CPLD to the USB chip, usually a Cypress FX2 or FX3.

4.3.2.46 DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
```

Parameter address for module DYNAPSE_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 125µs time-slices, corresponding to how USB schedules its operations (a value of 4 for example would mean waiting at most 0.5ms until sending a short USB packet to the host).

4.3.2.47 DYNAPSE_CONFIG_USB_RUN

```
#define DYNAPSE_CONFIG_USB_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_USB: enable the USB FIFO module, which transfers the data from the FPGA/CPLD to the USB chip, to be then sent to the host. Turning this off will suppress any USB data communication!

4.3.2.48 DYNAPSE_X4BOARD_COREX

```
#define DYNAPSE_X4BOARD_COREX 4
```

Parameter address for module DYNAPSE_X4BOARD_COREX: Number of cores in the x direction of the board

4.3.2.49 DYNAPSE_X4BOARD_COREY

```
#define DYNAPSE_X4BOARD_COREY 4
```

Parameter address for module DYNAPSE_X4BOARD_COREY: Number of cores in the x direction of the board

4.3.2.50 DYNAPSE_X4BOARD_NEUX

```
#define DYNAPSE_X4BOARD_NEUX 64
```

Parameter address for module DYNAPSE_X4BOARD_NEUX: Number of neurons in the x direction of the board

4.3.2.51 DYNAPSE_X4BOARD_NEUY

```
#define DYNAPSE_X4BOARD_NEUY 64
```

Parameter address for module DYNAPSE_X4BOARD_NEUY: Number of neurons in the x direction of the board

4.3.3 Function Documentation

4.3.3.1 caerDynapseInfoGet()

```
struct caer_dynapse_info caerDynapseInfoGet (
            caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, the logic version, and so on. See the 'struct caer_dynapse↩_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.4 devices/usb.h File Reference

```
#include "../libcaer.h"
#include "../events/packetContainer.h"
```

Macros

- #define CAER_HOST_CONFIG_USB -1
- #define CAER_HOST_CONFIG_DATAEXCHANGE -2
- #define CAER_HOST_CONFIG_PACKETS -3
- #define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0
- #define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1
- #define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0
- #define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1
- #define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2
- #define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3
- #define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0
- #define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1

Typedefs

- typedef struct caer_device_handle * caerDeviceHandle

Functions

- caerDeviceHandle caerDeviceOpen (uint16_t deviceID, uint16_t deviceType, uint8_t busNumberRestrict, uint8_t devAddressRestrict, const char *serialNumberRestrict)
- bool caerDeviceClose (caerDeviceHandle *handle)
- bool caerDeviceSendDefaultConfig (caerDeviceHandle handle)
- bool caerDeviceConfigSet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t param)
- bool caerDeviceConfigGet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t *param)
- bool caerDeviceDataStart (caerDeviceHandle handle, void(*dataNotifyIncrease)(void *ptr), void(*dataNotifyDecrease)(void *ptr), void *dataNotifyUserPtr, void(*dataShutdownNotify)(void *ptr), void *dataShutdownUserPtr)
- bool caerDeviceDataStop (caerDeviceHandle handle)
- caerEventPacketContainer caerDeviceDataGet (caerDeviceHandle handle)

4.4.1 Detailed Description

Common functions to access, configure and exchange data with supported USB devices. Also contains defines for host/USB related configuration options.

4.4.2 Macro Definition Documentation

4.4.2.1 CAER_HOST_CONFIG_DATAEXCHANGE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE -2
```

Module address: host-side data exchange (ring-buffer) configuration.

4.4.2.2 CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: when calling caerDeviceDataGet(), the function can either be blocking, meaning it waits until it has a valid EventPacketContainer to return, or not, meaning it returns right away. This behavior can be set with this flag. Please see the caerDeviceDataGet() documentation for more information on its return values.

4.4.2.3 CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: set size of elements that can be held by the thread-safe FIFO buffer between the USB data transfer thread and the main thread. The default values are usually fine, only change them if you're running into lots of dropped/missing packets; you can turn on the INFO log level to see when this is the case.

4.4.2.4 CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to start all the data producer modules on the device (DVS, APS, Mux, ...) automatically when starting the USB data transfer thread with caer↩ DeviceDataStart() or not. If disabled, be aware you will have to start the right modules manually, which can be useful if you need precise control over which ones are running at any time.

4.4.2.5 CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to stop all the data producer modules on the device (DVS, APS, Mux, ...) automatically when stopping the USB data transfer thread with caer↩ DeviceDataStop() or not. If disabled, be aware you will have to stop the right modules manually, to halt the data flow, which can be useful if you need precise control over which ones are running at any time.

4.4.2.6 CAER_HOST_CONFIG_PACKETS

```
#define CAER_HOST_CONFIG_PACKETS -3
```

Module address: host-side event packets generation configuration.

4.4.2.7 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the time interval between subsequent packet containers. The value is in microseconds, and is checked across all types of events contained in the Event↩ PacketContainer.

4.4.2.8 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the maximum number of events any of a packet container's packets may hold before it's made available to the user. This is checked for each number of events held in each typed EventPacket that is a part of the EventPacketContainer.

4.4.2.9 CAER_HOST_CONFIG_USB

```
#define CAER_HOST_CONFIG_USB -1
```

Module address: host-side USB configuration.

4.4.2.10 CAER_HOST_CONFIG_USB_BUFFER_NUMBER

```
#define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0
```

Parameter address for module CAER_HOST_CONFIG_USB: set number of buffers used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.4.2.11 CAER_HOST_CONFIG_USB_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1
```

Parameter address for module CAER_HOST_CONFIG_USB: set size of each buffer used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.4.3 Typedef Documentation

4.4.3.1 caerDeviceHandle

```
typedef struct caer_device_handle* caerDeviceHandle
```

Reference to an open device on which to operate.

4.4.4 Function Documentation

4.4.4.1 caerDeviceClose()

```
bool caerDeviceClose (
            caerDeviceHandle * handle )
```

Close a previously opened USB device and invalidate its handle.

Parameters

handle	pointer to a valid device handle. Will set handle to NULL if closing is successful, to prevent further usage of this handle for other operations.

Returns

true if closing was successful, false on errors.

4.4.4.2 caerDeviceConfigGet()

```
bool caerDeviceConfigGet (
              caerDeviceHandle handle,
              int8_t modAddr,
              uint8_t paramAddr,
              uint32_t * param )
```

Get the value of a configuration parameter.

Parameters

handle	a valid device handle.
modAddr	a module address, used to specify which configuration module one wants to query. Negative addresses are used for host-side configuration, while positive addresses (including zero) are used for device-side configuration.
paramAddr	a parameter address, to select a specific parameter to query from this particular configuration module. Only positive numbers (including zero) are allowed.
param	a pointer to an integer, in which to store the configuration parameter's current value. The integer will always be either set to zero (on failure), or to the current value (on success).

Returns

true if sending the configuration was successful, false on errors.

4.4.4.3 caerDeviceConfigSet()

```
bool caerDeviceConfigSet (
              caerDeviceHandle handle,
              int8_t modAddr,
              uint8_t paramAddr,
              uint32_t param )
```

Set a configuration parameter to a given value.

Parameters

handle	a valid device handle.
modAddr	a module address, used to specify which configuration module one wants to update. Negative addresses are used for host-side configuration, while positive addresses (including zero) are used for device-side configuration.
paramAddr	a parameter address, to select a specific parameter to update from this particular configuration module. Only positive numbers (including zero) are allowed.
param	a configuration parameter's new value.

Returns

true if sending the configuration was successful, false on errors.

4.4.4.4 caerDeviceDataGet()

```
caerEventPacketContainer caerDeviceDataGet (
            caerDeviceHandle handle )
```

Get an event packet container, which contains events of various types generated by the device, from the USB data transfer thread for further processing. The returned data structures are allocated in memory and will need to be freed. The caerEventPacketContainerFree() function can be used to correctly free the full container memory. For single caerEventPackets, just use free(). This function can be made blocking with the CAER_HOST_CONFIG_D↩ ATAEXCHANGE_BLOCKING configuration parameter. By default it is non-blocking.

Parameters

handle	a valid device handle.

Returns

a valid event packet container. NULL will be returned on errors, or when there is no container available in non-blocking mode. Always check for this!

4.4.4.5 caerDeviceDataStart()

```
bool caerDeviceDataStart (
            caerDeviceHandle handle,
            void(*)(void *ptr) dataNotifyIncrease,
            void(*)(void *ptr) dataNotifyDecrease,
            void * dataNotifyUserPtr,
            void(*)(void *ptr) dataShutdownNotify,
            void * dataShutdownUserPtr )
```

Start getting data from the device, setting up the USB data transfer thread and starting the data producers (see CA↩ ER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS). Supports notification of new data and shutdown events via user-defined call-backs.

Parameters

handle	a valid device handle.
dataNotifyIncrease	function pointer, called every time a new piece of data available and has been put in the FIFO buffer for consumption. dataNotifyUserPtr will be passed as parameter to the function.
dataNotifyDecrease	function pointer, called every time a new piece of data has been consumed from the FIFO buffer inside caerDeviceDataGet(). dataNotifyUserPtr will be passed as parameter to the function.
dataNotifyUserPtr	pointer that will be passed to the dataNotifyIncrease and dataNotifyDecrease functions. Can be NULL.
dataShutdownNotify	function pointer, called on shut-down of the USB data transfer thread. This can be used to detect exceptional shut-downs that do not come from calling caerDeviceDataStop(), such as when the device is disconnected or all USB transfers fail.
dataShutdownUserPtr	pointer that will be passed to the dataShutdownNotify function. Can be NULL.

Returns

true if starting the data transfer was successful, false on errors.

4.4.4.6 caerDeviceDataStop()

```
bool caerDeviceDataStop (
              caerDeviceHandle handle )
```

Stop getting data from the device, shutting down the USB data transfer thread and stopping the data producers (see CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS). This normal shut-down will also generate a notification (see caerDeviceDataStart()).

Parameters

handle	a valid device handle.

Returns

true if stopping the data transfer was successful, false on errors.

4.4.4.7 caerDeviceOpen()

```
caerDeviceHandle caerDeviceOpen (
              uint16_t deviceID,
              uint16_t deviceType,
              uint8_t busNumberRestrict,
              uint8_t devAddressRestrict,
              const char * serialNumberRestrict )
```

Open a specified USB device, assign an ID to it and return a handle for further usage. Various means can be employed to limit the selection of the device.

Parameters

deviceID	a unique ID to identify the device from others. Will be used as the source for EventPackets being generate from its data.
deviceType	type of the device to open. Currently supported are: CAER_DEVICE_DVS128, CAER_DEVICE_DAVIS_FX2, CAER_DEVICE_DAVIS_FX3
busNumberRestrict	restrict the search for viable devices to only this USB bus number.
devAddressRestrict	restrict the search for viable devices to only this USB device address.
serialNumberRestrict	restrict the search for viable devices to only devices which do possess the given Serial Number in their USB SerialNumber descriptor.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this!

4.4.4.8 caerDeviceSendDefaultConfig()

```
bool caerDeviceSendDefaultConfig (
              caerDeviceHandle handle )
```

Send a set of good default configuration settings to the device. This avoids users having to set every configuration option each time, especially when wanting to get going quickly or just needing to change a few settings to get to the desired operating mode.

Parameters

handle	a valid device handle.

Returns

true if sending the configuration was successful, false on errors.

4.5 events/common.h File Reference

```
#include "../libcaer.h"
```

Macros

- #define TS_OVERFLOW_SHIFT 31
- #define CAER_EVENT_PACKET_HEADER_SIZE 28
- #define CAER_ITERATOR_ALL_START(PACKET_HEADER, EVENT_TYPE)
- #define CAER_ITERATOR_ALL_END }
- #define CAER_ITERATOR_VALID_START(PACKET_HEADER, EVENT_TYPE)
- #define CAER_ITERATOR_VALID_END }

- #define VALID_MARK_SHIFT 0
- #define VALID_MARK_MASK 0x00000001

Typedefs

- typedef struct caer_event_packet_header * caerEventPacketHeader
- typedef const struct caer_event_packet_header * **caerEventPacketHeaderConst**

Enumerations

- enum caer_default_event_types {
  SPECIAL_EVENT = 0, POLARITY_EVENT = 1, FRAME_EVENT = 2, IMU6_EVENT = 3,
  IMU9_EVENT = 4, SAMPLE_EVENT = 5, EAR_EVENT = 6, CONFIG_EVENT = 7,
  POINT1D_EVENT = 8, POINT2D_EVENT = 9, POINT3D_EVENT = 10, POINT4D_EVENT = 11,
  SPIKE_EVENT = 12 }

Functions

- PACKED_STRUCT (struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32_t eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t eventNumber;int32_t eventValid;})
- static int16_t caerEventPacketHeaderGetEventType (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventType (caerEventPacketHeader header, int16_t eventType)
- static int16_t caerEventPacketHeaderGetEventSource (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventSource (caerEventPacketHeader header, int16_t eventSource)
- static int32_t caerEventPacketHeaderGetEventSize (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventSize (caerEventPacketHeader header, int32_t eventSize)
- static int32_t caerEventPacketHeaderGetEventTSOffset (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventTSOffset (caerEventPacketHeader header, int32_t eventTSOffset)
- static int32_t caerEventPacketHeaderGetEventTSOverflow (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventTSOverflow (caerEventPacketHeader header, int32_t eventTSOverflow)
- static int32_t caerEventPacketHeaderGetEventCapacity (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventCapacity (caerEventPacketHeader header, int32_t eventsCapacity)
- static int32_t caerEventPacketHeaderGetEventNumber (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventNumber (caerEventPacketHeader header, int32_t eventsNumber)
- static int32_t caerEventPacketHeaderGetEventValid (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventValid (caerEventPacketHeader header, int32_t eventsValid)
- static const void * caerGenericEventGetEvent (caerEventPacketHeaderConst headerPtr, int32_t n)
- static int32_t caerGenericEventGetTimestamp (const void *eventPtr, caerEventPacketHeaderConst headerPtr)
- static int64_t caerGenericEventGetTimestamp64 (const void *eventPtr, caerEventPacketHeaderConst headerPtr)
- static bool caerGenericEventIsValid (const void *eventPtr)
- static void caerEventPacketClean (void *eventPacket)
- **memset** (((uint8_t *) header)+offset, 0,(size_t)((eventCapacity - eventValid) *eventSize))
- **caerEventPacketHeaderSetEventNumber** (header, eventValid)

- static caerEventPacketHeader caerEventPacketResize (caerEventPacketHeader packet, int32_t newEvent↩ Capacity)
- static caerEventPacketHeader caerEventPacketGrow (caerEventPacketHeader packet, int32_t newEvent↩ Capacity)
- static caerEventPacketHeader caerEventPacketAppend (caerEventPacketHeader packet, caerEventPacket↩ Header appendPacket)
- static void * caerEventPacketCopy (const void *eventPacket)
- static void * caerEventPacketCopyOnlyEvents (const void *eventPacket)
- static void * caerEventPacketCopyOnlyValidEvents (const void *eventPacket)
- **caerEventPacketHeaderSetEventCapacity** (eventPacketCopy, eventValid)
- **caerEventPacketHeaderSetEventNumber** (eventPacketCopy, eventValid)
- **return** (eventPacketCopy)

4.5.1 Detailed Description

Common EventPacket header format definition and handling functions. Every EventPacket, of any type, has as a first member a common header, which describes various properties of the contained events. This allows easy parsing of events. See the 'struct caer_event_packet_header' documentation for more details.

4.5.2 Macro Definition Documentation

4.5.2.1 CAER_EVENT_PACKET_HEADER_SIZE

```
#define CAER_EVENT_PACKET_HEADER_SIZE 28
```

Size of the EventPacket header. This is constant across all supported systems.

4.5.2.2 CAER_ITERATOR_ALL_END

```
#define CAER_ITERATOR_ALL_END }
```

Generic iterator close statement.

4.5.2.3 CAER_ITERATOR_ALL_START

```
#define CAER_ITERATOR_ALL_START(
              PACKET_HEADER,
              EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; \
        caerIteratorCounter < caerEventPacketHeaderGetEventNumber(
      PACKET_HEADER); \
        caerIteratorCounter++) { \
        EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(
      PACKET_HEADER, caerIteratorCounter);
```

Generic iterator over all events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.5.2.4 CAER_ITERATOR_VALID_END

```
#define CAER_ITERATOR_VALID_END }
```

Generic iterator close statement.

4.5.2.5 CAER_ITERATOR_VALID_START

```
#define CAER_ITERATOR_VALID_START(
               PACKET_HEADER,
               EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; \
       caerIteratorCounter < caerEventPacketHeaderGetEventNumber(
      PACKET_HEADER); \
       caerIteratorCounter++) { \
       EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(
      PACKET_HEADER, caerIteratorCounter); \
       if (!caerGenericEventIsValid(caerIteratorElement)) { continue; }
```

Generic iterator over only the valid events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.5.2.6 TS_OVERFLOW_SHIFT

```
#define TS_OVERFLOW_SHIFT 31
```

64bit timestamp support: since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least). The TSOverflow needs to be shifted by 31 thus when constructing such a timestamp.

4.5.2.7 VALID_MARK_MASK

```
#define VALID_MARK_MASK 0x00000001
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.5.2.8 VALID_MARK_SHIFT

```
#define VALID_MARK_SHIFT 0
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.5.3 Typedef Documentation

4.5.3.1 caerEventPacketHeader

```
typedef struct caer_event_packet_header* caerEventPacketHeader
```

Type for pointer to EventPacket header data structure.

4.5.4 Enumeration Type Documentation

4.5.4.1 caer_default_event_types

```
enum caer_default_event_types
```

List of supported event types. Each event type has its own integer representation. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Enumerator

SPECIAL_EVENT	Special events.
POLARITY_EVENT	Polarity (change, DVS) events.
FRAME_EVENT	Frame (intensity, APS) events.
IMU6_EVENT	6 axes IMU events.
IMU9_EVENT	9 axes IMU events.
SAMPLE_EVENT	ADC sample events.
EAR_EVENT	Ear (cochlea) events.
CONFIG_EVENT	Device configuration events.
POINT1D_EVENT	1D measurement events.
POINT2D_EVENT	2D measurement events.
POINT3D_EVENT	3D measurement events.
POINT4D_EVENT	4D measurement events.
SPIKE_EVENT	Spike events.

4.5.5 Function Documentation

4.5.5.1 caerEventPacketAppend()

```
static caerEventPacketHeader caerEventPacketAppend (
            caerEventPacketHeader packet,
            caerEventPacketHeader appendPacket )  [inline], [static]
```

Appends an event packet to another. This is a simple append operation, no timestamp reordering is done. Please ensure time is monotonically increasing over the two packets! Use free() to reclaim this memory afterwards.

Parameters

packet	the main events packet.
appendPacket	the events packet to append on the main one.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way. The appendPacket handle is never touched in any way.

4.5.5.2 caerEventPacketClean()

```
static void caerEventPacketClean (
            void * eventPacket )  [inline], [static]
```

Clean a packet by removing all invalid events, so that the total number of events is the number of valid events. The packet's capacity doesn't change.

Parameters

eventPacket	an event packet to clean.

4.5.5.3 caerEventPacketCopy()

```
static void* caerEventPacketCopy (
            const void * eventPacket )  [inline], [static]
```

Make a full copy of an event packet (up to eventCapacity).

Parameters

eventPacket	an event packet to copy.

Returns

a full copy of an event packet.

4.5.5.4 caerEventPacketCopyOnlyEvents()

```
static void* caerEventPacketCopyOnlyEvents (
            const void * eventPacket )  [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently present events (eventNumber, valid+invalid), and not including the possible extra unused events (up to eventCapacity).

Parameters

eventPacket	an event packet to copy.

Returns

a sized down copy of an event packet.

4.5.5.5 caerEventPacketCopyOnlyValidEvents()

```
static void* caerEventPacketCopyOnlyValidEvents (
            const void * eventPacket )  [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

eventPacket	an event packet to copy.

Returns

a copy of an event packet, containing only valid events.

4.5.5.6 caerEventPacketGrow()

```
static caerEventPacketHeader caerEventPacketGrow (
            caerEventPacketHeader packet,
            int32_t newEventCapacity )  [inline], [static]
```

Grows an event packet. This only supports strictly increasing the size of a packet. For a more flexible resize operation, see caerEventPacketResize(). Use free() to reclaim this memory afterwards.

Parameters

packet	the current event packet.
newEventCapacity	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way.

4.5.5.7 caerEventPacketHeaderGetEventCapacity()

```
static int32_t caerEventPacketHeaderGetEventCapacity (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the maximum number of events this packet can store.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of events this packet can hold.

4.5.5.8 caerEventPacketHeaderGetEventNumber()

```
static int32_t caerEventPacketHeaderGetEventNumber (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of events in this packet.

4.5.5.9 caerEventPacketHeaderGetEventSize()

```
static int32_t caerEventPacketHeaderGetEventSize (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event size in bytes.

4.5.5.10 caerEventPacketHeaderGetEventSource()

```
static int16_t caerEventPacketHeaderGetEventSource (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the numerical event source ID, representing the event source that generated all the events present in this packet.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the numerical event source ID.

4.5.5.11 caerEventPacketHeaderGetEventTSOffset()

```
static int32_t caerEventPacketHeaderGetEventTSOffset (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event timestamp offset in bytes.

4.5.5.12 caerEventPacketHeaderGetEventTSOverflow()

```
static int32_t caerEventPacketHeaderGetEventTSOverflow (
            caerEventPacketHeaderConst header ) [inline], [static]
```

Get the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the packet-level timestamp overflow counter, in microseconds.

4.5.5.13 caerEventPacketHeaderGetEventType()

```
static int16_t caerEventPacketHeaderGetEventType (
            caerEventPacketHeaderConst header ) [inline], [static]
```

Return the numerical event type ID, representing the event type this EventPacket is containing.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the numerical event type (see 'enum caer_default_event_types').

4.5.5.14 caerEventPacketHeaderGetEventValid()

```
static int32_t caerEventPacketHeaderGetEventValid (
            caerEventPacketHeaderConst header ) [inline], [static]
```

Get the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of valid events in this packet.

4.5.5.15 caerEventPacketHeaderSetEventCapacity()

```
static void caerEventPacketHeaderSetEventCapacity (
              caerEventPacketHeader header,
              int32_t eventsCapacity )  [inline], [static]
```

Set the maximum number of events this packet can store. This is determined at packet allocation time and should not be changed during the life-time of the packet.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsCapacity	the number of events this packet can hold.

4.5.5.16 caerEventPacketHeaderSetEventNumber()

```
static void caerEventPacketHeaderSetEventNumber (
              caerEventPacketHeader header,
              int32_t eventsNumber )  [inline], [static]
```

Set the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsNumber	the number of events in this packet.

4.5.5.17 caerEventPacketHeaderSetEventSize()

```
static void caerEventPacketHeaderSetEventSize (
              caerEventPacketHeader header,
              int32_t eventSize )  [inline], [static]
```

Set the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventSize	the event size in bytes.

4.5.5.18 caerEventPacketHeaderSetEventSource()

```
static void caerEventPacketHeaderSetEventSource (
            caerEventPacketHeader header,
            int16_t eventSource )  [inline], [static]
```

Set the numerical event source ID, representing the event source that generated all the events present in this packet. This ID should be unique at least within a process, if not within the whole system, to guarantee correct identification of who generated an event later on.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventSource	the numerical event source ID.

4.5.5.19 caerEventPacketHeaderSetEventTSOffset()

```
static void caerEventPacketHeaderSetEventTSOffset (
            caerEventPacketHeader header,
            int32_t eventTSOffset )  [inline], [static]
```

Set the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventTSOffset	the event timestamp offset in bytes.

4.5.5.20 caerEventPacketHeaderSetEventTSOverflow()

```
static void caerEventPacketHeaderSetEventTSOverflow (
            caerEventPacketHeader header,
            int32_t eventTSOverflow )  [inline], [static]
```

Set the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventTSOverflow	the packet-level timestamp overflow counter, in microseconds.

4.5.5.21 caerEventPacketHeaderSetEventType()

```
static void caerEventPacketHeaderSetEventType (
            caerEventPacketHeader header,
            int16_t eventType )  [inline], [static]
```

Set the numerical event type ID, representing the event type this EventPacket will contain. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventType	the numerical event type (see 'enum caer_default_event_types').

4.5.5.22 caerEventPacketHeaderSetEventValid()

```
static void caerEventPacketHeaderSetEventValid (
            caerEventPacketHeader header,
            int32_t eventsValid )  [inline], [static]
```

Set the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsValid	the number of valid events in this packet.

4.5.5.23 caerEventPacketResize()

```
static caerEventPacketHeader caerEventPacketResize (
            caerEventPacketHeader packet,
            int32_t newEventCapacity )  [inline], [static]
```

Resize an event packet. First, the packet is cleaned (all invalid events removed), then:

- If the old and new event capacity are equal, nothing else changes.

- If the new capacity is bigger, the packet is enlarged and the new events are initialized to all zeros (invalid).
- If the new capacity is smaller, the packet is truncated at the given point. Use free() to reclaim this memory afterwards.

Parameters

packet	the current event packet.
newEventCapacity	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is still valid, but will have been cleaned of all invalid events!

4.5.5.24 caerGenericEventGetEvent()

```
static const void* caerGenericEventGetEvent (
            caerEventPacketHeaderConst headerPtr,
            int32_t n )  [inline], [static]
```

Get a generic pointer to an event, without having to know what event type the packet is containing.

Parameters

headerPtr	a valid EventPacket header pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

a generic pointer to the requested event. NULL on error. This points to unmodifiable memory, as it should never be used for anything other than read operations, such as caerGenericEventGetTimestamp(). Don't modify the memory, you have no idea what it is! If you do know, just use the proper typed packet functions.

4.5.5.25 caerGenericEventGetTimestamp()

```
static int32_t caerGenericEventGetTimestamp (
            const void * eventPtr,
            caerEventPacketHeaderConst headerPtr )  [inline], [static]
```

Get the main 32 bit timestamp for a generic event, without having to know what event type the packet is containing.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.
headerPtr	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 32 bit timestamp of this event.

4.5.5.26 caerGenericEventGetTimestamp64()

```
static int64_t caerGenericEventGetTimestamp64 (
            const void * eventPtr,
            caerEventPacketHeaderConst headerPtr )  [inline], [static]
```

Get the main 64 bit timestamp for a generic event, without having to know what event type the packet is containing. This takes the per-packet timestamp into account too, generating a timestamp that doesn't suffer from overflow problems.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.
headerPtr	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 64 bit timestamp of this event.

4.5.5.27 caerGenericEventIsValid()

```
static bool caerGenericEventIsValid (
            const void * eventPtr )  [inline], [static]
```

Check if the given generic event is valid or not.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.

Returns

true if the event is valid, false otherwise.

4.5.5.28 PACKED_STRUCT()

```
PACKED_STRUCT (
            struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32_t
```

```
eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t event↩
Number;int32_t eventValid;}  )
```

EventPacket header data structure definition. The size, also defined in CAER_EVENT_PACKET_HEADER_SIZE, must always be constant. The header is common to all types of event packets and is always the very first member of an event packet data structure. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

4.6 events/config.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_CONFIGURATION_ITERATOR_ALL_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_ITERATOR_ALL_END }
- #define CAER_CONFIGURATION_ITERATOR_VALID_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_ITERATOR_VALID_END }

- #define MODULE_ADDR_SHIFT 1
- #define MODULE_ADDR_MASK 0x0000007F

Typedefs

- typedef struct caer_configuration_event * caerConfigurationEvent
- typedef struct caer_configuration_event_packet * caerConfigurationEventPacket

Functions

- PACKED_STRUCT (struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameter↩Address;uint32_t parameter;int32_t timestamp;})
- PACKED_STRUCT (struct caer_configuration_event_packet { struct caer_event_packet_header packet↩Header;struct caer_configuration_event events[];})
- caerConfigurationEventPacket caerConfigurationEventPacketAllocate (int32_t eventCapacity, int16_t event↩Source, int32_t tsOverflow)
- static caerConfigurationEvent caerConfigurationEventPacketGetEvent (caerConfigurationEventPacket packet, int32_t n)
- static int32_t caerConfigurationEventGetTimestamp (caerConfigurationEvent event)
- static int64_t caerConfigurationEventGetTimestamp64 (caerConfigurationEvent event, caerConfiguration↩EventPacket packet)
- static void caerConfigurationEventSetTimestamp (caerConfigurationEvent event, int32_t timestamp)
- static bool caerConfigurationEventIsValid (caerConfigurationEvent event)
- static void caerConfigurationEventValidate (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static void caerConfigurationEventInvalidate (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static uint8_t caerConfigurationEventGetModuleAddress (caerConfigurationEvent event)
- static void caerConfigurationEventSetModuleAddress (caerConfigurationEvent event, uint8_t module↩Address)
- static uint8_t caerConfigurationEventGetParameterAddress (caerConfigurationEvent event)
- static void caerConfigurationEventSetParameterAddress (caerConfigurationEvent event, uint8_t parameter↩Address)
- static uint32_t caerConfigurationEventGetParameter (caerConfigurationEvent event)
- static void caerConfigurationEventSetParameter (caerConfigurationEvent event, uint32_t parameter)

4.6.1 Detailed Description

Configuration Events format definition and handling functions. This event contains information about the current configuration of the device. By having configuration as a standardized event format, it becomes host-software agnostic, and it also becomes part of the event stream, enabling easy tracking of changes through time, by putting them into the event stream at the moment they happen. While the resolution of the timestamps for these events is in microseconds for compatibility with all other event types, the precision is in the order of ∼1-20 milliseconds, given that these events are generated and injected on the host-side.

4.6.2 Macro Definition Documentation

4.6.2.1 CAER_CONFIGURATION_ITERATOR_ALL_END

```
#define CAER_CONFIGURATION_ITERATOR_ALL_END }
```

Iterator close statement.

4.6.2.2 CAER_CONFIGURATION_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_ITERATOR_ALL_START(
              CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0; \
        caerConfigurationIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(CONFIGURATION_PACKET)->packetHeader); \
        caerConfigurationIteratorCounter++) { \
        caerConfigurationEvent caerConfigurationIteratorElement =
      caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET,
      caerConfigurationIteratorCounter);
```

Iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIterator↩ Counter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.6.2.3 CAER_CONFIGURATION_ITERATOR_VALID_END

```
#define CAER_CONFIGURATION_ITERATOR_VALID_END }
```

Iterator close statement.

4.6.2.4 CAER_CONFIGURATION_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_ITERATOR_VALID_START(
              CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0; \
       caerConfigurationIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(CONFIGURATION_PACKET)->packetHeader); \
       caerConfigurationIteratorCounter++) { \
       caerConfigurationEvent caerConfigurationIteratorElement =
     caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET,
     caerConfigurationIteratorCounter); \
       if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
     { continue; }
```

Iterator over only the valid configuration events in a packet. Returns the current index in the 'caerConfiguration↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.6.2.5 MODULE_ADDR_MASK

```
#define MODULE_ADDR_MASK 0x0000007F
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.6.2.6 MODULE_ADDR_SHIFT

```
#define MODULE_ADDR_SHIFT 1
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.6.3 Typedef Documentation

4.6.3.1 caerConfigurationEvent

```
typedef struct caer_configuration_event* caerConfigurationEvent
```

Type for pointer to configuration event data structure.

4.6.3.2 caerConfigurationEventPacket

```
typedef struct caer_configuration_event_packet* caerConfigurationEventPacket
```

Type for pointer to configuration event packet data structure.

4.6.4 Function Documentation

4.6.4.1 caerConfigurationEventGetModuleAddress()

```
static uint8_t caerConfigurationEventGetModuleAddress (
            caerConfigurationEvent event )  [inline], [static]
```

Get the configuration event's module address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration module address.

4.6.4.2 caerConfigurationEventGetParameter()

```
static uint32_t caerConfigurationEventGetParameter (
            caerConfigurationEvent event )  [inline], [static]
```

Get the configuration event's parameter.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration parameter.

4.6.4.3 caerConfigurationEventGetParameterAddress()

```
static uint8_t caerConfigurationEventGetParameterAddress (
            caerConfigurationEvent event )  [inline], [static]
```

Get the configuration event's parameter address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration parameter address.

4.6.4.4 caerConfigurationEventGetTimestamp()

```
static int32_t caerConfigurationEventGetTimestamp (
            caerConfigurationEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.6.4.5 caerConfigurationEventGetTimestamp64()

```
static int64_t caerConfigurationEventGetTimestamp64 (
            caerConfigurationEvent event,
            caerConfigurationEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.6.4.6 caerConfigurationEventInvalidate()

```
static void caerConfigurationEventInvalidate (
            caerConfigurationEvent event,
            caerConfigurationEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.6.4.7 caerConfigurationEventIsValid()

```
static bool caerConfigurationEventIsValid (
            caerConfigurationEvent event )  [inline], [static]
```

Check if this configuration event is valid.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.6.4.8 caerConfigurationEventPacketAllocate()

```
caerConfigurationEventPacket caerConfigurationEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new configuration events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid ConfigurationEventPacket handle or NULL on error.

4.6.4.9 caerConfigurationEventPacketGetEvent()

```
static caerConfigurationEvent caerConfigurationEventPacketGetEvent (
            caerConfigurationEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the configuration event at the given index from the event packet.

Parameters

packet	a valid ConfigurationEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested configuration event. NULL on error.

4.6.4.10 caerConfigurationEventSetModuleAddress()

```
static void caerConfigurationEventSetModuleAddress (
            caerConfigurationEvent event,
            uint8_t moduleAddress )  [inline], [static]
```

Set the configuration event's module address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
moduleAddress	configuration module address.

4.6.4.11 caerConfigurationEventSetParameter()

```
static void caerConfigurationEventSetParameter (
            caerConfigurationEvent event,
            uint32_t parameter )  [inline], [static]
```

Set the configuration event's parameter.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
parameter	configuration parameter.

4.6.4.12 caerConfigurationEventSetParameterAddress()

```
static void caerConfigurationEventSetParameterAddress (
            caerConfigurationEvent event,
            uint8_t parameterAddress )  [inline], [static]
```

Set the configuration event's parameter address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
parameterAddress	configuration parameter address.

4.6.4.13 caerConfigurationEventSetTimestamp()

```
static void caerConfigurationEventSetTimestamp (
            caerConfigurationEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.6.4.14 caerConfigurationEventValidate()

```
static void caerConfigurationEventValidate (
            caerConfigurationEvent event,
            caerConfigurationEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.6.4.15 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameterAddress;uint32↩
_t parameter;int32_t timestamp;}  )
```

Configuration event data structure definition. This contains the actual configuration module address, the parameter address and the actual parameter content, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.6.4.16 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_configuration_event_packet { struct caer_event_packet_header packet↩
Header;struct caer_configuration_event events[];}  )
```

Configuration event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.7 events/ear.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_EAR_ITERATOR_ALL_START(EAR_PACKET)
- #define CAER_EAR_ITERATOR_ALL_END }
- #define CAER_EAR_ITERATOR_VALID_START(EAR_PACKET)
- #define CAER_EAR_ITERATOR_VALID_END }

- #define EAR_SHIFT 1
- #define EAR_MASK 0x0000000F
- #define CHANNEL_SHIFT 5
- #define CHANNEL_MASK 0x000007FF
- #define NEURON_SHIFT 16
- #define NEURON_MASK 0x000000FF
- #define FILTER_SHIFT 24
- #define FILTER_MASK 0x000000FF

Typedefs

- typedef struct caer_ear_event * caerEarEvent
- typedef struct caer_ear_event_packet * caerEarEventPacket

Functions

- PACKED_STRUCT (struct caer_ear_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct caer_ear_event events[ ];})
- caerEarEventPacket caerEarEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t ts↩ Overflow)
- static caerEarEvent caerEarEventPacketGetEvent (caerEarEventPacket packet, int32_t n)
- static int32_t caerEarEventGetTimestamp (caerEarEvent event)
- static int64_t caerEarEventGetTimestamp64 (caerEarEvent event, caerEarEventPacket packet)
- static void caerEarEventSetTimestamp (caerEarEvent event, int32_t timestamp)
- static bool caerEarEventIsValid (caerEarEvent event)
- static void caerEarEventValidate (caerEarEvent event, caerEarEventPacket packet)
- static void caerEarEventInvalidate (caerEarEvent event, caerEarEventPacket packet)
- static uint8_t caerEarEventGetEar (caerEarEvent event)
- static void caerEarEventSetEar (caerEarEvent event, uint8_t ear)
- static uint16_t caerEarEventGetChannel (caerEarEvent event)
- static void caerEarEventSetChannel (caerEarEvent event, uint16_t channel)
- static uint8_t **caerEarEventGetNeuron** (caerEarEvent event)
- static void **caerEarEventSetNeuron** (caerEarEvent event, uint8_t neuron)
- static uint8_t **caerEarEventGetFilter** (caerEarEvent event)
- static void **caerEarEventSetFilter** (caerEarEvent event, uint8_t filter)

4.7.1 Detailed Description

Ear (Cochlea) Events format definition and handling functions. This encodes events from a silicon cochlea chip, containing information about which ear (microphone) generated the event, as well as which channel was involved and additional information on filters and neurons.

4.7.2 Macro Definition Documentation

4.7.2.1 CAER_EAR_ITERATOR_ALL_END

```
#define CAER_EAR_ITERATOR_ALL_END }
```

Iterator close statement.

4.7.2.2 CAER_EAR_ITERATOR_ALL_START

```
#define CAER_EAR_ITERATOR_ALL_START(
               EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0; \
       caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader); \
       caerEarIteratorCounter++) { \
       caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter);
```

Iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.7.2.3 CAER_EAR_ITERATOR_VALID_END

```
#define CAER_EAR_ITERATOR_VALID_END }
```

Iterator close statement.

4.7.2.4 CAER_EAR_ITERATOR_VALID_START

```
#define CAER_EAR_ITERATOR_VALID_START(
              EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0; \
        caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader); \
        caerEarIteratorCounter++) { \
        caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter); \
        if (!caerEarEventIsValid(caerEarIteratorElement)) { continue; }
```

Iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.7.2.5 CHANNEL_MASK

```
#define CHANNEL_MASK 0x000007FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.6 CHANNEL_SHIFT

```
#define CHANNEL_SHIFT 5
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.7 EAR_MASK

```
#define EAR_MASK 0x0000000F
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.8 EAR_SHIFT

```
#define EAR_SHIFT 1
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.9 FILTER_MASK

```
#define FILTER_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.10 FILTER_SHIFT

```
#define FILTER_SHIFT 24
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.11 NEURON_MASK

```
#define NEURON_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.2.12 NEURON_SHIFT

```
#define NEURON_SHIFT 16
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.7.3 Typedef Documentation

4.7.3.1 caerEarEvent

```
typedef struct caer_ear_event* caerEarEvent
```

Type for pointer to ear (cochlea) event data structure.

4.7.3.2 caerEarEventPacket

```
typedef struct caer_ear_event_packet* caerEarEventPacket
```

Type for pointer to ear (cochlea) event packet data structure.

4.7.4 Function Documentation

4.7.4.1 caerEarEventGetChannel()

```
static uint16_t caerEarEventGetChannel (
            caerEarEvent event ) [inline], [static]
```

Get the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

the channel (frequency band) ID.

4.7.4.2 caerEarEventGetEar()

```
static uint8_t caerEarEventGetEar (
            caerEarEvent event ) [inline], [static]
```

Get the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

the ear (microphone) ID.

4.7.4.3 caerEarEventGetTimestamp()

```
static int32_t caerEarEventGetTimestamp (
            caerEarEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.7.4.4 caerEarEventGetTimestamp64()

```
static int64_t caerEarEventGetTimestamp64 (
            caerEarEvent event,
            caerEarEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.7.4.5 caerEarEventInvalidate()

```
static void caerEarEventInvalidate (
            caerEarEvent event,
            caerEarEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.7.4.6 caerEarEventIsValid()

```
static bool caerEarEventIsValid (
            caerEarEvent event )  [inline], [static]
```

Check if this ear (cochlea) event is valid.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.7.4.7 caerEarEventPacketAllocate()

```
caerEarEventPacket caerEarEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new ear (cochlea) events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid EarEventPacket handle or NULL on error.

4.7.4.8 caerEarEventPacketGetEvent()

```
static caerEarEvent caerEarEventPacketGetEvent (
            caerEarEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the ear (cochlea) event at the given index from the event packet.

Parameters

packet	a valid EarEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ear (cochlea) event. NULL on error.

4.7.4.9 caerEarEventSetChannel()

```
static void caerEarEventSetChannel (
            caerEarEvent event,
            uint16_t channel )  [inline], [static]
```

Set the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
channel	the channel (frequency band) ID.

4.7.4.10 caerEarEventSetEar()

```
static void caerEarEventSetEar (
            caerEarEvent event,
            uint8_t ear )  [inline], [static]
```

Set the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
ear	the ear (microphone) ID.

4.7.4.11 caerEarEventSetTimestamp()

```
static void caerEarEventSetTimestamp (
            caerEarEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.7.4.12 caerEarEventValidate()

```
static void caerEarEventValidate (
            caerEarEvent event,
            caerEarEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.7.4.13 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_ear_event { uint32_t data;int32_t timestamp;}  )
```

Ear (cochlea) event data structure definition. Contains information on events gotten from a cochlea chip: ears, channels, neurons and filters are stored. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.7.4.14 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct
caer_ear_event events[];}  )
```

Ear (cochlea) event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.8 events/frame.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_FRAME_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_ITERATOR_ALL_END }
- #define CAER_FRAME_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_ITERATOR_VALID_END }
- #define CAER_FRAME_REVERSE_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_REVERSE_ITERATOR_ALL_END }
- #define CAER_FRAME_REVERSE_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_REVERSE_ITERATOR_VALID_END }

- #define COLOR_CHANNELS_SHIFT 1
- #define COLOR_CHANNELS_MASK 0x00000007
- #define COLOR_FILTER_SHIFT 4
- #define COLOR_FILTER_MASK 0x0000000F
- #define ROI_IDENTIFIER_SHIFT 8
- #define ROI_IDENTIFIER_MASK 0x0000007F

Typedefs

- typedef struct caer_frame_event * caerFrameEvent
- typedef struct caer_frame_event_packet * caerFrameEventPacket

Enumerations

- enum caer_frame_event_color_channels { GRAYSCALE = 1, RGB = 3, RGBA = 4 }
- enum caer_frame_event_color_filter {
  MONO = 0, RGBG = 1, GRGB = 2, GBGR = 3,
  BGRG = 4, RGBW = 5, GRWB = 6, WBGR = 7,
  BWRG = 8 }

Functions

- PACKED_STRUCT (struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32_t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32_t positionY;uint16_t pixels[ ];})
- PACKED_STRUCT (struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;})
- caerFrameEventPacket caerFrameEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int32_t maxLengthX, int32_t maxLengthY, int16_t maxChannelNumber)
- static caerFrameEvent caerFrameEventPacketGetEvent (caerFrameEventPacket packet, int32_t n)
- static int32_t caerFrameEventGetTSStartOfFrame (caerFrameEvent event)
- static int64_t caerFrameEventGetTSStartOfFrame64 (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventSetTSStartOfFrame (caerFrameEvent event, int32_t startFrame)
- static int32_t caerFrameEventGetTSEndOfFrame (caerFrameEvent event)
- static int64_t caerFrameEventGetTSEndOfFrame64 (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventSetTSEndOfFrame (caerFrameEvent event, int32_t endFrame)
- static int32_t caerFrameEventGetTSStartOfExposure (caerFrameEvent event)
- static int64_t caerFrameEventGetTSStartOfExposure64 (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventSetTSStartOfExposure (caerFrameEvent event, int32_t startExposure)
- static int32_t caerFrameEventGetTSEndOfExposure (caerFrameEvent event)
- static int64_t caerFrameEventGetTSEndOfExposure64 (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventSetTSEndOfExposure (caerFrameEvent event, int32_t endExposure)
- static int32_t caerFrameEventGetExposureLength (caerFrameEvent event)
- static int32_t caerFrameEventGetTimestamp (caerFrameEvent event)
- static int64_t caerFrameEventGetTimestamp64 (caerFrameEvent event, caerFrameEventPacket packet)
- static bool caerFrameEventIsValid (caerFrameEvent event)
- static void caerFrameEventValidate (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventInvalidate (caerFrameEvent event, caerFrameEventPacket packet)
- static size_t caerFrameEventPacketGetPixelsSize (caerFrameEventPacket packet)
- static size_t caerFrameEventPacketGetPixelsMaxIndex (caerFrameEventPacket packet)
- static uint8_t caerFrameEventGetROIIdentifier (caerFrameEvent event)
- static void caerFrameEventSetROIIdentifier (caerFrameEvent event, uint8_t roiIdentifier)
- static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (caerFrameEvent event)
- static void caerFrameEventSetColorFilter (caerFrameEvent event, enum caer_frame_event_color_filter colorFilter)
- static int32_t caerFrameEventGetLengthX (caerFrameEvent event)
- static int32_t caerFrameEventGetLengthY (caerFrameEvent event)
- static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (caerFrameEvent event)
- static void caerFrameEventSetLengthXLengthYChannelNumber (caerFrameEvent event, int32_t lengthX, int32_t lengthY, enum caer_frame_event_color_channels channelNumber, caerFrameEventPacket packet)
- static size_t caerFrameEventGetPixelsMaxIndex (caerFrameEvent event)
- static size_t caerFrameEventGetPixelsSize (caerFrameEvent event)
- static int32_t caerFrameEventGetPositionX (caerFrameEvent event)
- static void caerFrameEventSetPositionX (caerFrameEvent event, int32_t positionX)
- static int32_t caerFrameEventGetPositionY (caerFrameEvent event)
- static void caerFrameEventSetPositionY (caerFrameEvent event, int32_t positionY)
- static uint16_t caerFrameEventGetPixel (caerFrameEvent event, int32_t xAddress, int32_t yAddress)
- static void caerFrameEventSetPixel (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)
- static uint16_t caerFrameEventGetPixelForChannel (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void caerFrameEventSetPixelForChannel (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)

- static uint16_t caerFrameEventGetPixelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t yAddress)
- static void caerFrameEventSetPixelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)
- static uint16_t caerFrameEventGetPixelForChannelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void caerFrameEventSetPixelForChannelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)
- static uint16_t * caerFrameEventGetPixelArrayUnsafe (caerFrameEvent event)

4.8.1 Detailed Description

Frame Events format definition and handling functions. This event type encodes intensity frames, like you would get from a normal APS camera. It supports multiple channels for color, color filter information, as well as multiple Regions of Interest (ROI). The (0, 0) pixel is in the upper left corner of the screen, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

4.8.2 Macro Definition Documentation

4.8.2.1 CAER_FRAME_ITERATOR_ALL_END

```
#define CAER_FRAME_ITERATOR_ALL_END }
```

Iterator close statement.

4.8.2.2 CAER_FRAME_ITERATOR_ALL_START

```
#define CAER_FRAME_ITERATOR_ALL_START(
              FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
       caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (FRAME_PACKET)->packetHeader); \
       caerFrameIteratorCounter++) { \
       caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
      FRAME_PACKET, caerFrameIteratorCounter);
```

Iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.8.2.3 CAER_FRAME_ITERATOR_VALID_END

```
#define CAER_FRAME_ITERATOR_VALID_END }
```

Iterator close statement.

4.8.2.4 CAER_FRAME_ITERATOR_VALID_START

```
#define CAER_FRAME_ITERATOR_VALID_START(
              FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
       caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(&
     (FRAME_PACKET)->packetHeader); \
       caerFrameIteratorCounter++) { \
       caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
     FRAME_PACKET, caerFrameIteratorCounter); \
       if (!caerFrameEventIsValid(caerFrameIteratorElement)) { continue; }
```

Iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.8.2.5 CAER_FRAME_REVERSE_ITERATOR_ALL_END

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.8.2.6 CAER_FRAME_REVERSE_ITERATOR_ALL_START

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_START(
              FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
     (&(FRAME_PACKET)->packetHeader) - 1; \
       caerFrameIteratorCounter >= 0; \
       caerFrameIteratorCounter--) { \
       caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
     FRAME_PACKET, caerFrameIteratorCounter);
```

Reverse iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.8.2.7 CAER_FRAME_REVERSE_ITERATOR_VALID_END

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.8.2.8 CAER_FRAME_REVERSE_ITERATOR_VALID_START

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_START(
              FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(FRAME_PACKET)->packetHeader) - 1; \
        caerFrameIteratorCounter >= 0; \
        caerFrameIteratorCounter--) { \
        caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
      FRAME_PACKET, caerFrameIteratorCounter); \
        if (!caerFrameEventIsValid(caerFrameIteratorElement)) { continue; }
```

Reverse iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIterator↩ Counter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caer↩ FrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.8.2.9 COLOR_CHANNELS_MASK

```
#define COLOR_CHANNELS_MASK 0x00000007
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.2.10 COLOR_CHANNELS_SHIFT

```
#define COLOR_CHANNELS_SHIFT 1
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.2.11 COLOR_FILTER_MASK

```
#define COLOR_FILTER_MASK 0x0000000F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.2.12 COLOR_FILTER_SHIFT

```
#define COLOR_FILTER_SHIFT 4
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.2.13 ROI_IDENTIFIER_MASK

```
#define ROI_IDENTIFIER_MASK 0x0000007F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.2.14 ROI_IDENTIFIER_SHIFT

```
#define ROI_IDENTIFIER_SHIFT 8
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame↩ _event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.8.3 Typedef Documentation

4.8.3.1 caerFrameEvent

```
typedef struct caer_frame_event* caerFrameEvent
```

Type for pointer to frame event data structure.

4.8.3.2 caerFrameEventPacket

```
typedef struct caer_frame_event_packet* caerFrameEventPacket
```

Type for pointer to frame event packet data structure.

4.8.4 Enumeration Type Documentation

4.8.4.1 caer_frame_event_color_channels

```
enum caer_frame_event_color_channels
```

List of all frame event color channel identifiers. Used to interpret the frame event color channel field.

Enumerator

GRAYSCALE	Grayscale, one channel only.
RGB	Red Green Blue, 3 color channels.
RGBA	Red Green Blue Alpha, 3 color channels plus transparency.

4.8.4.2 caer_frame_event_color_filter

```
enum caer_frame_event_color_filter
```

List of all frame event color filter identifiers. Used to interpret the frame event color filter field.

Enumerator

MONO	No color filter present, all light passes.
RGBG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 1.
GRGB	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 2.
GBGR	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 3.
BGRG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 4.
RGBW	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 1.
GRWB	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 2.
WBGR	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 3.
BWRG	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 4.

4.8.5 Function Documentation

4.8.5.1 caerFrameEventGetChannelNumber()

```
static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (
            caerFrameEvent event )  [inline], [static]
```

Get the actual color channels number for the current frame. This can be used to store RGB frames for example.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame color channels number.

4.8.5.2 caerFrameEventGetColorFilter()

```
static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (
            caerFrameEvent event ) [inline], [static]
```

Get the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

color filter identifier.

4.8.5.3 caerFrameEventGetExposureLength()

```
static int32_t caerFrameEventGetExposureLength (
            caerFrameEvent event ) [inline], [static]
```

The total length, in microseconds, of the frame exposure time.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

the exposure time in microseconds.

4.8.5.4 caerFrameEventGetLengthX()

```
static int32_t caerFrameEventGetLengthX (
            caerFrameEvent event ) [inline], [static]
```

Get the actual X axis length for the current frame.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame X axis length.

4.8.5.5 caerFrameEventGetLengthY()

```
static int32_t caerFrameEventGetLengthY (
            caerFrameEvent event )  [inline], [static]
```

Get the actual Y axis length for the current frame.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame Y axis length.

4.8.5.6 caerFrameEventGetPixel()

```
static uint16_t caerFrameEventGetPixel (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).

Returns

pixel value (normalized to 16 bit depth).

4.8.5.7 caerFrameEventGetPixelArrayUnsafe()

```
static uint16_t* caerFrameEventGetPixelArrayUnsafe (
            caerFrameEvent event )  [inline], [static]
```

Get a direct reference to the underlying pixels array. This can be used to both get and set values. No checks at all are performed at any point, nor any conversions, use this at your own risk! Remember that the 16 bit pixel values are in little-endian! The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

the pixels array (16 bit integers are little-endian).

4.8.5.8 caerFrameEventGetPixelForChannel()

```
static uint16_t caerFrameEventGetPixelForChannel (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
channel	the channel number (checked).

Returns

pixel value (normalized to 16 bit depth).

4.8.5.9 caerFrameEventGetPixelForChannelUnsafe()

```
static uint16_t caerFrameEventGetPixelForChannelUnsafe (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
channel	the channel number (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.8.5.10 caerFrameEventGetPixelsMaxIndex()

```
static size_t caerFrameEventGetPixelsMaxIndex (
            caerFrameEvent event )  [inline], [static]
```

Get the maximum valid index into the pixel array, at which you can still get valid pixels.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

maximum valid pixels array index.

4.8.5.11 caerFrameEventGetPixelsSize()

```
static size_t caerFrameEventGetPixelsSize (
            caerFrameEvent event )  [inline], [static]
```

Get the maximum size of the pixels array in bytes, in which you can still get valid pixels.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

maximum valid pixels array size in bytes.

4.8.5.12 caerFrameEventGetPixelUnsafe()

```
static uint16_t caerFrameEventGetPixelUnsafe (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.8.5.13 caerFrameEventGetPositionX()

```
static int32_t caerFrameEventGetPositionX (
             caerFrameEvent event )  [inline], [static]
```

Get the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

X axis position offset.

4.8.5.14 caerFrameEventGetPositionY()

```
static int32_t caerFrameEventGetPositionY (
             caerFrameEvent event )  [inline], [static]
```

Get the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

Y axis position offset.

4.8.5.15 caerFrameEventGetROIIdentifier()

```
static uint8_t caerFrameEventGetROIIdentifier (
            caerFrameEvent event ) [inline], [static]
```

Get the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

numerical ROI identifier.

4.8.5.16 caerFrameEventGetTimestamp()

```
static int32_t caerFrameEventGetTimestamp (
            caerFrameEvent event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. This is a median of the exposure timestamps. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGet↩EventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.8.5.17 caerFrameEventGetTimestamp64()

```
static int64_t caerFrameEventGetTimestamp64 (
            caerFrameEvent event,
            caerFrameEventPacket packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. This is a median of the exposure timestamps. See 'caerEvent↩PacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.8.5.18 caerFrameEventGetTSEndOfExposure()

```
static int32_t caerFrameEventGetTSEndOfExposure (
            caerFrameEvent event ) [inline], [static]
```

Get the 32bit end of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond end of exposure timestamp.

4.8.5.19 caerFrameEventGetTSEndOfExposure64()

```
static int64_t caerFrameEventGetTSEndOfExposure64 (
            caerFrameEvent event,
            caerFrameEventPacket packet ) [inline], [static]
```

Get the 64bit end of exposure timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of exposure timestamp.

4.8.5.20 caerFrameEventGetTSEndOfFrame()

```
static int32_t caerFrameEventGetTSEndOfFrame (
            caerFrameEvent event ) [inline], [static]
```

Get the 32bit end of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond end of frame timestamp.

4.8.5.21 caerFrameEventGetTSEndOfFrame64()

```
static int64_t caerFrameEventGetTSEndOfFrame64 (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Get the 64bit end of frame capture timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTS↩Overflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of frame timestamp.

4.8.5.22 caerFrameEventGetTSStartOfExposure()

```
static int32_t caerFrameEventGetTSStartOfExposure (
            caerFrameEvent event )  [inline], [static]
```

Get the 32bit start of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond start of exposure timestamp.

4.8.5.23 caerFrameEventGetTSStartOfExposure64()

```
static int64_t caerFrameEventGetTSStartOfExposure64 (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Get the 64bit start of exposure timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of exposure timestamp.

4.8.5.24 caerFrameEventGetTSStartOfFrame()

```
static int32_t caerFrameEventGetTSStartOfFrame (
            caerFrameEvent event )  [inline], [static]
```

Get the 32bit start of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond start of frame timestamp.

4.8.5.25 caerFrameEventGetTSStartOfFrame64()

```
static int64_t caerFrameEventGetTSStartOfFrame64 (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Get the 64bit start of frame capture timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTS↩Overflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of frame timestamp.

4.8.5.26 caerFrameEventInvalidate()

```
static void caerFrameEventInvalidate (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.8.5.27 caerFrameEventIsValid()

```
static bool caerFrameEventIsValid (
            caerFrameEvent event )  [inline], [static]
```

Check if this frame event is valid.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.8.5.28 caerFrameEventPacketAllocate()

```
caerFrameEventPacket caerFrameEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow,
            int32_t maxLengthX,
            int32_t maxLengthY,
            int16_t maxChannelNumber )
```

Allocate a new frame events packet. Use free() to reclaim this memory. The frame events allocate memory for a maximum sized pixels array, depending on the parameters passed to this function, so that every event occupies the same amount of memory (constant size). The actual frames inside of it might be smaller than that, for example when using ROI, and their actual size is stored inside the frame event and should always be queried from there. The unused part of a pixels array is guaranteed to be zeros.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.
maxLengthX	the maximum expected X axis size for frames in this packet.
maxLengthY	the maximum expected Y axis size for frames in this packet.
maxChannelNumber	the maximum expected number of channels for frames in this packet.

Returns

a valid FrameEventPacket handle or NULL on error.

4.8.5.29 caerFrameEventPacketGetEvent()

```
static caerFrameEvent caerFrameEventPacketGetEvent (
            caerFrameEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the frame event at the given index from the event packet.

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested frame event. NULL on error.

4.8.5.30 caerFrameEventPacketGetPixelsMaxIndex()

```
static size_t caerFrameEventPacketGetPixelsMaxIndex (
            caerFrameEventPacket packet )  [inline], [static]
```

Get the maximum index into the pixels array, based upon how much memory was allocated to it by 'caerFrameEventPacketAllocate()'.

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.

Returns

maximum pixels array index.

4.8.5.31 caerFrameEventPacketGetPixelsSize()

```
static size_t caerFrameEventPacketGetPixelsSize (
            caerFrameEventPacket packet )  [inline], [static]
```

Get the maximum size of the pixels array in bytes, based upon how much memory was allocated to it by 'caerFrameEventPacketAllocate()'.

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.

Returns

maximum pixels array size in bytes.

4.8.5.32 caerFrameEventSetColorFilter()

```
static void caerFrameEventSetColorFilter (
            caerFrameEvent event,
            enum caer_frame_event_color_filter colorFilter )  [inline], [static]
```

Set the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
colorFilter	color filter identifier.

4.8.5.33 caerFrameEventSetLengthXLengthYChannelNumber()

```
static void caerFrameEventSetLengthXLengthYChannelNumber (
            caerFrameEvent event,
            int32_t lengthX,
            int32_t lengthY,
            enum caer_frame_event_color_channels channelNumber,
            caerFrameEventPacket packet )  [inline], [static]
```

Set the X and Y axes length and the color channels number for a frame, while taking into account the maximum amount of memory available for the pixel array, as allocated in 'caerFrameEventPacketAllocate()'.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
lengthX	the frame's X axis length.
lengthY	the frame's Y axis length.
channelNumber	the number of color channels for this frame.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.8.5.34 caerFrameEventSetPixel()

```
static void caerFrameEventSetPixel (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
pixelValue	pixel value (normalized to 16 bit depth).

4.8.5.35 caerFrameEventSetPixelForChannel()

```
static void caerFrameEventSetPixelForChannel (
            caerFrameEvent event,
```

```
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel,
            uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
channel	the channel number (checked).
pixelValue	pixel value (normalized to 16 bit depth).

4.8.5.36 caerFrameEventSetPixelForChannelUnsafe()

```
static void caerFrameEventSetPixelForChannelUnsafe (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel,
            uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
channel	the channel number (unchecked).
pixelValue	pixel value (normalized to 16 bit depth).

4.8.5.37 caerFrameEventSetPixelUnsafe()

```
static void caerFrameEventSetPixelUnsafe (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
pixelValue	pixel value (normalized to 16 bit depth).

4.8.5.38 caerFrameEventSetPositionX()

```
static void caerFrameEventSetPositionX (
            caerFrameEvent event,
            int32_t positionX )  [inline], [static]
```

Set the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
positionX	X axis position offset.

4.8.5.39 caerFrameEventSetPositionY()

```
static void caerFrameEventSetPositionY (
            caerFrameEvent event,
            int32_t positionY )  [inline], [static]
```

Set the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
positionY	Y axis position offset.

4.8.5.40 caerFrameEventSetROIIdentifier()

```
static void caerFrameEventSetROIIdentifier (
            caerFrameEvent event,
            uint8_t roiIdentifier )  [inline], [static]
```

Set the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
roiIdentifier	numerical ROI identifier.

4.8.5.41 caerFrameEventSetTSEndOfExposure()

```
static void caerFrameEventSetTSEndOfExposure (
              caerFrameEvent event,
              int32_t endExposure )  [inline], [static]
```

Set the 32bit end of exposure timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
endExposure	a positive 32bit microsecond timestamp.

4.8.5.42 caerFrameEventSetTSEndOfFrame()

```
static void caerFrameEventSetTSEndOfFrame (
              caerFrameEvent event,
              int32_t endFrame )  [inline], [static]
```

Set the 32bit end of frame capture timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
endFrame	a positive 32bit microsecond timestamp.

4.8.5.43 caerFrameEventSetTSStartOfExposure()

```
static void caerFrameEventSetTSStartOfExposure (
              caerFrameEvent event,
              int32_t startExposure )  [inline], [static]
```

Set the 32bit start of exposure timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
startExposure	a positive 32bit microsecond timestamp.

4.8.5.44 caerFrameEventSetTSStartOfFrame()

```
static void caerFrameEventSetTSStartOfFrame (
            caerFrameEvent event,
            int32_t startFrame )  [inline], [static]
```

Set the 32bit start of frame capture timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
startFrame	a positive 32bit microsecond timestamp.

4.8.5.45 caerFrameEventValidate()

```
static void caerFrameEventValidate (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.8.5.46 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32↩
_t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32↩
_t positionY;uint16_t pixels[];}  )
```

Frame event data structure definition. This contains the actual information on the frame (ROI, color channels, color filter), several timestamps to signal start and end of capture and of exposure, as well as the actual pixels, in a 16 bit normalized format. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.8.5.47 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;}
)
```

Frame event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block. Direct access to the events array is not possible for Frame events. To calculate position offsets, use the 'eventSize' field in the packet header.

4.9 events/imu6.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_IMU6_ITERATOR_ALL_START(IMU6_PACKET)
- #define CAER_IMU6_ITERATOR_ALL_END }
- #define CAER_IMU6_ITERATOR_VALID_START(IMU6_PACKET)
- #define CAER_IMU6_ITERATOR_VALID_END }

Typedefs

- typedef struct caer_imu6_event * caerIMU6Event
- typedef struct caer_imu6_event_packet * caerIMU6EventPacket

Functions

- PACKED_STRUCT (struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;})
- PACKED_STRUCT (struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu6_event events[ ];})
- caerIMU6EventPacket caerIMU6EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerIMU6Event caerIMU6EventPacketGetEvent (caerIMU6EventPacket packet, int32_t n)
- static int32_t caerIMU6EventGetTimestamp (caerIMU6Event event)
- static int64_t caerIMU6EventGetTimestamp64 (caerIMU6Event event, caerIMU6EventPacket packet)
- static void caerIMU6EventSetTimestamp (caerIMU6Event event, int32_t timestamp)
- static bool caerIMU6EventIsValid (caerIMU6Event event)
- static void caerIMU6EventValidate (caerIMU6Event event, caerIMU6EventPacket packet)
- static void caerIMU6EventInvalidate (caerIMU6Event event, caerIMU6EventPacket packet)
- static float caerIMU6EventGetAccelX (caerIMU6Event event)
- static void caerIMU6EventSetAccelX (caerIMU6Event event, float accelX)
- static float caerIMU6EventGetAccelY (caerIMU6Event event)
- static void caerIMU6EventSetAccelY (caerIMU6Event event, float accelY)
- static float caerIMU6EventGetAccelZ (caerIMU6Event event)
- static void caerIMU6EventSetAccelZ (caerIMU6Event event, float accelZ)
- static float caerIMU6EventGetGyroX (caerIMU6Event event)
- static void caerIMU6EventSetGyroX (caerIMU6Event event, float gyroX)
- static float caerIMU6EventGetGyroY (caerIMU6Event event)
- static void caerIMU6EventSetGyroY (caerIMU6Event event, float gyroY)
- static float caerIMU6EventGetGyroZ (caerIMU6Event event)
- static void caerIMU6EventSetGyroZ (caerIMU6Event event, float gyroZ)
- static float caerIMU6EventGetTemp (caerIMU6Event event)
- static void caerIMU6EventSetTemp (caerIMU6Event event, float temp)

4.9.1 Detailed Description

IMU6 (6 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included.

4.9.2 Macro Definition Documentation

4.9.2.1 CAER_IMU6_ITERATOR_ALL_END

```
#define CAER_IMU6_ITERATOR_ALL_END }
```

Iterator close statement.

4.9.2.2 CAER_IMU6_ITERATOR_ALL_START

```
#define CAER_IMU6_ITERATOR_ALL_START(
               IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
        caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      IMU6_PACKET)->packetHeader); \
        caerIMU6IteratorCounter++) { \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter);
```

Iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.9.2.3 CAER_IMU6_ITERATOR_VALID_END

```
#define CAER_IMU6_ITERATOR_VALID_END }
```

Iterator close statement.

4.9.2.4 CAER_IMU6_ITERATOR_VALID_START

```
#define CAER_IMU6_ITERATOR_VALID_START(
               IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
        caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      IMU6_PACKET)->packetHeader); \
        caerIMU6IteratorCounter++) { \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter); \
        if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) { continue; }
```

Iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.9.3 Typedef Documentation

4.9.3.1 caerIMU6Event

```
typedef struct caer_imu6_event* caerIMU6Event
```

Type for pointer to IMU 6-axes event data structure.

4.9.3.2 caerIMU6EventPacket

```
typedef struct caer_imu6_event_packet* caerIMU6EventPacket
```

Type for pointer to IMU 6-axes event packet data structure.

4.9.4 Function Documentation

4.9.4.1 caerIMU6EventGetAccelX()

```
static float caerIMU6EventGetAccelX (
            caerIMU6Event event )  [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the X axis.

4.9.4.2 caerIMU6EventGetAccelY()

```
static float caerIMU6EventGetAccelY (
            caerIMU6Event event )  [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the Y axis.

4.9.4.3 caerIMU6EventGetAccelZ()

```
static float caerIMU6EventGetAccelZ (
            caerIMU6Event event )  [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the Z axis.

4.9.4.4 caerIMU6EventGetGyroX()

```
static float caerIMU6EventGetGyroX (
            caerIMU6Event event )  [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the X axis (roll).

4.9.4.5 caerIMU6EventGetGyroY()

```
static float caerIMU6EventGetGyroY (
            caerIMU6Event event )  [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the Y axis (pitch).

4.9.4.6 caerIMU6EventGetGyroZ()

```
static float caerIMU6EventGetGyroZ (
            caerIMU6Event event )  [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the Z axis (yaw).

4.9.4.7 caerIMU6EventGetTemp()

```
static float caerIMU6EventGetTemp (
            caerIMU6Event event )  [inline], [static]
```

Get the temperature reading. This is in °C.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

temperature in °C.

4.9.4.8 caerIMU6EventGetTimestamp()

```
static int32_t caerIMU6EventGetTimestamp (
            caerIMU6Event event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.9.4.9 caerIMU6EventGetTimestamp64()

```
static int64_t caerIMU6EventGetTimestamp64 (
            caerIMU6Event event,
            caerIMU6EventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.9.4.10 caerIMU6EventInvalidate()

```
static void caerIMU6EventInvalidate (
            caerIMU6Event event,
            caerIMU6EventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.9.4.11 caerIMU6EventIsValid()

```
static bool caerIMU6EventIsValid (
            caerIMU6Event event )  [inline], [static]
```

Check if this IMU 6-axes event is valid.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.9.4.12 caerIMU6EventPacketAllocate()

```
caerIMU6EventPacket caerIMU6EventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new IMU 6-axes events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid IMU6EventPacket handle or NULL on error.

4.9.4.13 caerIMU6EventPacketGetEvent()

```
static caerIMU6Event caerIMU6EventPacketGetEvent (
            caerIMU6EventPacket packet,
            int32_t n )  [inline], [static]
```

Get the IMU 6-axes event at the given index from the event packet.

Parameters

packet	a valid IMU6EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 6-axes event. NULL on error.

4.9.4.14 caerIMU6EventSetAccelX()

```
static void caerIMU6EventSetAccelX (
            caerIMU6Event event,
            float accelX )  [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelX	acceleration on the X axis.

4.9.4.15 caerIMU6EventSetAccelY()

```
static void caerIMU6EventSetAccelY (
            caerIMU6Event event,
            float accelY )  [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelY	acceleration on the Y axis.

4.9.4.16 caerIMU6EventSetAccelZ()

```
static void caerIMU6EventSetAccelZ (
            caerIMU6Event event,
            float accelZ )  [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelZ	acceleration on the Z axis.

4.9.4.17 caerIMU6EventSetGyroX()

```
static void caerIMU6EventSetGyroX (
            caerIMU6Event event,
            float gyroX )  [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroX	angular velocity on the X axis (roll).

4.9.4.18 caerIMU6EventSetGyroY()

```
static void caerIMU6EventSetGyroY (
            caerIMU6Event event,
            float gyroY )  [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroY	angular velocity on the Y axis (pitch).

4.9.4.19 caerIMU6EventSetGyroZ()

```
static void caerIMU6EventSetGyroZ (
            caerIMU6Event event,
            float gyroZ )  [inline], [static]
```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroZ	angular velocity on the Z axis (yaw).

4.9.4.20 caerIMU6EventSetTemp()

```
static void caerIMU6EventSetTemp (
            caerIMU6Event event,
            float temp )  [inline], [static]
```

Set the temperature reading. This is in °C.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
temp	temperature in °C.

4.9.4.21 caerIMU6EventSetTimestamp()

```
static void caerIMU6EventSetTimestamp (
            caerIMU6Event event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.9.4.22 caerIMU6EventValidate()

```
static void caerIMU6EventValidate (
            caerIMU6Event event,
            caerIMU6EventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.9.4.23 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float
accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;}  )
```

IMU 6-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.9.4.24 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct
caer_imu6_event events[];}  )
```

IMU 6-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.10 events/imu9.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_IMU9_ITERATOR_ALL_START(IMU9_PACKET)
- #define CAER_IMU9_ITERATOR_ALL_END }
- #define CAER_IMU9_ITERATOR_VALID_START(IMU9_PACKET)
- #define CAER_IMU9_ITERATOR_VALID_END }

Typedefs

- typedef struct caer_imu9_event * caerIMU9Event
- typedef struct caer_imu9_event_packet * caerIMU9EventPacket

Functions

- PACKED_STRUCT (struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float comp_y;float comp_z;})
- PACKED_STRUCT (struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu9_event events[ ];})
- caerIMU9EventPacket caerIMU9EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerIMU9Event caerIMU9EventPacketGetEvent (caerIMU9EventPacket packet, int32_t n)
- static int32_t caerIMU9EventGetTimestamp (caerIMU9Event event)
- static int64_t caerIMU9EventGetTimestamp64 (caerIMU9Event event, caerIMU9EventPacket packet)
- static void caerIMU9EventSetTimestamp (caerIMU9Event event, int32_t timestamp)
- static bool caerIMU9EventIsValid (caerIMU9Event event)
- static void caerIMU9EventValidate (caerIMU9Event event, caerIMU9EventPacket packet)
- static void caerIMU9EventInvalidate (caerIMU9Event event, caerIMU9EventPacket packet)
- static float caerIMU9EventGetAccelX (caerIMU9Event event)
- static void caerIMU9EventSetAccelX (caerIMU9Event event, float accelX)
- static float caerIMU9EventGetAccelY (caerIMU9Event event)
- static void caerIMU9EventSetAccelY (caerIMU9Event event, float accelY)
- static float caerIMU9EventGetAccelZ (caerIMU9Event event)
- static void caerIMU9EventSetAccelZ (caerIMU9Event event, float accelZ)
- static float caerIMU9EventGetGyroX (caerIMU9Event event)
- static void caerIMU9EventSetGyroX (caerIMU9Event event, float gyroX)
- static float caerIMU9EventGetGyroY (caerIMU9Event event)
- static void caerIMU9EventSetGyroY (caerIMU9Event event, float gyroY)
- static float caerIMU9EventGetGyroZ (caerIMU9Event event)
- static void caerIMU9EventSetGyroZ (caerIMU9Event event, float gyroZ)
- static float caerIMU9EventGetTemp (caerIMU9Event event)
- static void caerIMU9EventSetTemp (caerIMU9Event event, float temp)
- static float caerIMU9EventGetCompX (caerIMU9Event event)
- static void caerIMU9EventSetCompX (caerIMU9Event event, float compX)
- static float caerIMU9EventGetCompY (caerIMU9Event event)
- static void caerIMU9EventSetCompY (caerIMU9Event event, float compY)
- static float caerIMU9EventGetCompZ (caerIMU9Event event)
- static void caerIMU9EventSetCompZ (caerIMU9Event event, float compZ)

4.10.1 Detailed Description

IMU9 (9 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included. Further, 3-axes from the magnetometer are included, which can be used to get a compass-like heading.

4.10.2 Macro Definition Documentation

4.10.2.1 CAER_IMU9_ITERATOR_ALL_END

```
#define CAER_IMU9_ITERATOR_ALL_END }
```

Iterator close statement.

4.10.2.2 CAER_IMU9_ITERATOR_ALL_START

```
#define CAER_IMU9_ITERATOR_ALL_START(
               IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0; \
        caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      IMU9_PACKET)->packetHeader); \
        caerIMU9IteratorCounter++) { \
        caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter);
```

Iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.10.2.3 CAER_IMU9_ITERATOR_VALID_END

```
#define CAER_IMU9_ITERATOR_VALID_END }
```

Iterator close statement.

4.10.2.4 CAER_IMU9_ITERATOR_VALID_START

```
#define CAER_IMU9_ITERATOR_VALID_START(
               IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0; \
        caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      IMU9_PACKET)->packetHeader); \
        caerIMU9IteratorCounter++) { \
        caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter); \
        if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) { continue; }
```

Iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.10.3 Typedef Documentation

4.10.3.1 caerIMU9Event

```
typedef struct caer_imu9_event* caerIMU9Event
```

Type for pointer to IMU 9-axes event data structure.

4.10.3.2 caerIMU9EventPacket

```
typedef struct caer_imu9_event_packet* caerIMU9EventPacket
```

Type for pointer to IMU 9-axes event packet data structure.

4.10.4 Function Documentation

4.10.4.1 caerIMU9EventGetAccelX()

```
static float caerIMU9EventGetAccelX (
            caerIMU9Event event )  [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the X axis.

4.10.4.2 caerIMU9EventGetAccelY()

```
static float caerIMU9EventGetAccelY (
            caerIMU9Event event )  [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the Y axis.

4.10.4.3 caerIMU9EventGetAccelZ()

```
static float caerIMU9EventGetAccelZ (
            caerIMU9Event event )  [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the Z axis.

4.10.4.4 caerIMU9EventGetCompX()

```
static float caerIMU9EventGetCompX (
            caerIMU9Event event )  [inline], [static]
```

Get the X axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

X axis compass heading.

4.10.4.5 caerIMU9EventGetCompY()

```
static float caerIMU9EventGetCompY (
            caerIMU9Event event )  [inline], [static]
```

Get the Y axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

Y axis compass heading.

4.10.4.6 caerIMU9EventGetCompZ()

```
static float caerIMU9EventGetCompZ (
            caerIMU9Event event )  [inline], [static]
```

Get the Z axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

Z axis compass heading.

4.10.4.7 caerIMU9EventGetGyroX()

```
static float caerIMU9EventGetGyroX (
            caerIMU9Event event )  [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the X axis (roll).

4.10.4.8 caerIMU9EventGetGyroY()

```
static float caerIMU9EventGetGyroY (
            caerIMU9Event event )  [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the Y axis (pitch).

4.10.4.9 caerIMU9EventGetGyroZ()

```
static float caerIMU9EventGetGyroZ (
            caerIMU9Event event )  [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the Z axis (yaw).

4.10.4.10 caerIMU9EventGetTemp()

```
static float caerIMU9EventGetTemp (
            caerIMU9Event event )  [inline], [static]
```

Get the temperature reading. This is in °C.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

temperature in °C.

4.10.4.11 caerIMU9EventGetTimestamp()

```
static int32_t caerIMU9EventGetTimestamp (
            caerIMU9Event event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.10.4.12 caerIMU9EventGetTimestamp64()

```
static int64_t caerIMU9EventGetTimestamp64 (
            caerIMU9Event event,
            caerIMU9EventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.10.4.13 caerIMU9EventInvalidate()

```
static void caerIMU9EventInvalidate (
            caerIMU9Event event,
            caerIMU9EventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.14 caerIMU9EventIsValid()

```
static bool caerIMU9EventIsValid (
            caerIMU9Event event )  [inline], [static]
```

Check if this IMU 9-axes event is valid.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.10.4.15 caerIMU9EventPacketAllocate()

```
caerIMU9EventPacket caerIMU9EventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new IMU 9-axes events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid IMU9EventPacket handle or NULL on error.

4.10.4.16 caerIMU9EventPacketGetEvent()

```
static caerIMU9Event caerIMU9EventPacketGetEvent (
            caerIMU9EventPacket packet,
            int32_t n )  [inline], [static]
```

Get the IMU 9-axes event at the given index from the event packet.

Parameters

packet	a valid IMU9EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 9-axes event. NULL on error.

4.10.4.17 caerIMU9EventSetAccelX()

```
static void caerIMU9EventSetAccelX (
            caerIMU9Event event,
            float accelX )  [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelX	acceleration on the X axis.

4.10.4.18 caerIMU9EventSetAccelY()

```
static void caerIMU9EventSetAccelY (
            caerIMU9Event event,
            float accelY )  [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelY	acceleration on the Y axis.

4.10.4.19 caerIMU9EventSetAccelZ()

```
static void caerIMU9EventSetAccelZ (
            caerIMU9Event event,
            float accelZ )  [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelZ	acceleration on the Z axis.

4.10.4.20 caerIMU9EventSetCompX()

```
static void caerIMU9EventSetCompX (
            caerIMU9Event event,
            float compX )  [inline], [static]
```

Set the X axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compX	X axis compass heading.

4.10.4.21 caerIMU9EventSetCompY()

```
static void caerIMU9EventSetCompY (
            caerIMU9Event event,
            float compY )  [inline], [static]
```

Set the Y axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compY	Y axis compass heading.

4.10.4.22 caerIMU9EventSetCompZ()

```
static void caerIMU9EventSetCompZ (
            caerIMU9Event event,
            float compZ )  [inline], [static]
```

Set the Z axis compass heading (from magnetometer). This is in µT.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compZ	Z axis compass heading.

4.10.4.23 caerIMU9EventSetGyroX()

```
static void caerIMU9EventSetGyroX (
            caerIMU9Event event,
            float gyroX )  [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroX	angular velocity on the X axis (roll).

4.10.4.24 caerIMU9EventSetGyroY()

```
static void caerIMU9EventSetGyroY (
            caerIMU9Event event,
            float gyroY )  [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroY	angular velocity on the Y axis (pitch).

4.10.4.25 caerIMU9EventSetGyroZ()

```
static void caerIMU9EventSetGyroZ (
            caerIMU9Event event,
            float gyroZ )  [inline], [static]
```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroZ	angular velocity on the Z axis (yaw).

4.10.4.26 caerIMU9EventSetTemp()

```
static void caerIMU9EventSetTemp (
```

```
            caerIMU9Event event,
            float temp )  [inline], [static]
```

Set the temperature reading. This is in °C.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
temp	temperature in °C.

4.10.4.27 caerIMU9EventSetTimestamp()

```
static void caerIMU9EventSetTimestamp (
            caerIMU9Event event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.10.4.28 caerIMU9EventValidate()

```
static void caerIMU9EventValidate (
            caerIMU9Event event,
            caerIMU9EventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.29 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float
```

```
accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float
comp_y;float comp_z;}  )
```

IMU 9-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature, and magnetometer readings. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.10.4.30 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct
caer_imu9_event events[];}  )
```

IMU 9-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.11 events/packetContainer.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(PACKET_CONTAINER)
- #define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END } }

Typedefs

- typedef struct caer_event_packet_container * caerEventPacketContainer

Functions

- PACKED_STRUCT (struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_t highestEventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPacketsNumber;caerEventPacketHeader eventPackets[];})
- caerEventPacketContainer caerEventPacketContainerAllocate (int32_t eventPacketsNumber)
- void caerEventPacketContainerFree (caerEventPacketContainer container)
- static int32_t caerEventPacketContainerGetEventPacketsNumber (caerEventPacketContainer container)
- static void caerEventPacketContainerSetEventPacketsNumber (caerEventPacketContainer container, int32_t eventPacketsNumber)
- static caerEventPacketHeader caerEventPacketContainerGetEventPacket (caerEventPacketContainer container, int32_t n)
- static void caerEventPacketContainerSetEventPacket (caerEventPacketContainer container, int32_t n, caerEventPacketHeader packetHeader)
- static int64_t caerEventPacketContainerGetLowestEventTimestamp (caerEventPacketContainer container)
- static int64_t caerEventPacketContainerGetHighestEventTimestamp (caerEventPacketContainer container)
- static int32_t caerEventPacketContainerGetEventsNumber (caerEventPacketContainer container)
- static int32_t caerEventPacketContainerGetEventsValidNumber (caerEventPacketContainer container)
- static caerEventPacketHeader caerEventPacketContainerFindEventPacketByType (caerEventPacketContainer container, int16_t typeID)
- static caerEventPacketContainer caerEventPacketContainerCopyAllEvents (caerEventPacketContainer container)
- static caerEventPacketContainer caerEventPacketContainerCopyValidEvents (caerEventPacketContainer container)

4.11.1 Detailed Description

EventPacketContainer format definition and handling functions. An EventPacketContainer is a logical construct that contains packets of events (EventPackets) of different event types, with the aim of keeping related events of differing types, such as DVS and IMU data, together. Such a relation is usually based on time intervals, trying to keep groups of event happening in a certain time-slice together. This time-order is based on the *main* time-stamp of an event, the one whose offset is referenced in the event packet header and that is used by the caerGenericEvent∗() functions. It's guaranteed that all conforming input modules keep to this rule, generating containers that include all events from all types within the given time-slice. The smallest and largest timestamps are tracked at the packet container level as a convenience, to avoid having to examine all packets for this often useful piece of information. All integers are in their native host format, as this is a purely internal, in-memory data structure, never meant for exchange between different systems (and different endianness).

== Packet Containers and Input Modules == The "packeting system" works in this way: events are accumulated by type in a packet, and that packet is part of a packet container, by an input module. The packet container is then sent out for processing when either the configured time limit or the size limit are hit. The time limit is always active, in microseconds, and basically tells you the time-span an event packet covers. This enables regular, constant delivery of packets, that cover a period of time. The size limit is an addon to prevent packets to grow to immense sizes (like if the time limit is high and there is lots of activity). As soon as a packet hits the number of events in the size limit, it is sent out. The regular time limit is not reset in this case. This size limit can be disabled by setting it to 0. The cAER DVS128/DAVIS/File/Network input modules call these two configuration variables "PacketContainerInterval" and "PacketContainerMaxPacketSize". Too small packet sizes or intervals simply mean more packets, which may negatively affect performance. It's usually a good idea to set the size to something around 4-8K, and the time to a good value based on the application you're building, so if you need ms-reaction-time, you probably want to set it to 1000µs, so that you do get new data every ms. If on the other hand you're looking at a static scene and just want to detect that something is passing by once every while, a higher number like 100ms might also be perfectly appropriate.

4.11.2 Macro Definition Documentation

4.11.2.1 CAER_EVENT_PACKET_CONTAINER_ITERATOR_END

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END } }
```

Iterator close statement.

4.11.2.2 CAER_EVENT_PACKET_CONTAINER_ITERATOR_START

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(
              PACKET_CONTAINER )
```

Value:

```
if ((PACKET_CONTAINER) != NULL) { \
        for (int32_t caerEventPacketContainerIteratorCounter = 0; \
            caerEventPacketContainerIteratorCounter <
      caerEventPacketContainerGetEventPacketsNumber(PACKET_CONTAINER
      ); \
            caerEventPacketContainerIteratorCounter++) { \
            caerEventPacketHeader caerEventPacketContainerIteratorElement =
      caerEventPacketContainerGetEventPacket(PACKET_CONTAINER,
      caerEventPacketContainerIteratorCounter); \
            if (caerEventPacketContainerIteratorElement == NULL) { continue; }
```

Iterator over all event packets in an event packet container. Returns the current index in the 'caerEventPacket↩ ContainerIteratorCounter' variable of type 'int32_t' and the current event packet in the 'caerEventPacketContainer↩ IteratorElement' variable of type caerEventPacketHeader. The current packet may be NULL, in which case it is skipped during iteration.

PACKET_CONTAINER: a valid EventPacketContainer handle. If NULL, no iteration is performed.

4.11.3 Typedef Documentation

4.11.3.1 caerEventPacketContainer

```
typedef struct caer_event_packet_container* caerEventPacketContainer
```

Type for pointer to EventPacketContainer data structure.

4.11.4 Function Documentation

4.11.4.1 caerEventPacketContainerAllocate()

```
caerEventPacketContainer caerEventPacketContainerAllocate (
            int32_t eventPacketsNumber )
```

Allocate a new EventPacketContainer with enough space to store up to the given number of EventPacket references. All packet references will be NULL initially.

Parameters

eventPacketsNumber	the maximum number of EventPacket references that can be stored in this container.

Returns

a valid EventPacketContainer handle or NULL on error.

4.11.4.2 caerEventPacketContainerCopyAllEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyAllEvents (
            caerEventPacketContainer container )  [inline], [static]
```

Make a deep copy of an event packet container and all of its event packets and their current events.

Parameters

container	an event packet container to copy.

Returns

a deep copy of an event packet container, containing all events.

4.11.4.3 caerEventPacketContainerCopyValidEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyValidEvents (
            caerEventPacketContainer container )  [inline], [static]
```

Make a deep copy of an event packet container, with its event packets sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

container	an event packet container to copy.

Returns

a deep copy of an event packet container, containing only valid events.

4.11.4.4 caerEventPacketContainerFindEventPacketByType()

```
static caerEventPacketHeader caerEventPacketContainerFindEventPacketByType (
            caerEventPacketContainer container,
            int16_t typeID )  [inline], [static]
```

Get the reference for an EventPacket stored in this container with the given event type. This returns the first found event packet with that type ID, or NULL if we get to the end without finding any such event packet.

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
typeID	the event type to search for.

Returns

a reference to an EventPacket with a certain type or NULL if none found.

4.11.4.5 caerEventPacketContainerFree()

```
void caerEventPacketContainerFree (
            caerEventPacketContainer container )
```

Free the memory occupied by an EventPacketContainer, as well as freeing all of its contained EventPackets and their memory. If you don't want the contained EventPackets to be freed, make sure that you set their reference to NULL before calling this.

Parameters

container	the container to be freed.

4.11.4.6 caerEventPacketContainerGetEventPacket()

```
static caerEventPacketHeader caerEventPacketContainerGetEventPacket (
            caerEventPacketContainer container,
            int32_t n )  [inline], [static]
```

Get the reference for the EventPacket stored in this container at the given index.

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
n	the index of the EventPacket to get.

Returns

a reference to an EventPacket or NULL on error.

4.11.4.7 caerEventPacketContainerGetEventPacketsNumber()

```
static int32_t caerEventPacketContainerGetEventPacketsNumber (
            caerEventPacketContainer container )  [inline], [static]
```

Get the maximum number of EventPacket references that can be stored in this particular EventPacketContainer.

Parameters

container	a valid EventPacketContainer handle. If NULL, zero is returned.

Returns

the number of EventPacket references that can be contained.

4.11.4.8 caerEventPacketContainerGetEventsNumber()

```
static int32_t caerEventPacketContainerGetEventsNumber (
            caerEventPacketContainer container )  [inline], [static]
```

Get the number of events contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, 0 is returned.

Returns

the number of events in this container.

4.11.4.9 caerEventPacketContainerGetEventsValidNumber()

```
static int32_t caerEventPacketContainerGetEventsValidNumber (
            caerEventPacketContainer container )  [inline], [static]
```

Get the number of valid events contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, 0 is returned.

Returns

the number of valid events in this container.

4.11.4.10 caerEventPacketContainerGetHighestEventTimestamp()

```
static int64_t caerEventPacketContainerGetHighestEventTimestamp (
            caerEventPacketContainer container )  [inline], [static]
```

Get the highest timestamp contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, -1 is returned.

Returns

the highest timestamp (in µs) or -1 if not initialized.

4.11.4.11 caerEventPacketContainerGetLowestEventTimestamp()

```
static int64_t caerEventPacketContainerGetLowestEventTimestamp (
            caerEventPacketContainer container )  [inline], [static]
```

Get the lowest timestamp contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, -1 is returned.

Returns

the lowest timestamp (in µs) or -1 if not initialized.

4.11.4.12 caerEventPacketContainerSetEventPacket()

```
static void caerEventPacketContainerSetEventPacket (
            caerEventPacketContainer container,
            int32_t n,
            caerEventPacketHeader packetHeader )  [inline], [static]
```

Set the reference for the EventPacket stored in this container at the given index.

Parameters

container	a valid EventPacketContainer handle. If NULL, nothing happens.
n	the index of the EventPacket to set.
packetHeader	a reference to an EventPacket's header. Can be NULL.

4.11.4.13 caerEventPacketContainerSetEventPacketsNumber()

```
static void caerEventPacketContainerSetEventPacketsNumber (
            caerEventPacketContainer container,
            int32_t eventPacketsNumber )  [inline], [static]
```

Set the maximum number of EventPacket references that can be stored in this particular EventPacketContainer. This should never be used directly, caerEventPacketContainerAllocate() sets this for you.

Parameters

container	a valid EventPacketContainer handle. If NULL, nothing happens.
eventPacketsNumber	the number of EventPacket references that can be contained.

4.11.4.14 PACKED_STRUCT()

```
PACKED_STRUCT (
            struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_t highest↩
EventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPacketsNumber;caer↩
EventPacketHeader eventPackets[];}  )
```

EventPacketContainer data structure definition. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

4.12 events/point1d.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_POINT1D_ITERATOR_ALL_START(POINT1D_PACKET)
- #define CAER_POINT1D_ITERATOR_ALL_END }
- #define CAER_POINT1D_ITERATOR_VALID_START(POINT1D_PACKET)
- #define CAER_POINT1D_ITERATOR_VALID_END }

- #define POINT1D_TYPE_SHIFT 1
- #define POINT1D_TYPE_MASK 0x0000007F
- #define POINT1D_SCALE_SHIFT 8
- #define POINT1D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point1d_event * caerPoint1DEvent
- typedef struct caer_point1d_event_packet * caerPoint1DEventPacket

Functions

- PACKED_STRUCT (struct caer_point1d_event { uint32_t info;float x;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point1d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point1d_event events[ ];})
- caerPoint1DEventPacket caerPoint1DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint1DEvent caerPoint1DEventPacketGetEvent (caerPoint1DEventPacket packet, int32_t n)
- static int32_t caerPoint1DEventGetTimestamp (caerPoint1DEvent event)
- static int64_t caerPoint1DEventGetTimestamp64 (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static void caerPoint1DEventSetTimestamp (caerPoint1DEvent event, int32_t timestamp)
- static bool caerPoint1DEventIsValid (caerPoint1DEvent event)
- static void caerPoint1DEventValidate (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static void caerPoint1DEventInvalidate (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static uint8_t caerPoint1DEventGetType (caerPoint1DEvent event)
- static void caerPoint1DEventSetType (caerPoint1DEvent event, uint8_t type)
- static int8_t caerPoint1DEventGetScale (caerPoint1DEvent event)
- static void caerPoint1DEventSetScale (caerPoint1DEvent event, int8_t scale)
- static float caerPoint1DEventGetX (caerPoint1DEvent event)
- static void caerPoint1DEventSetX (caerPoint1DEvent event, float x)

4.12.1 Detailed Description

THIS EVENT DEFINITIONS IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE CHANGES AND REVISIONS!

Point1D Events format definition and handling functions. This contains one dimensional data points as floats, together with support for distinguishing type and scale.

4.12.2 Macro Definition Documentation

4.12.2.1 CAER_POINT1D_ITERATOR_ALL_END

```
#define CAER_POINT1D_ITERATOR_ALL_END }
```

Iterator close statement.

4.12.2.2 CAER_POINT1D_ITERATOR_ALL_START

```
#define CAER_POINT1D_ITERATOR_ALL_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) { \
        caerPoint1DEvent caerPoint1DIteratorElement =
      caerPoint1DEventPacketGetEvent(POINT1D_PACKET, caerPoint1DIteratorCounter);
```

Iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.12.2.3 CAER_POINT1D_ITERATOR_VALID_END

```
#define CAER_POINT1D_ITERATOR_VALID_END }
```

Iterator close statement.

4.12.2.4 CAER_POINT1D_ITERATOR_VALID_START

```
#define CAER_POINT1D_ITERATOR_VALID_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) { \
        caerPoint1DEvent caerPoint1DIteratorElement =
      caerPoint1DEventPacketGetEvent(POINT1D_PACKET, caerPoint1DIteratorCounter); \
        if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) { continue; }
```

Iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.12.2.5 POINT1D_SCALE_MASK

```
#define POINT1D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.6 POINT1D_SCALE_SHIFT

```
#define POINT1D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.7 POINT1D_TYPE_MASK

```
#define POINT1D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.8 POINT1D_TYPE_SHIFT

```
#define POINT1D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.3 Typedef Documentation

4.12.3.1 caerPoint1DEvent

```
typedef struct caer_point1d_event* caerPoint1DEvent
```

Type for pointer to Point1D event data structure.

4.12.3.2 caerPoint1DEventPacket

```
typedef struct caer_point1d_event_packet* caerPoint1DEventPacket
```

Type for pointer to Point1D event packet data structure.

4.12.4 Function Documentation

4.12.4.1 caerPoint1DEventGetScale()

```
static int8_t caerPoint1DEventGetScale (
            caerPoint1DEvent event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

the Point1D measurement scale.

4.12.4.2 caerPoint1DEventGetTimestamp()

```
static int32_t caerPoint1DEventGetTimestamp (
            caerPoint1DEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.12.4.3 caerPoint1DEventGetTimestamp64()

```
static int64_t caerPoint1DEventGetTimestamp64 (
            caerPoint1DEvent event,
            caerPoint1DEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.12.4.4 caerPoint1DEventGetType()

```
static uint8_t caerPoint1DEventGetType (
            caerPoint1DEvent event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

the Point1D measurement type.

4.12.4.5 caerPoint1DEventGetX()

```
static float caerPoint1DEventGetX (
              caerPoint1DEvent event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.12.4.6 caerPoint1DEventInvalidate()

```
static void caerPoint1DEventInvalidate (
              caerPoint1DEvent event,
              caerPoint1DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.4.7 caerPoint1DEventIsValid()

```
static bool caerPoint1DEventIsValid (
              caerPoint1DEvent event )  [inline], [static]
```

Check if this Point1D event is valid.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.12.4.8 caerPoint1DEventPacketAllocate()

```
caerPoint1DEventPacket caerPoint1DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new Point1D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point1DEventPacket handle or NULL on error.

4.12.4.9 caerPoint1DEventPacketGetEvent()

```
static caerPoint1DEvent caerPoint1DEventPacketGetEvent (
            caerPoint1DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point1D event at the given index from the event packet.

Parameters

packet	a valid Point1DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point1D event. NULL on error.

4.12.4.10 caerPoint1DEventSetScale()

```
static void caerPoint1DEventSetScale (
            caerPoint1DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
scale	the Point1D measurement scale.

4.12.4.11 caerPoint1DEventSetTimestamp()

```
static void caerPoint1DEventSetTimestamp (
             caerPoint1DEvent event,
             int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.12.4.12 caerPoint1DEventSetType()

```
static void caerPoint1DEventSetType (
             caerPoint1DEvent event,
             uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
type	the Point1D measurement type.

4.12.4.13 caerPoint1DEventSetX()

```
static void caerPoint1DEventSetX (
             caerPoint1DEvent event,
             float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.12.4.14 caerPoint1DEventValidate()

```
static void caerPoint1DEventValidate (
            caerPoint1DEvent event,
            caerPoint1DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.4.15 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point1d_event { uint32_t info;float x;int32_t timestamp;}  )
```

Point1D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The one measurement (x) is stored as a float. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.12.4.16 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point1d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point1d_event events[];}  )
```

Point1D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.13 events/point2d.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_POINT2D_ITERATOR_ALL_START(POINT2D_PACKET)
- #define CAER_POINT2D_ITERATOR_ALL_END }
- #define CAER_POINT2D_ITERATOR_VALID_START(POINT2D_PACKET)
- #define CAER_POINT2D_ITERATOR_VALID_END }

- #define POINT2D_TYPE_SHIFT 1
- #define POINT2D_TYPE_MASK 0x0000007F
- #define POINT2D_SCALE_SHIFT 8
- #define POINT2D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point2d_event * caerPoint2DEvent
- typedef struct caer_point2d_event_packet * caerPoint2DEventPacket

Functions

- PACKED_STRUCT (struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point2d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point2d_event events[ ];})
- caerPoint2DEventPacket caerPoint2DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint2DEvent caerPoint2DEventPacketGetEvent (caerPoint2DEventPacket packet, int32_t n)
- static int32_t caerPoint2DEventGetTimestamp (caerPoint2DEvent event)
- static int64_t caerPoint2DEventGetTimestamp64 (caerPoint2DEvent event, caerPoint2DEventPacket packet)
- static void caerPoint2DEventSetTimestamp (caerPoint2DEvent event, int32_t timestamp)
- static bool caerPoint2DEventIsValid (caerPoint2DEvent event)
- static void caerPoint2DEventValidate (caerPoint2DEvent event, caerPoint2DEventPacket packet)
- static void caerPoint2DEventInvalidate (caerPoint2DEvent event, caerPoint2DEventPacket packet)
- static uint8_t caerPoint2DEventGetType (caerPoint2DEvent event)
- static void caerPoint2DEventSetType (caerPoint2DEvent event, uint8_t type)
- static int8_t caerPoint2DEventGetScale (caerPoint2DEvent event)
- static void caerPoint2DEventSetScale (caerPoint2DEvent event, int8_t scale)
- static float caerPoint2DEventGetX (caerPoint2DEvent event)
- static void caerPoint2DEventSetX (caerPoint2DEvent event, float x)
- static float caerPoint2DEventGetY (caerPoint2DEvent event)
- static void caerPoint2DEventSetY (caerPoint2DEvent event, float y)

4.13.1 Detailed Description

THIS EVENT DEFINITIONS IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE CHANGES AND REVISIONS!

Point2D Events format definition and handling functions. This contains two dimensional data points as floats, together with support for distinguishing type and scale.

4.13.2 Macro Definition Documentation

4.13.2.1 CAER_POINT2D_ITERATOR_ALL_END

```
#define CAER_POINT2D_ITERATOR_ALL_END }
```

Iterator close statement.

4.13.2.2 CAER_POINT2D_ITERATOR_ALL_START

```
#define CAER_POINT2D_ITERATOR_ALL_START(
              POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0; \
       caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT2D_PACKET)->packetHeader); \
       caerPoint2DIteratorCounter++) { \
       caerPoint2DEvent caerPoint2DIteratorElement =
     caerPoint2DEventPacketGetEvent(POINT2D_PACKET, caerPoint2DIteratorCounter);
```

Iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.13.2.3 CAER_POINT2D_ITERATOR_VALID_END

```
#define CAER_POINT2D_ITERATOR_VALID_END }
```

Iterator close statement.

4.13.2.4 CAER_POINT2D_ITERATOR_VALID_START

```
#define CAER_POINT2D_ITERATOR_VALID_START(
              POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0; \
       caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT2D_PACKET)->packetHeader); \
       caerPoint2DIteratorCounter++) { \
       caerPoint2DEvent caerPoint2DIteratorElement =
     caerPoint2DEventPacketGetEvent(POINT2D_PACKET, caerPoint2DIteratorCounter); \
       if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) { continue; }
```

Iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.13.2.5 POINT2D_SCALE_MASK

```
#define POINT2D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.13.2.6 POINT2D_SCALE_SHIFT

```
#define POINT2D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.13.2.7 POINT2D_TYPE_MASK

```
#define POINT2D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.13.2.8 POINT2D_TYPE_SHIFT

```
#define POINT2D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.13.3 Typedef Documentation

4.13.3.1 caerPoint2DEvent

```
typedef struct caer_point2d_event* caerPoint2DEvent
```

Type for pointer to Point2D event data structure.

4.13.3.2 caerPoint2DEventPacket

```
typedef struct caer_point2d_event_packet* caerPoint2DEventPacket
```

Type for pointer to Point2D event packet data structure.

4.13.4 Function Documentation

4.13.4.1 caerPoint2DEventGetScale()

```
static int8_t caerPoint2DEventGetScale (
            caerPoint2DEvent event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

the Point2D measurement scale.

4.13.4.2 caerPoint2DEventGetTimestamp()

```
static int32_t caerPoint2DEventGetTimestamp (
            caerPoint2DEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.13.4.3 caerPoint2DEventGetTimestamp64()

```
static int64_t caerPoint2DEventGetTimestamp64 (
            caerPoint2DEvent event,
            caerPoint2DEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.13.4.4 caerPoint2DEventGetType()

```
static uint8_t caerPoint2DEventGetType (
            caerPoint2DEvent event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

the Point2D measurement type.

4.13.4.5 caerPoint2DEventGetX()

```
static float caerPoint2DEventGetX (
            caerPoint2DEvent event ) [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.13.4.6 caerPoint2DEventGetY()

```
static float caerPoint2DEventGetY (
            caerPoint2DEvent event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.13.4.7 caerPoint2DEventInvalidate()

```
static void caerPoint2DEventInvalidate (
            caerPoint2DEvent event,
            caerPoint2DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.8 caerPoint2DEventIsValid()

```
static bool caerPoint2DEventIsValid (
            caerPoint2DEvent event )  [inline], [static]
```

Check if this Point2D event is valid.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.13.4.9 caerPoint2DEventPacketAllocate()

```
caerPoint2DEventPacket caerPoint2DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new Point2D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point2DEventPacket handle or NULL on error.

4.13.4.10 caerPoint2DEventPacketGetEvent()

```
static caerPoint2DEvent caerPoint2DEventPacketGetEvent (
            caerPoint2DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point2D event at the given index from the event packet.

Parameters

packet	a valid Point2DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point2D event. NULL on error.

4.13.4.11 caerPoint2DEventSetScale()

```
static void caerPoint2DEventSetScale (
            caerPoint2DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
scale	the Point2D measurement scale.

4.13.4.12 caerPoint2DEventSetTimestamp()

```
static void caerPoint2DEventSetTimestamp (
              caerPoint2DEvent event,
              int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.13.4.13 caerPoint2DEventSetType()

```
static void caerPoint2DEventSetType (
              caerPoint2DEvent event,
              uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
type	the Point2D measurement type.

4.13.4.14 caerPoint2DEventSetX()

```
static void caerPoint2DEventSetX (
              caerPoint2DEvent event,
              float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.13.4.15 caerPoint2DEventSetY()

```
static void caerPoint2DEventSetY (
            caerPoint2DEvent event,
            float y )  [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.13.4.16 caerPoint2DEventValidate()

```
static void caerPoint2DEventValidate (
            caerPoint2DEvent event,
            caerPoint2DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.17 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;}  )
```

Point2D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The two measurements (x, y) are stored as floats. Floats are in IE↩ EE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.13.4.18 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point2d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point2d_event events[];}  )
```

Point2D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.14 events/point3d.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_POINT3D_ITERATOR_ALL_START(POINT3D_PACKET)
- #define CAER_POINT3D_ITERATOR_ALL_END }
- #define CAER_POINT3D_ITERATOR_VALID_START(POINT3D_PACKET)
- #define CAER_POINT3D_ITERATOR_VALID_END }

- #define POINT3D_TYPE_SHIFT 1
- #define POINT3D_TYPE_MASK 0x0000007F
- #define POINT3D_SCALE_SHIFT 8
- #define POINT3D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point3d_event * caerPoint3DEvent
- typedef struct caer_point3d_event_packet * caerPoint3DEventPacket

Functions

- PACKED_STRUCT (struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point3d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point3d_event events[];})
- caerPoint3DEventPacket caerPoint3DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint3DEvent caerPoint3DEventPacketGetEvent (caerPoint3DEventPacket packet, int32_t n)
- static int32_t caerPoint3DEventGetTimestamp (caerPoint3DEvent event)
- static int64_t caerPoint3DEventGetTimestamp64 (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static void caerPoint3DEventSetTimestamp (caerPoint3DEvent event, int32_t timestamp)
- static bool caerPoint3DEventIsValid (caerPoint3DEvent event)
- static void caerPoint3DEventValidate (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static void caerPoint3DEventInvalidate (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static uint8_t caerPoint3DEventGetType (caerPoint3DEvent event)
- static void caerPoint3DEventSetType (caerPoint3DEvent event, uint8_t type)
- static int8_t caerPoint3DEventGetScale (caerPoint3DEvent event)
- static void caerPoint3DEventSetScale (caerPoint3DEvent event, int8_t scale)
- static float caerPoint3DEventGetX (caerPoint3DEvent event)
- static void caerPoint3DEventSetX (caerPoint3DEvent event, float x)
- static float caerPoint3DEventGetY (caerPoint3DEvent event)
- static void caerPoint3DEventSetY (caerPoint3DEvent event, float y)
- static float caerPoint3DEventGetZ (caerPoint3DEvent event)
- static void caerPoint3DEventSetZ (caerPoint3DEvent event, float z)

4.14.1 Detailed Description

THIS EVENT DEFINITIONS IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE CHANGES AND REVISIONS!

Point3D Events format definition and handling functions. This contains three dimensional data points as floats, together with support for distinguishing type and scale.

4.14.2 Macro Definition Documentation

4.14.2.1 CAER_POINT3D_ITERATOR_ALL_END

```
#define CAER_POINT3D_ITERATOR_ALL_END }
```

Iterator close statement.

4.14.2.2 CAER_POINT3D_ITERATOR_ALL_START

```
#define CAER_POINT3D_ITERATOR_ALL_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0; \
        caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
        caerPoint3DIteratorCounter++) { \
        caerPoint3DEvent caerPoint3DIteratorElement =
      caerPoint3DEventPacketGetEvent(POINT3D_PACKET, caerPoint3DIteratorCounter);
```

Iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.14.2.3 CAER_POINT3D_ITERATOR_VALID_END

```
#define CAER_POINT3D_ITERATOR_VALID_END }
```

Iterator close statement.

4.14.2.4 CAER_POINT3D_ITERATOR_VALID_START

```
#define CAER_POINT3D_ITERATOR_VALID_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0; \
        caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
        caerPoint3DIteratorCounter++) { \
        caerPoint3DEvent caerPoint3DIteratorElement =
      caerPoint3DEventPacketGetEvent(POINT3D_PACKET, caerPoint3DIteratorCounter); \
        if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) { continue; }
```

Iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.14.2.5 POINT3D_SCALE_MASK

```
#define POINT3D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.14.2.6 POINT3D_SCALE_SHIFT

```
#define POINT3D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.14.2.7 POINT3D_TYPE_MASK

```
#define POINT3D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.14.2.8 POINT3D_TYPE_SHIFT

```
#define POINT3D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.14.3 Typedef Documentation

4.14.3.1 caerPoint3DEvent

```
typedef struct caer_point3d_event* caerPoint3DEvent
```

Type for pointer to Point3D event data structure.

4.14.3.2 caerPoint3DEventPacket

```
typedef struct caer_point3d_event_packet* caerPoint3DEventPacket
```

Type for pointer to Point3D event packet data structure.

4.14.4 Function Documentation

4.14.4.1 caerPoint3DEventGetScale()

```
static int8_t caerPoint3DEventGetScale (
            caerPoint3DEvent event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

the Point3D measurement scale.

4.14.4.2 caerPoint3DEventGetTimestamp()

```
static int32_t caerPoint3DEventGetTimestamp (
            caerPoint3DEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.14.4.3 caerPoint3DEventGetTimestamp64()

```
static int64_t caerPoint3DEventGetTimestamp64 (
            caerPoint3DEvent event,
            caerPoint3DEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.14.4.4 caerPoint3DEventGetType()

```
static uint8_t caerPoint3DEventGetType (
            caerPoint3DEvent event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

the Point3D measurement type.

4.14.4.5 caerPoint3DEventGetX()

```
static float caerPoint3DEventGetX (
            caerPoint3DEvent event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.14.4.6 caerPoint3DEventGetY()

```
static float caerPoint3DEventGetY (
            caerPoint3DEvent event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.14.4.7 caerPoint3DEventGetZ()

```
static float caerPoint3DEventGetZ (
            caerPoint3DEvent event )  [inline], [static]
```

Get the Z axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

Z axis measurement.

4.14.4.8 caerPoint3DEventInvalidate()

```
static void caerPoint3DEventInvalidate (
            caerPoint3DEvent event,
            caerPoint3DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.9 caerPoint3DEventIsValid()

```
static bool caerPoint3DEventIsValid (
            caerPoint3DEvent event )  [inline], [static]
```

Check if this Point3D event is valid.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.14.4.10 caerPoint3DEventPacketAllocate()

```
caerPoint3DEventPacket caerPoint3DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new Point3D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point3DEventPacket handle or NULL on error.

4.14.4.11 caerPoint3DEventPacketGetEvent()

```
static caerPoint3DEvent caerPoint3DEventPacketGetEvent (
            caerPoint3DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point3D event at the given index from the event packet.

Parameters

packet	a valid Point3DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point3D event. NULL on error.

4.14.4.12 caerPoint3DEventSetScale()

```
static void caerPoint3DEventSetScale (
            caerPoint3DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
scale	the Point3D measurement scale.

4.14.4.13 caerPoint3DEventSetTimestamp()

```
static void caerPoint3DEventSetTimestamp (
            caerPoint3DEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.14.4.14 caerPoint3DEventSetType()

```
static void caerPoint3DEventSetType (
              caerPoint3DEvent event,
              uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
type	the Point3D measurement type.

4.14.4.15 caerPoint3DEventSetX()

```
static void caerPoint3DEventSetX (
              caerPoint3DEvent event,
              float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.14.4.16 caerPoint3DEventSetY()

```
static void caerPoint3DEventSetY (
              caerPoint3DEvent event,
              float y )  [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.14.4.17 caerPoint3DEventSetZ()

```
static void caerPoint3DEventSetZ (
            caerPoint3DEvent event,
            float z )  [inline], [static]
```

Set the Z axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
z	Z axis measurement.

4.14.4.18 caerPoint3DEventValidate()

```
static void caerPoint3DEventValidate (
            caerPoint3DEvent event,
            caerPoint3DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.19 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;}
)
```

Point3D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The three measurements (x, y, z) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.14.4.20 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point3d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point3d_event events[];}  )
```

Point3D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.15 events/point4d.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_POINT4D_ITERATOR_ALL_START(POINT4D_PACKET)
- #define CAER_POINT4D_ITERATOR_ALL_END }
- #define CAER_POINT4D_ITERATOR_VALID_START(POINT4D_PACKET)
- #define CAER_POINT4D_ITERATOR_VALID_END }

- #define POINT4D_TYPE_SHIFT 1
- #define POINT4D_TYPE_MASK 0x0000007F
- #define POINT4D_SCALE_SHIFT 8
- #define POINT4D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point4d_event * caerPoint4DEvent
- typedef struct caer_point4d_event_packet * caerPoint4DEventPacket

Functions

- PACKED_STRUCT (struct caer_point4d_event { uint32_t info;float x;float y;float z;float w;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point4d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point4d_event events[];})
- caerPoint4DEventPacket caerPoint4DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint4DEvent caerPoint4DEventPacketGetEvent (caerPoint4DEventPacket packet, int32_t n)
- static int32_t caerPoint4DEventGetTimestamp (caerPoint4DEvent event)
- static int64_t caerPoint4DEventGetTimestamp64 (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static void caerPoint4DEventSetTimestamp (caerPoint4DEvent event, int32_t timestamp)
- static bool caerPoint4DEventIsValid (caerPoint4DEvent event)
- static void caerPoint4DEventValidate (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static void caerPoint4DEventInvalidate (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static uint8_t caerPoint4DEventGetType (caerPoint4DEvent event)

- static void caerPoint4DEventSetType (caerPoint4DEvent event, uint8_t type)
- static int8_t caerPoint4DEventGetScale (caerPoint4DEvent event)
- static void caerPoint4DEventSetScale (caerPoint4DEvent event, int8_t scale)
- static float caerPoint4DEventGetX (caerPoint4DEvent event)
- static void caerPoint4DEventSetX (caerPoint4DEvent event, float x)
- static float caerPoint4DEventGetY (caerPoint4DEvent event)
- static void caerPoint4DEventSetY (caerPoint4DEvent event, float y)
- static float caerPoint4DEventGetZ (caerPoint4DEvent event)
- static void caerPoint4DEventSetZ (caerPoint4DEvent event, float z)
- static float caerPoint4DEventGetW (caerPoint4DEvent event)
- static void caerPoint4DEventSetW (caerPoint4DEvent event, float w)

4.15.1 Detailed Description

THIS EVENT DEFINITION IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE C↩ HANGES AND REVISIONS!

Point4D Events format definition and handling functions. This contains four dimensional data points as floats, together with support for distinguishing type and scale. Useful for homogeneous coordinates for example.

4.15.2 Macro Definition Documentation

4.15.2.1 CAER_POINT4D_ITERATOR_ALL_END

```
#define CAER_POINT4D_ITERATOR_ALL_END }
```

Iterator close statement.

4.15.2.2 CAER_POINT4D_ITERATOR_ALL_START

```
#define CAER_POINT4D_ITERATOR_ALL_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0; \
       caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
       caerPoint4DIteratorCounter++) { \
       caerPoint4DEvent caerPoint4DIteratorElement =
      caerPoint4DEventPacketGetEvent(POINT4D_PACKET, caerPoint4DIteratorCounter);
```

Iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.15.2.3 CAER_POINT4D_ITERATOR_VALID_END

```
#define CAER_POINT4D_ITERATOR_VALID_END }
```

Iterator close statement.

4.15.2.4 CAER_POINT4D_ITERATOR_VALID_START

```
#define CAER_POINT4D_ITERATOR_VALID_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0; \
        caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
        caerPoint4DIteratorCounter++) { \
        caerPoint4DEvent caerPoint4DIteratorElement =
      caerPoint4DEventPacketGetEvent(POINT4D_PACKET, caerPoint4DIteratorCounter); \
        if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) { continue; }
```

Iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.15.2.5 POINT4D_SCALE_MASK

```
#define POINT4D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.6 POINT4D_SCALE_SHIFT

```
#define POINT4D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.7 POINT4D_TYPE_MASK

```
#define POINT4D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.8 POINT4D_TYPE_SHIFT

```
#define POINT4D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.3 Typedef Documentation

4.15.3.1 caerPoint4DEvent

```
typedef struct caer_point4d_event* caerPoint4DEvent
```

Type for pointer to Point4D event data structure.

4.15.3.2 caerPoint4DEventPacket

```
typedef struct caer_point4d_event_packet* caerPoint4DEventPacket
```

Type for pointer to Point4D event packet data structure.

4.15.4 Function Documentation

4.15.4.1 caerPoint4DEventGetScale()

```
static int8_t caerPoint4DEventGetScale (
            caerPoint4DEvent event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

the Point4D measurement scale.

4.15.4.2 caerPoint4DEventGetTimestamp()

```
static int32_t caerPoint4DEventGetTimestamp (
            caerPoint4DEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.15.4.3 caerPoint4DEventGetTimestamp64()

```
static int64_t caerPoint4DEventGetTimestamp64 (
            caerPoint4DEvent event,
            caerPoint4DEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.15.4.4 caerPoint4DEventGetType()

```
static uint8_t caerPoint4DEventGetType (
            caerPoint4DEvent event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

the Point4D measurement type.

4.15.4.5 caerPoint4DEventGetW()

```
static float caerPoint4DEventGetW (
            caerPoint4DEvent event )  [inline], [static]
```

Get the W axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

W axis measurement.

4.15.4.6 caerPoint4DEventGetX()

```
static float caerPoint4DEventGetX (
            caerPoint4DEvent event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.15.4.7 caerPoint4DEventGetY()

```
static float caerPoint4DEventGetY (
            caerPoint4DEvent event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.15.4.8 caerPoint4DEventGetZ()

```
static float caerPoint4DEventGetZ (
            caerPoint4DEvent event )  [inline], [static]
```

Get the Z axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

Z axis measurement.

4.15.4.9 caerPoint4DEventInvalidate()

```
static void caerPoint4DEventInvalidate (
            caerPoint4DEvent event,
            caerPoint4DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.10 caerPoint4DEventIsValid()

```
static bool caerPoint4DEventIsValid (
            caerPoint4DEvent event )  [inline], [static]
```

Check if this Point4D event is valid.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.15.4.11 caerPoint4DEventPacketAllocate()

```
caerPoint4DEventPacket caerPoint4DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new Point4D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point4DEventPacket handle or NULL on error.

4.15.4.12 caerPoint4DEventPacketGetEvent()

```
static caerPoint4DEvent caerPoint4DEventPacketGetEvent (
            caerPoint4DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point4D event at the given index from the event packet.

Parameters

packet	a valid Point4DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point4D event. NULL on error.

4.15.4.13 caerPoint4DEventSetScale()

```
static void caerPoint4DEventSetScale (
            caerPoint4DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
scale	the Point4D measurement scale.

4.15.4.14 caerPoint4DEventSetTimestamp()

```
static void caerPoint4DEventSetTimestamp (
            caerPoint4DEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.15.4.15 caerPoint4DEventSetType()

```
static void caerPoint4DEventSetType (
            caerPoint4DEvent event,
            uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
type	the Point4D measurement type.

4.15.4.16 caerPoint4DEventSetW()

```
static void caerPoint4DEventSetW (
              caerPoint4DEvent event,
              float w )  [inline], [static]
```

Set the W axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
w	W axis measurement.

4.15.4.17 caerPoint4DEventSetX()

```
static void caerPoint4DEventSetX (
              caerPoint4DEvent event,
              float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.15.4.18 caerPoint4DEventSetY()

```
static void caerPoint4DEventSetY (
              caerPoint4DEvent event,
              float y )  [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.15.4.19 caerPoint4DEventSetZ()

```
static void caerPoint4DEventSetZ (
              caerPoint4DEvent event,
              float z )  [inline], [static]
```

Set the Z axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
z	Z axis measurement.

4.15.4.20 caerPoint4DEventValidate()

```
static void caerPoint4DEventValidate (
            caerPoint4DEvent event,
            caerPoint4DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.21 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point4d_event { uint32_t info;float x;float y;float z;float w;int32_t
timestamp;}  )
```

Point4D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The four measurements (x, y, z, w) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.15.4.22 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point4d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point4d_event events[];}  )
```

Point4D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.16 events/polarity.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_POLARITY_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_ITERATOR_ALL_END }
- #define CAER_POLARITY_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_ITERATOR_VALID_END }
- #define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }
- #define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }

- #define POLARITY_SHIFT 1
- #define POLARITY_MASK 0x00000001
- #define Y_ADDR_SHIFT 2
- #define Y_ADDR_MASK 0x00007FFF
- #define X_ADDR_SHIFT 17
- #define X_ADDR_MASK 0x00007FFF

Typedefs

- typedef struct caer_polarity_event * caerPolarityEvent
- typedef struct caer_polarity_event_packet * caerPolarityEventPacket

Functions

- PACKED_STRUCT (struct caer_polarity_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_polarity_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_polarity_event events[ ];})
- caerPolarityEventPacket caerPolarityEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPolarityEvent caerPolarityEventPacketGetEvent (caerPolarityEventPacket packet, int32_t n)
- static int32_t caerPolarityEventGetTimestamp (caerPolarityEvent event)
- static int64_t caerPolarityEventGetTimestamp64 (caerPolarityEvent event, caerPolarityEventPacket packet)
- static void caerPolarityEventSetTimestamp (caerPolarityEvent event, int32_t timestamp)
- static bool caerPolarityEventIsValid (caerPolarityEvent event)
- static void caerPolarityEventValidate (caerPolarityEvent event, caerPolarityEventPacket packet)
- static void caerPolarityEventInvalidate (caerPolarityEvent event, caerPolarityEventPacket packet)
- static bool caerPolarityEventGetPolarity (caerPolarityEvent event)
- static void caerPolarityEventSetPolarity (caerPolarityEvent event, bool polarity)
- static uint16_t caerPolarityEventGetY (caerPolarityEvent event)
- static void caerPolarityEventSetY (caerPolarityEvent event, uint16_t yAddress)
- static uint16_t caerPolarityEventGetX (caerPolarityEvent event)
- static void caerPolarityEventSetX (caerPolarityEvent event, uint16_t xAddress)

4.16.1 Detailed Description

Polarity Events format definition and handling functions. This event contains change information, with an X/Y address and an ON/OFF polarity. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics.

4.16.2 Macro Definition Documentation

4.16.2.1 CAER_POLARITY_ITERATOR_ALL_END

```
#define CAER_POLARITY_ITERATOR_ALL_END }
```

Iterator close statement.

4.16.2.2 CAER_POLARITY_ITERATOR_ALL_START

```
#define CAER_POLARITY_ITERATOR_ALL_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0; \
        caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
        caerPolarityIteratorCounter++) { \
        caerPolarityEvent caerPolarityIteratorElement =
      caerPolarityEventPacketGetEvent(POLARITY_PACKET, caerPolarityIteratorCounter
      );
```

Iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.16.2.3 CAER_POLARITY_ITERATOR_VALID_END

```
#define CAER_POLARITY_ITERATOR_VALID_END }
```

Iterator close statement.

4.16.2.4 CAER_POLARITY_ITERATOR_VALID_START

```
#define CAER_POLARITY_ITERATOR_VALID_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0; \
        caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
        caerPolarityIteratorCounter++) { \
        caerPolarityEvent caerPolarityIteratorElement =
      caerPolarityEventPacketGetEvent(POLARITY_PACKET, caerPolarityIteratorCounter
      ); \
        if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) { continue; }
```

Iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.16.2.5 CAER_POLARITY_REVERSE_ITERATOR_ALL_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.16.2.6 CAER_POLARITY_REVERSE_ITERATOR_ALL_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader) - 1; \
        caerPolarityIteratorCounter >= 0; \
        caerPolarityIteratorCounter--) { \
        caerPolarityEvent caerPolarityIteratorElement =
      caerPolarityEventPacketGetEvent(POLARITY_PACKET, caerPolarityIteratorCounter
      );
```

Reverse iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.16.2.7 CAER_POLARITY_REVERSE_ITERATOR_VALID_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.16.2.8 CAER_POLARITY_REVERSE_ITERATOR_VALID_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader) - 1; \
        caerPolarityIteratorCounter >= 0; \
        caerPolarityIteratorCounter--) { \
        caerPolarityEvent caerPolarityIteratorElement =
      caerPolarityEventPacketGetEvent(POLARITY_PACKET, caerPolarityIteratorCounter
      ); \
        if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) { continue; }
```

Reverse iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarity↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.16.2.9 POLARITY_MASK

```
#define POLARITY_MASK 0x00000001
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.2.10 POLARITY_SHIFT

```
#define POLARITY_SHIFT 1
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.2.11 X_ADDR_MASK

```
#define X_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.2.12 X_ADDR_SHIFT

```
#define X_ADDR_SHIFT 17
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.2.13 Y_ADDR_MASK

```
#define Y_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.2.14 Y_ADDR_SHIFT

```
#define Y_ADDR_SHIFT 2
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 14 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.16.3 Typedef Documentation

4.16.3.1 caerPolarityEvent

```
typedef struct caer_polarity_event* caerPolarityEvent
```

Type for pointer to polarity event data structure.

4.16.3.2 caerPolarityEventPacket

```
typedef struct caer_polarity_event_packet* caerPolarityEventPacket
```

Type for pointer to polarity event packet data structure.

4.16.4 Function Documentation

4.16.4.1 caerPolarityEventGetPolarity()

```
static bool caerPolarityEventGetPolarity (
            caerPolarityEvent event )  [inline], [static]
```

Get the change event polarity. 1 is ON, 0 is OFF.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

event polarity value.

4.16.4.2 caerPolarityEventGetTimestamp()

```
static int32_t caerPolarityEventGetTimestamp (
            caerPolarityEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.16.4.3 caerPolarityEventGetTimestamp64()

```
static int64_t caerPolarityEventGetTimestamp64 (
            caerPolarityEvent event,
            caerPolarityEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.16.4.4 caerPolarityEventGetX()

```
static uint16_t caerPolarityEventGetX (
            caerPolarityEvent event )  [inline], [static]
```

Get the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

the event X address.

4.16.4.5 caerPolarityEventGetY()

```
static uint16_t caerPolarityEventGetY (
            caerPolarityEvent event )  [inline], [static]
```

Get the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

the event Y address.

4.16.4.6 caerPolarityEventInvalidate()

```
static void caerPolarityEventInvalidate (
            caerPolarityEvent event,
            caerPolarityEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.16.4.7 caerPolarityEventIsValid()

```
static bool caerPolarityEventIsValid (
            caerPolarityEvent event )  [inline], [static]
```

Check if this polarity event is valid.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.16.4.8 caerPolarityEventPacketAllocate()

```
caerPolarityEventPacket caerPolarityEventPacketAllocate (
            int32_t eventCapacity,
```

```
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new polarity events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid PolarityEventPacket handle or NULL on error.

4.16.4.9 caerPolarityEventPacketGetEvent()

```
static caerPolarityEvent caerPolarityEventPacketGetEvent (
            caerPolarityEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the polarity event at the given index from the event packet.

Parameters

packet	a valid PolarityEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested polarity event. NULL on error.

4.16.4.10 caerPolarityEventSetPolarity()

```
static void caerPolarityEventSetPolarity (
            caerPolarityEvent event,
            bool polarity )  [inline], [static]
```

Set the change event polarity. 1 is ON, 0 is OFF.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
polarity	event polarity value.

4.16.4.11 caerPolarityEventSetTimestamp()

```
static void caerPolarityEventSetTimestamp (
            caerPolarityEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.16.4.12 caerPolarityEventSetX()

```
static void caerPolarityEventSetX (
            caerPolarityEvent event,
            uint16_t xAddress )  [inline], [static]
```

Set the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
xAddress	the event X address.

4.16.4.13 caerPolarityEventSetY()

```
static void caerPolarityEventSetY (
            caerPolarityEvent event,
            uint16_t yAddress )  [inline], [static]
```

Set the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
yAddress	the event Y address.

4.16.4.14 caerPolarityEventValidate()

```
static void caerPolarityEventValidate (
             caerPolarityEvent event,
             caerPolarityEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.16.4.15 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
             struct caer_polarity_event { uint32_t data;int32_t timestamp;}  )
```

Polarity event data structure definition. This contains the actual X/Y addresses, the polarity, as well as the 32 bit event timestamp. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.16.4.16 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
             struct caer_polarity_event_packet { struct caer_event_packet_header packet↩
Header;struct caer_polarity_event events[];}  )
```

Polarity event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.17 events/sample.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_SAMPLE_ITERATOR_ALL_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_ITERATOR_ALL_END }
- #define CAER_SAMPLE_ITERATOR_VALID_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_ITERATOR_VALID_END }

- #define SAMPLE_TYPE_SHIFT 1
- #define SAMPLE_TYPE_MASK 0x0000007F
- #define SAMPLE_SHIFT 8
- #define SAMPLE_MASK 0x00FFFFFF

Typedefs

- typedef struct caer_sample_event * caerSampleEvent
- typedef struct caer_sample_event_packet * caerSampleEventPacket

Functions

- PACKED_STRUCT (struct caer_sample_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_sample_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_sample_event events[ ];})
- caerSampleEventPacket caerSampleEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSampleEvent caerSampleEventPacketGetEvent (caerSampleEventPacket packet, int32_t n)
- static int32_t caerSampleEventGetTimestamp (caerSampleEvent event)
- static int64_t caerSampleEventGetTimestamp64 (caerSampleEvent event, caerSampleEventPacket packet)
- static void caerSampleEventSetTimestamp (caerSampleEvent event, int32_t timestamp)
- static bool caerSampleEventIsValid (caerSampleEvent event)
- static void caerSampleEventValidate (caerSampleEvent event, caerSampleEventPacket packet)
- static void caerSampleEventInvalidate (caerSampleEvent event, caerSampleEventPacket packet)
- static uint8_t caerSampleEventGetType (caerSampleEvent event)
- static void caerSampleEventSetType (caerSampleEvent event, uint8_t type)
- static uint32_t caerSampleEventGetSample (caerSampleEvent event)
- static void caerSampleEventSetSample (caerSampleEvent event, uint32_t sample)

4.17.1 Detailed Description

Sample (ADC) Events format definition and handling functions. Represents different types of ADC readings, up to 24 bits of resolution.

4.17.2 Macro Definition Documentation

4.17.2.1 CAER_SAMPLE_ITERATOR_ALL_END

```
#define CAER_SAMPLE_ITERATOR_ALL_END }
```

Iterator close statement.

4.17.2.2 CAER_SAMPLE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_ITERATOR_ALL_START(
              SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0; \
       caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SAMPLE_PACKET)->packetHeader); \
       caerSampleIteratorCounter++) { \
       caerSampleEvent caerSampleIteratorElement =
      caerSampleEventPacketGetEvent(SAMPLE_PACKET, caerSampleIteratorCounter);
```

Iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.17.2.3 CAER_SAMPLE_ITERATOR_VALID_END

```
#define CAER_SAMPLE_ITERATOR_VALID_END }
```

Iterator close statement.

4.17.2.4 CAER_SAMPLE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_ITERATOR_VALID_START(
              SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0; \
        caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SAMPLE_PACKET)->packetHeader); \
        caerSampleIteratorCounter++) { \
        caerSampleEvent caerSampleIteratorElement =
      caerSampleEventPacketGetEvent(SAMPLE_PACKET, caerSampleIteratorCounter); \
        if (!caerSampleEventIsValid(caerSampleIteratorElement)) { continue; }
```

Iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.17.2.5 SAMPLE_MASK

```
#define SAMPLE_MASK 0x00FFFFFF
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.6 SAMPLE_SHIFT

```
#define SAMPLE_SHIFT 8
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.7 SAMPLE_TYPE_MASK

```
#define SAMPLE_TYPE_MASK 0x0000007F
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.8 SAMPLE_TYPE_SHIFT

```
#define SAMPLE_TYPE_SHIFT 1
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.3 Typedef Documentation

4.17.3.1 caerSampleEvent

```
typedef struct caer_sample_event* caerSampleEvent
```

Type for pointer to ADC sample event data structure.

4.17.3.2 caerSampleEventPacket

```
typedef struct caer_sample_event_packet* caerSampleEventPacket
```

Type for pointer to ADC sample event packet data structure.

4.17.4 Function Documentation

4.17.4.1 caerSampleEventGetSample()

```
static uint32_t caerSampleEventGetSample (
            caerSampleEvent event )  [inline], [static]
```

Get the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

the ADC sample value.

4.17.4.2 caerSampleEventGetTimestamp()

```
static int32_t caerSampleEventGetTimestamp (
            caerSampleEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.17.4.3 caerSampleEventGetTimestamp64()

```
static int64_t caerSampleEventGetTimestamp64 (
            caerSampleEvent event,
            caerSampleEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.17.4.4 caerSampleEventGetType()

```
static uint8_t caerSampleEventGetType (
            caerSampleEvent event )  [inline], [static]
```

Get the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

the ADC sample type.

4.17.4.5 caerSampleEventInvalidate()

```
static void caerSampleEventInvalidate (
            caerSampleEvent event,
            caerSampleEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.6 caerSampleEventIsValid()

```
static bool caerSampleEventIsValid (
            caerSampleEvent event )  [inline], [static]
```

Check if this ADC sample event is valid.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.17.4.7 caerSampleEventPacketAllocate()

```
caerSampleEventPacket caerSampleEventPacketAllocate (
            int32_t eventCapacity,
```

```
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new ADC sample events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SampleEventPacket handle or NULL on error.

4.17.4.8 caerSampleEventPacketGetEvent()

```
static caerSampleEvent caerSampleEventPacketGetEvent (
            caerSampleEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the ADC sample event at the given index from the event packet.

Parameters

packet	a valid SampleEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ADC sample event. NULL on error.

4.17.4.9 caerSampleEventSetSample()

```
static void caerSampleEventSetSample (
            caerSampleEvent event,
            uint32_t sample )  [inline], [static]
```

Set the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
sample	the ADC sample value.

4.17.4.10 caerSampleEventSetTimestamp()

```
static void caerSampleEventSetTimestamp (
            caerSampleEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.17.4.11 caerSampleEventSetType()

```
static void caerSampleEventSetType (
            caerSampleEvent event,
            uint8_t type )  [inline], [static]
```

Set the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
type	the ADC sample type.

4.17.4.12 caerSampleEventValidate()

```
static void caerSampleEventValidate (
            caerSampleEvent event,
            caerSampleEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.13 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_sample_event { uint32_t data;int32_t timestamp;}  )
```

ADC sample event data structure definition. Contains a type indication to separate different ADC readouts, as well as a value for that readout, up to 24 bits resolution. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.17.4.14 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_sample_event_packet { struct caer_event_packet_header packetHeader;struct
caer_sample_event events[];}  )
```

ADC sample event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.18 events/special.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_SPECIAL_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_ITERATOR_ALL_END }
- #define CAER_SPECIAL_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_ITERATOR_VALID_END }

- #define TYPE_SHIFT 1
- #define TYPE_MASK 0x0000007F
- #define DATA_SHIFT 8
- #define DATA_MASK 0x00FFFFFF

Typedefs

- typedef struct caer_special_event * caerSpecialEvent
- typedef const struct caer_special_event * **caerSpecialEventConst**
- typedef struct caer_special_event_packet * caerSpecialEventPacket
- typedef const struct caer_special_event_packet * **caerSpecialEventPacketConst**

Enumerations

- enum caer_special_event_types {
  TIMESTAMP_WRAP = 0, TIMESTAMP_RESET = 1, EXTERNAL_INPUT_RISING_EDGE = 2, EXTERNAL_INPUT_FALLING_EDGE = 3,
  EXTERNAL_INPUT_PULSE = 4, DVS_ROW_ONLY = 5, EXTERNAL_INPUT1_RISING_EDGE = 6, EXTERNAL_INPUT1_FALLING_EDGE = 7,
  EXTERNAL_INPUT1_PULSE = 8, EXTERNAL_INPUT2_RISING_EDGE = 9, EXTERNAL_INPUT2_FALLING_EDGE = 10, EXTERNAL_INPUT2_PULSE = 11,
  EXTERNAL_GENERATOR_RISING_EDGE = 12, EXTERNAL_GENERATOR_FALLING_EDGE = 13, APS_FRAME_START = 14, APS_FRAME_END = 15,
  APS_EXPOSURE_START = 16, APS_EXPOSURE_END = 17 }

Functions

- PACKED_STRUCT (struct caer_special_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_special_event_packet { struct caer_event_packet_header packetHeader;struct caer_special_event events[ ];})
- caerSpecialEventPacket caerSpecialEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSpecialEvent caerSpecialEventPacketGetEvent (caerSpecialEventPacket packet, int32_t n)
- static caerSpecialEventConst caerSpecialEventPacketGetEventConst (caerSpecialEventPacketConst packet, int32_t n)
- static int32_t caerSpecialEventGetTimestamp (caerSpecialEventConst event)
- static int64_t caerSpecialEventGetTimestamp64 (caerSpecialEventConst event, caerSpecialEventPacketConst packet)
- static void caerSpecialEventSetTimestamp (caerSpecialEvent event, int32_t timestamp)
- static bool caerSpecialEventIsValid (caerSpecialEventConst event)
- static void caerSpecialEventValidate (caerSpecialEvent event, caerSpecialEventPacket packet)
- static void caerSpecialEventInvalidate (caerSpecialEvent event, caerSpecialEventPacket packet)
- static uint8_t caerSpecialEventGetType (caerSpecialEventConst event)
- static void caerSpecialEventSetType (caerSpecialEvent event, uint8_t type)
- static uint32_t caerSpecialEventGetData (caerSpecialEventConst event)
- static void caerSpecialEventSetData (caerSpecialEvent event, uint32_t data)
- static caerSpecialEvent caerSpecialEventPacketFindEventByType (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst caerSpecialEventPacketFindEventByTypeConst (caerSpecialEventPacketConst packet, uint8_t type)
- static caerSpecialEvent caerSpecialEventPacketFindValidEventByType (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst caerSpecialEventPacketFindValidEventByTypeConst (caerSpecialEventPacketConst packet, uint8_t type)

4.18.1 Detailed Description

Special Events format definition and handling functions. This event type encodes special occurrences, such as timestamp related notifications or external input events.

4.18.2 Macro Definition Documentation

4.18.2.1 CAER_SPECIAL_CONST_ITERATOR_ALL_START

```
#define CAER_SPECIAL_CONST_ITERATOR_ALL_START(
              SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0; \
        caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SPECIAL_PACKET)->packetHeader); \
        caerSpecialIteratorCounter++) { \
        caerSpecialEventConst caerSpecialIteratorElement =
      caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter);
```

Const-Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caer↩SpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.18.2.2 CAER_SPECIAL_CONST_ITERATOR_VALID_START

```
#define CAER_SPECIAL_CONST_ITERATOR_VALID_START(
              SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0; \
        caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SPECIAL_PACKET)->packetHeader); \
        caerSpecialIteratorCounter++) { \
        caerSpecialEventConst caerSpecialIteratorElement =
      caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter); \
        if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) { continue; }
```

Const-Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIterator↩Counter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.18.2.3 CAER_SPECIAL_ITERATOR_ALL_END

```
#define CAER_SPECIAL_ITERATOR_ALL_END }
```

Iterator close statement.

4.18.2.4 CAER_SPECIAL_ITERATOR_ALL_START

```
#define CAER_SPECIAL_ITERATOR_ALL_START(
               SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0; \
       caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SPECIAL_PACKET)->packetHeader); \
       caerSpecialIteratorCounter++) { \
       caerSpecialEvent caerSpecialIteratorElement =
     caerSpecialEventPacketGetEvent(SPECIAL_PACKET, caerSpecialIteratorCounter);
```

Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.18.2.5 CAER_SPECIAL_ITERATOR_VALID_END

```
#define CAER_SPECIAL_ITERATOR_VALID_END }
```

Iterator close statement.

4.18.2.6 CAER_SPECIAL_ITERATOR_VALID_START

```
#define CAER_SPECIAL_ITERATOR_VALID_START(
               SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0; \
       caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SPECIAL_PACKET)->packetHeader); \
       caerSpecialIteratorCounter++) { \
       caerSpecialEvent caerSpecialIteratorElement =
     caerSpecialEventPacketGetEvent(SPECIAL_PACKET, caerSpecialIteratorCounter); \
       if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) { continue; }
```

Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.18.2.7 DATA_MASK

```
#define DATA_MASK 0x00FFFFFF
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.8 DATA_SHIFT

```
#define DATA_SHIFT 8
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.9 TYPE_MASK

```
#define TYPE_MASK 0x0000007F
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.10 TYPE_SHIFT

```
#define TYPE_SHIFT 1
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.3 Typedef Documentation

4.18.3.1 caerSpecialEvent

```
typedef struct caer_special_event* caerSpecialEvent
```

Type for pointer to special event data structure.

4.18.3.2 caerSpecialEventPacket

```
typedef struct caer_special_event_packet* caerSpecialEventPacket
```

Type for pointer to special event packet data structure.

4.18.4 Enumeration Type Documentation

4.18.4.1 caer_special_event_types

```
enum caer_special_event_types
```

List of all special event type identifiers. Used to interpret the special event type field.

Enumerator

TIMESTAMP_WRAP	A 32 bit timestamp wrap occurred.
TIMESTAMP_RESET	A timestamp reset occurred.
EXTERNAL_INPUT_RISING_EDGE	A rising edge was detected (External Input module on device).
EXTERNAL_INPUT_FALLING_EDGE	A falling edge was detected (External Input module on device).
EXTERNAL_INPUT_PULSE	A pulse was detected (External Input module on device).
DVS_ROW_ONLY	A DVS row-only event was detected (a row address without any following column addresses).
EXTERNAL_INPUT1_RISING_EDGE	A rising edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_FALLING_EDGE	A falling edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_PULSE	A pulse was detected (External Input 1 module on device).
EXTERNAL_INPUT2_RISING_EDGE	A rising edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_FALLING_EDGE	A falling edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_PULSE	A pulse was detected (External Input 2 module on device).
EXTERNAL_GENERATOR_RISING_EDGE	A rising edge was generated (External Input Generator module on device).
EXTERNAL_GENERATOR_FALLING_EDGE	A falling edge was generated (External Input Generator module on device).
APS_FRAME_START	An APS frame capture has started (Frame Event will follow).
APS_FRAME_END	An APS frame capture has completed (Frame Event is alongside).
APS_EXPOSURE_START	An APS frame exposure has started (Frame Event will follow).
APS_EXPOSURE_END	An APS frame exposure has completed (Frame Event will follow).

4.18.5 Function Documentation

4.18.5.1 caerSpecialEventGetData()

```
static uint32_t caerSpecialEventGetData (
            caerSpecialEventConst event )  [inline], [static]
```

Get the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

the special event data.

4.18.5.2 caerSpecialEventGetTimestamp()

```
static int32_t caerSpecialEventGetTimestamp (
            caerSpecialEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.18.5.3 caerSpecialEventGetTimestamp64()

```
static int64_t caerSpecialEventGetTimestamp64 (
            caerSpecialEventConst event,
            caerSpecialEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.18.5.4 caerSpecialEventGetType()

```
static uint8_t caerSpecialEventGetType (
            caerSpecialEventConst event )  [inline], [static]
```

Get the numerical special event type.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

the special event type (see 'enum caer_special_event_types').

4.18.5.5 caerSpecialEventInvalidate()

```
static void caerSpecialEventInvalidate (
            caerSpecialEvent event,
            caerSpecialEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.5.6 caerSpecialEventIsValid()

```
static bool caerSpecialEventIsValid (
            caerSpecialEventConst event )  [inline], [static]
```

Check if this special event is valid.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.18.5.7 caerSpecialEventPacketAllocate()

```
caerSpecialEventPacket caerSpecialEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new special events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SpecialEventPacket handle or NULL on error.

4.18.5.8 caerSpecialEventPacketFindEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindEventByType (
            caerSpecialEventPacket packet,
            uint8_t type )  [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested special event or NULL on error/not found.

4.18.5.9 caerSpecialEventPacketFindEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindEventByTypeConst (
            caerSpecialEventPacketConst packet,
            uint8_t type )  [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested read-only special event or NULL on error/not found.

4.18.5.10 caerSpecialEventPacketFindValidEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindValidEventByType (
            caerSpecialEventPacket packet,
            uint8_t type )  [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested valid special event or NULL on error/not found.

4.18.5.11 caerSpecialEventPacketFindValidEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindValidEventByTypeConst (
            caerSpecialEventPacketConst packet,
            uint8_t type )  [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested read-only valid special event or NULL on error/not found.

4.18.5.12 caerSpecialEventPacketGetEvent()

```
static caerSpecialEvent caerSpecialEventPacketGetEvent (
            caerSpecialEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the special event at the given index from the event packet.

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested special event. NULL on error.

4.18.5.13 caerSpecialEventPacketGetEventConst()

```
static caerSpecialEventConst caerSpecialEventPacketGetEventConst (
            caerSpecialEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the special event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only special event. NULL on error.

4.18.5.14 caerSpecialEventSetData()

```
static void caerSpecialEventSetData (
            caerSpecialEvent event,
            uint32_t data )  [inline], [static]
```

Set the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
data	the special event data.

4.18.5.15 caerSpecialEventSetTimestamp()

```
static void caerSpecialEventSetTimestamp (
            caerSpecialEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.18.5.16 caerSpecialEventSetType()

```
static void caerSpecialEventSetType (
            caerSpecialEvent event,
            uint8_t type )  [inline], [static]
```

Set the numerical special event type.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
type	the special event type (see 'enum caer_special_event_types').

4.18.5.17 caerSpecialEventValidate()

```
static void caerSpecialEventValidate (
            caerSpecialEvent event,
            caerSpecialEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.5.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_special_event { uint32_t data;int32_t timestamp;}  )
```

Special event data structure definition. This contains the actual data, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.18.5.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_special_event_packet { struct caer_event_packet_header packetHeader;struct
caer_special_event events[];}  )
```

Special event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.19 events/spike.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_SPIKE_ITERATOR_ALL_START(SPIKE_PACKET)
- #define CAER_SPIKE_ITERATOR_ALL_END }
- #define CAER_SPIKE_ITERATOR_VALID_START(SPIKE_PACKET)
- #define CAER_SPIKE_ITERATOR_VALID_END }

- #define SPIKE_SOURCE_CORE_ID_SHIFT 1
- #define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F
- #define SPIKE_CHIP_ID_SHIFT 6
- #define SPIKE_CHIP_ID_MASK 0x0000003F
- #define SPIKE_NEURON_ID_SHIFT 12
- #define SPIKE_NEURON_ID_MASK 0x000FFFFF

Typedefs

- typedef struct caer_spike_event * caerSpikeEvent
- typedef struct caer_spike_event_packet * caerSpikeEventPacket

Functions

- PACKED_STRUCT (struct caer_spike_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct caer_spike_event events[ ];})
- caerSpikeEventPacket caerSpikeEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSpikeEvent caerSpikeEventPacketGetEvent (caerSpikeEventPacket packet, int32_t n)
- static int32_t caerSpikeEventGetTimestamp (caerSpikeEvent event)
- static int64_t caerSpikeEventGetTimestamp64 (caerSpikeEvent event, caerSpikeEventPacket packet)
- static void caerSpikeEventSetTimestamp (caerSpikeEvent event, int32_t timestamp)
- static bool caerSpikeEventIsValid (caerSpikeEvent event)
- static void caerSpikeEventValidate (caerSpikeEvent event, caerSpikeEventPacket packet)
- static void caerSpikeEventInvalidate (caerSpikeEvent event, caerSpikeEventPacket packet)
- static uint8_t caerSpikeEventGetSourceCoreID (caerSpikeEvent event)
- static void caerSpikeEventSetSourceCoreID (caerSpikeEvent event, uint8_t sourceCoreID)
- static uint8_t caerSpikeEventGetChipID (caerSpikeEvent event)
- static void caerSpikeEventSetChipID (caerSpikeEvent event, uint8_t chipID)
- static uint32_t caerSpikeEventGetNeuronID (caerSpikeEvent event)
- static void caerSpikeEventSetNeuronID (caerSpikeEvent event, uint32_t neuronID)
- static uint16_t caerSpikeEventGetY (caerSpikeEvent event)
- static uint16_t caerSpikeEventGetX (caerSpikeEvent event)

4.19.1 Detailed Description

THIS EVENT DEFINITIONS IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE CHANGES AND REVISIONS!

Spike Events format definition and handling functions. This contains spikes generated by a neuron-array chip.

4.19.2 Macro Definition Documentation

4.19.2.1 CAER_SPIKE_ITERATOR_ALL_END

```
#define CAER_SPIKE_ITERATOR_ALL_END }
```

Iterator close statement.

4.19.2.2 CAER_SPIKE_ITERATOR_ALL_START

```
#define CAER_SPIKE_ITERATOR_ALL_START(
               SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0; \
       caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(&
     (SPIKE_PACKET)->packetHeader); \
       caerSpikeIteratorCounter++) { \
       caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(
     SPIKE_PACKET, caerSpikeIteratorCounter);
```

Iterator over all Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.19.2.3 CAER_SPIKE_ITERATOR_VALID_END

```
#define CAER_SPIKE_ITERATOR_VALID_END }
```

Iterator close statement.

4.19.2.4 CAER_SPIKE_ITERATOR_VALID_START

```
#define CAER_SPIKE_ITERATOR_VALID_START(
               SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0; \
       caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(&
     (SPIKE_PACKET)->packetHeader); \
       caerSpikeIteratorCounter++) { \
       caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(
     SPIKE_PACKET, caerSpikeIteratorCounter); \
       if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) { continue; }
```

Iterator over only the valid Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.19.2.5 SPIKE_CHIP_ID_MASK

```
#define SPIKE_CHIP_ID_MASK 0x0000003F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.6 SPIKE_CHIP_ID_SHIFT

```
#define SPIKE_CHIP_ID_SHIFT 6
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.7 SPIKE_NEURON_ID_MASK

```
#define SPIKE_NEURON_ID_MASK 0x000FFFFF
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.8 SPIKE_NEURON_ID_SHIFT

```
#define SPIKE_NEURON_ID_SHIFT 12
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.9 SPIKE_SOURCE_CORE_ID_MASK

```
#define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.10 SPIKE_SOURCE_CORE_ID_SHIFT

```
#define SPIKE_SOURCE_CORE_ID_SHIFT 1
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.3 Typedef Documentation

4.19.3.1 caerSpikeEvent

```
typedef struct caer_spike_event* caerSpikeEvent
```

Type for pointer to Spike event data structure.

4.19.3.2 caerSpikeEventPacket

```
typedef struct caer_spike_event_packet* caerSpikeEventPacket
```

Type for pointer to Spike event packet data structure.

4.19.4 Function Documentation

4.19.4.1 caerSpikeEventGetChipID()

```
static uint8_t caerSpikeEventGetChipID (
            caerSpikeEvent event ) [inline], [static]
```

Get the chip ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's chip ID.

4.19.4.2 caerSpikeEventGetNeuronID()

```
static uint32_t caerSpikeEventGetNeuronID (
            caerSpikeEvent event ) [inline], [static]
```

Get the neuron ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's neuron ID.

4.19.4.3 caerSpikeEventGetSourceCoreID()

```
static uint8_t caerSpikeEventGetSourceCoreID (
            caerSpikeEvent event ) [inline], [static]
```

Get the source core ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's source core ID.

4.19.4.4 caerSpikeEventGetTimestamp()

```
static int32_t caerSpikeEventGetTimestamp (
            caerSpikeEvent event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.19.4.5 caerSpikeEventGetTimestamp64()

```
static int64_t caerSpikeEventGetTimestamp64 (
            caerSpikeEvent event,
            caerSpikeEventPacket packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.19.4.6 caerSpikeEventGetX()

```
static uint16_t caerSpikeEventGetX (
            caerSpikeEvent event )  [inline], [static]
```

Get the X (column) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the event X address in pixels.

4.19.4.7 caerSpikeEventGetY()

```
static uint16_t caerSpikeEventGetY (
            caerSpikeEvent event )  [inline], [static]
```

Get the Y (row) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the event Y address in pixels.

4.19.4.8 caerSpikeEventInvalidate()

```
static void caerSpikeEventInvalidate (
            caerSpikeEvent event,
            caerSpikeEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.9 caerSpikeEventIsValid()

```
static bool caerSpikeEventIsValid (
            caerSpikeEvent event )  [inline], [static]
```

Check if this Spike event is valid.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.19.4.10 caerSpikeEventPacketAllocate()

```
caerSpikeEventPacket caerSpikeEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )
```

Allocate a new Spike events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SpikeEventPacket handle or NULL on error.

4.19.4.11 caerSpikeEventPacketGetEvent()

```
static caerSpikeEvent caerSpikeEventPacketGetEvent (
            caerSpikeEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Spike event at the given index from the event packet.

Parameters

packet	a valid SpikeEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Spike event. NULL on error.

4.19.4.12 caerSpikeEventSetChipID()

```
static void caerSpikeEventSetChipID (
            caerSpikeEvent event,
            uint8_t chipID )  [inline], [static]
```

Set the chip ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
chipID	the Spike's chip ID.

4.19.4.13 caerSpikeEventSetNeuronID()

```
static void caerSpikeEventSetNeuronID (
            caerSpikeEvent event,
            uint32_t neuronID )  [inline], [static]
```

Set the neuron ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
neuronID	the Spike's neuron ID.

4.19.4.14 caerSpikeEventSetSourceCoreID()

```
static void caerSpikeEventSetSourceCoreID (
              caerSpikeEvent event,
              uint8_t sourceCoreID )  [inline], [static]
```

Set the source core ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
sourceCoreID	the Spike's source core ID.

4.19.4.15 caerSpikeEventSetTimestamp()

```
static void caerSpikeEventSetTimestamp (
              caerSpikeEvent event,
              int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.19.4.16 caerSpikeEventValidate()

```
static void caerSpikeEventValidate (
              caerSpikeEvent event,
              caerSpikeEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.17 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_spike_event { uint32_t data;int32_t timestamp;}  )
```

Spike event data structure definition. This contains the core ID, the neuron ID and the timestamp of the received spike, together with the usual validity mark. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.19.4.18 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct
caer_spike_event events[];}  )
```

Spike event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.20 frame_utils.h File Reference

```
#include "events/frame.h"
```

Functions

- caerFrameEventPacket **caerFrameUtilsDemosaic** (caerFrameEventPacket framePacket)
- void **caerFrameUtilsContrast** (caerFrameEventPacket framePacket)

4.20.1 Detailed Description

Basic functions for frame enhancement and demosaicing, that don't require any external dependencies, such as OpenCV. Use of the OpenCV variants is recommended for quality and performance.

4.21 frame_utils_opencv.h File Reference

```
#include "events/frame.h"
```

Enumerations

- enum **caer_frame_utils_opencv_demosaic** { **DEMOSAIC_NORMAL**, **DEMOSAIC_EDGE_AWARE** }
- enum **caer_frame_utils_opencv_contrast** { **CONTRAST_NORMALIZATION**, **CONTRAST_HISTOGRA↩ M_EQUALIZATION**, **CONTRAST_CLAHE** }

Functions

- caerFrameEventPacket **caerFrameUtilsOpenCVDemosaic** (caerFrameEventPacket framePacket, enum caer_frame_utils_opencv_demosaic demosaicType)
- void **caerFrameUtilsOpenCVContrast** (caerFrameEventPacket framePacket, enum caer_frame_utils_↩ opencv_contrast contrastType)

4.21.1 Detailed Description

Functions for frame enhancement and demosaicing, using the popular OpenCV image processing library.

4.22 libcaer.h File Reference

```
#include <stddef.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include <errno.h>
#include "portable_endian.h"
#include "log.h"
```

Macros

- #define LIBCAER_VERSION ((2 * 10000) + (0 * 100) + 0)
- #define LIBCAER_NAME_STRING "libcaer"
- #define LIBCAER_VERSION_STRING "2.0.0"
- #define U8T(X) ((uint8_t) (X))
- #define U16T(X) ((uint16_t) (X))
- #define U32T(X) ((uint32_t) (X))
- #define U64T(X) ((uint64_t) (X))
- #define I8T(X) ((int8_t) (X))
- #define I16T(X) ((int16_t) (X))
- #define I32T(X) ((int32_t) (X))
- #define I64T(X) ((int64_t) (X))
- #define MASK_NUMBITS32(X) U32T(U32T(U32T(1) << X) - 1)
- #define MASK_NUMBITS64(X) U64T(U64T(U64T(1) << X) - 1)
- #define SWAP_VAR(type, x, y) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }

- #define CLEAR_NUMBITS32(VAR, SHIFT, MASK) (VAR) &= htole32(~(U32T(U32T(MASK) << (SHIFT))))
- #define CLEAR_NUMBITS16(VAR, SHIFT, MASK) (VAR) &= htole16(~(U16T(U16T(MASK) << (SHIFT))))
- #define CLEAR_NUMBITS8(VAR, SHIFT, MASK) (VAR) &= U8T(~(U8T(U8T(MASK) << (SHIFT))))

- #define SET_NUMBITS32(VAR, SHIFT, MASK, VALUE) (VAR) |= htole32(U32T((U32T(VALUE) & (MASK)) << (SHIFT)))
- #define SET_NUMBITS16(VAR, SHIFT, MASK, VALUE) (VAR) |= htole16(U16T((U16T(VALUE) & (MASK)) << (SHIFT)))
- #define SET_NUMBITS8(VAR, SHIFT, MASK, VALUE) (VAR) |= U8T((U8T(VALUE) & (MASK)) << (SHIFT))

- #define GET_NUMBITS32(VAR, SHIFT, MASK) ((le32toh(VAR) >> (SHIFT)) & (MASK))
- #define GET_NUMBITS16(VAR, SHIFT, MASK) ((le16toh(VAR) >> (SHIFT)) & (MASK))
- #define GET_NUMBITS8(VAR, SHIFT, MASK) ((U8T(VAR) >> (SHIFT)) & (MASK))

Functions

- static bool caerStrEquals (const char *s1, const char *s2)
- static bool caerStrEqualsUpTo (const char *s1, const char *s2, size_t len)
- static void caerIntegerToByteArray (const uint32_t integer, uint8_t *byteArray, const uint8_t byteArrayLength)
- static uint32_t caerByteArrayToInteger (const uint8_t *byteArray, const uint8_t byteArrayLength)

4.22.1 Detailed Description

Main libcaer header; provides inclusions for common system functions and definitions for useful macros used often in the code. Also includes the logging functions and definitions and several useful static inline functions for string comparison and byte array manipulation. When including libcaer, please make sure to always use the full path, ie. #include <libcaer/libcaer.h> and not just #include <libcaer.h>.

4.22.2 Macro Definition Documentation

4.22.2.1 CLEAR_NUMBITS16

```
#define CLEAR_NUMBITS16(
              VAR,
              SHIFT,
              MASK ) (VAR) &= htole16(~(U16T(U16T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.22.2.2 CLEAR_NUMBITS32

```
#define CLEAR_NUMBITS32(
              VAR,
              SHIFT,
              MASK ) (VAR) &= htole32(~(U32T(U32T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.22.2.3 CLEAR_NUMBITS8

```
#define CLEAR_NUMBITS8(
              VAR,
              SHIFT,
              MASK ) (VAR) &= U8T(~(U8T(U8T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.22.2.4 GET_NUMBITS16

```
#define GET_NUMBITS16(
              VAR,
              SHIFT,
              MASK ) ((le16toh(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.22.2.5 GET_NUMBITS32

```
#define GET_NUMBITS32(
              VAR,
              SHIFT,
              MASK ) ((le32toh(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.22.2.6 GET_NUMBITS8

```
#define GET_NUMBITS8(
              VAR,
              SHIFT,
              MASK ) ((U8T(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.22.2.7 I16T

```
#define I16T(
              X ) ((int16_t) (X))
```

Cast argument to int16_t (16bit signed integer).

4.22.2.8 I32T

```
#define I32T(
              X ) ((int32_t) (X))
```

Cast argument to int32_t (32bit signed integer).

4.22.2.9 I64T

```
#define I64T(
              X ) ((int64_t) (X))
```

Cast argument to int64_t (64bit signed integer).

4.22.2.10 I8T

```
#define I8T(
              X ) ((int8_t) (X))
```

Cast argument to int8_t (8bit signed integer).

4.22.2.11 LIBCAER_NAME_STRING

```
#define LIBCAER_NAME_STRING "libcaer"
```

libcaer name string.

4.22.2.12 LIBCAER_VERSION

```
#define LIBCAER_VERSION ((2 * 10000) + (0 * 100) + 0)
```

libcaer version (MAJOR $*$ 10000 + MINOR $*$ 100 + PATCH).

4.22.2.13 LIBCAER_VERSION_STRING

```
#define LIBCAER_VERSION_STRING "2.0.0"
```

libcaer version string.

4.22.2.14 MASK_NUMBITS32

```
#define MASK_NUMBITS32(
              X ) U32T(U32T(U32T(1) << X) - 1)
```

Mask and keep only the lower X bits of a 32bit (unsigned) integer.

4.22.2.15 MASK_NUMBITS64

```
#define MASK_NUMBITS64(
              X ) U64T(U64T(U64T(1) << X) - 1)
```

Mask and keep only the lower X bits of a 64bit (unsigned) integer.

4.22.2.16 SET_NUMBITS16

```
#define SET_NUMBITS16(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= htole16(U16T((U16T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.22.2.17 SET_NUMBITS32

```
#define SET_NUMBITS32(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= htole32(U32T((U32T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.22.2.18 SET_NUMBITS8

```
#define SET_NUMBITS8(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= U8T((U8T(VALUE) & (MASK)) << (SHIFT))
```

Set bits given by mask (amount) and shift (position) to a value.

4.22.2.19 SWAP_VAR

```
#define SWAP_VAR(
              type,
              x,
              y ) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }
```

Swap the two values of the two variables X and Y, of a common type TYPE.

4.22.2.20 U16T

```
#define U16T(
              X ) ((uint16_t) (X))
```

Cast argument to uint16_t (16bit unsigned integer).

4.22.2.21 U32T

```
#define U32T(
              X ) ((uint32_t) (X))
```

Cast argument to uint32_t (32bit unsigned integer).

4.22.2.22 U64T

```
#define U64T(
              X ) ((uint64_t) (X))
```

Cast argument to uint64_t (64bit unsigned integer).

4.22.2.23 U8T

```
#define U8T(
              X ) ((uint8_t) (X))
```

Cast argument to uint8_t (8bit unsigned integer).

4.22.3 Function Documentation

4.22.3.1 caerByteArrayToInteger()

```
static uint32_t caerByteArrayToInteger (
             const uint8_t * byteArray,
             const uint8_t byteArrayLength )  [inline], [static]
```

Convert an unsigned byte array of up to four bytes into a 32bit unsigned integer. The byte array length decides how many resulting bits in the integer are set, and the single bytes are placed in the integer following big-endian ordering.

Parameters

byteArray	pointer to the byte array with parts of the value stored.
byteArrayLength	length of the array from which to convert.

Returns

integer representing the value stored in the byte array.

4.22.3.2 caerIntegerToByteArray()

```
static void caerIntegerToByteArray (
             const uint32_t integer,
             uint8_t * byteArray,
             const uint8_t byteArrayLength )  [inline], [static]
```

Convert a 32bit unsigned integer into an unsigned byte array of up to four bytes. The integer will be stored in big-endian order, and the length will specify how many bits to convert, starting from the lowest bit.

Parameters

integer	the integer to convert.
byteArray	pointer to the byte array in which to store the converted values.
byteArrayLength	length of the byte array to convert to.

4.22.3.3 caerStrEquals()

```
static bool caerStrEquals (
            const char * s1,
            const char * s2 )  [inline], [static]
```

Compare two strings for equality.

Parameters

s1	the first string, cannot be NULL.
s2	the second string, cannot be NULL.

Returns

true if equal, false otherwise.

4.22.3.4 caerStrEqualsUpTo()

```
static bool caerStrEqualsUpTo (
            const char * s1,
            const char * s2,
            size_t len )  [inline], [static]
```

Compare two strings for equality, up to a specified maximum length.

Parameters

s1	the first string, cannot be NULL.
s2	the second string, cannot be NULL.
len	maximum comparison length, cannot be zero.

Returns

true if equal, false otherwise.

4.23 log.h File Reference

```
#include <stdint.h>
#include <stdarg.h>
```

Macros

- #define **ATTRIBUTE_FORMAT**

- #define CAER_LOG_EMERGENCY (0)
- #define CAER_LOG_ALERT (1)
- #define CAER_LOG_CRITICAL (2)
- #define CAER_LOG_ERROR (3)
- #define CAER_LOG_WARNING (4)
- #define CAER_LOG_NOTICE (5)
- #define CAER_LOG_INFO (6)
- #define CAER_LOG_DEBUG (7)

Functions

- void caerLogLevelSet (uint8_t logLevel)
- uint8_t caerLogLevelGet (void)
- void caerLogFileDescriptorsSet (int fd1, int fd2)
- void caerLog (uint8_t logLevel, const char *subSystem, const char *format,...) ATTRIBUTE_FORMAT
- void caerLogVA (uint8_t logLevel, const char *subSystem, const char *format, va_list args) ATTRIBUTE_↩ FORMAT_VA

4.23.1 Detailed Description

Logging functions to print useful messages for the user.

4.23.2 Macro Definition Documentation

4.23.2.1 CAER_LOG_ALERT

```
#define CAER_LOG_ALERT (1)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.2 CAER_LOG_CRITICAL

```
#define CAER_LOG_CRITICAL (2)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.3 CAER_LOG_DEBUG

```
#define CAER_LOG_DEBUG (7)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.4 CAER_LOG_EMERGENCY

```
#define CAER_LOG_EMERGENCY (0)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.5 CAER_LOG_ERROR

```
#define CAER_LOG_ERROR (3)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.6 CAER_LOG_INFO

```
#define CAER_LOG_INFO (6)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.7 CAER_LOG_NOTICE

```
#define CAER_LOG_NOTICE (5)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.2.8 CAER_LOG_WARNING

```
#define CAER_LOG_WARNING (4)
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.23.3 Function Documentation

4.23.3.1 caerLog()

```
void caerLog (
            uint8_t logLevel,
            const char * subSystem,
            const char * format,
             ...  )
```

Main logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. Please see their manual-page for more information.

Parameters

logLevel	the message-specific log level.
subSystem	a common, user-specified string to prepend before the message.
format	the message format string (see printf()).
...	the parameters to be formatted according to the format string (see printf()).

4.23.3.2 caerLogFileDescriptorsSet()

```
void caerLogFileDescriptorsSet (
            int fd1,
            int fd2 )
```

Set to which file descriptors log messages are sent. Up to two different file descriptors can be configured here. By default logging to STDERR only is enabled. If both file descriptors are identical, logging to it will only happen once, as if the second one was disabled.

Parameters

fd1	first file descriptor to log to. A negative value will disable it.
fd2	second file descriptor to log to. A negative value will disable it.

4.23.3.3 caerLogLevelGet()

```
uint8_t caerLogLevelGet (
            void  )
```

Get the current system-wide log level. Log messages are only printed if their level is equal or above this level.

Returns

the current system-wide log level.

4.23.3.4 caerLogLevelSet()

```
void caerLogLevelSet (
            uint8_t logLevel )
```

Set the system-wide log level. Log messages will only be printed if their level is equal or above this level.

Parameters

logLevel	the system-wide log level.

4.23.3.5 caerLogVA()

```
void caerLogVA (
            uint8_t logLevel,
            const char * subSystem,
            const char * format,
            va_list args )
```

Secondary logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. The argument list is a va_list as returned by va_start(), following the vprintf() family of functions in its functionality. Please see their manual-page for more information.

Parameters

logLevel	the message-specific log level.
subSystem	a common, user-specified string to prepend before the message.
format	the message format string (see printf()).
args	the parameters to be formatted according to the format string (see printf()). This is an argument list as returned by va_start().

4.24 portable_endian.h File Reference

4.24.1 Detailed Description

Endianness conversion functions for a wide variety of systems, including Linux, FreeBSD, MacOS X and Windows.

Index