

## BEST Interface

Generated on Tue Jun 23 2026 12:20:27 for BEST Interface by Doxygen 1.17.0

Tue Jun 23 2026 12:20:27



<b>1 [2.0.0] - 2026/06/15</b>	<b>1</b>
1.0.1 ADDED	1
<b>2 Topic Index</b>	<b>3</b>
2.1 Topics	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Topic Documentation</b>	<b>9</b>
5.1 Data functions	9
5.1.1 Detailed Description	10
5.1.2 Function Documentation	10
5.1.2.1 __attribute__() [1/2]	10
5.1.2.2 __attribute__() [2/2]	10
5.1.2.3 getBPMStats()	10
5.1.2.4 getBuffer()	10
5.1.2.5 getCurrentArray()	10
5.1.2.6 getCurrentArray_sel()	11
5.1.2.7 getDisplayFreq()	11
5.1.2.8 getFFT()	11
5.1.2.9 getPosArray()	12
5.1.2.10 getPosArray_sel()	12
5.1.2.11 getPosScalar_sel()	12
5.1.2.12 getPosStats()	13
5.1.2.13 getVoltageArray()	13
5.1.2.14 getVoltageArray_sel()	14
5.1.2.15 getVoltageScalar_sel()	14
5.1.2.16 setDisplayFreq()	14
5.2 Configuration functions	15
5.2.1 Detailed Description	16
5.2.2 Enumeration Type Documentation	17
5.2.2.1 best_pid_select	17
5.2.3 Function Documentation	17
5.2.3.1 getBPMoffset()	17
5.2.3.2 getBPMoffsetEnbox()	17
5.2.3.3 getBPMorient()	18
5.2.3.4 getBPMscale()	18
5.2.3.5 getBPMscaling()	19
5.2.3.6 getBPMscaling_sel()	19
5.2.3.7 getBPMselector()	19

---

5.2.3.8	getCrossBar()	20
5.2.3.9	getOffsetSingle()	20
5.2.3.10	getOutputMux()	21
5.2.3.11	getPIDoutSel()	21
5.2.3.12	getPIDparamSingle()	22
5.2.3.13	getROcandROI()	22
5.2.3.14	getROcenable()	23
5.2.3.15	getROclimits()	24
5.2.3.16	getWindAvg()	24
5.2.3.17	setBPMdevice()	24
5.2.3.18	setBPMoffset()	25
5.2.3.19	setBPMoffsetEnbox()	25
5.2.3.20	setBPMorient()	26
5.2.3.21	setBPMscale()	26
5.2.3.22	setBPMselector()	27
5.2.3.23	setCrossBar()	27
5.2.3.24	setEnboxSumDiff()	28
5.2.3.25	setEnboxType()	28
5.2.3.26	setOffset()	29
5.2.3.27	setOffsetSingle()	29
5.2.3.28	setOutputMux()	30
5.2.3.29	setPIDoutSel()	30
5.2.3.30	setPIDparams()	31
5.2.3.31	setPIDparamSingle()	32
5.2.3.32	setROcenable()	32
5.2.3.33	setROclimits()	33
5.2.3.34	setWindAvg()	33
5.3	Access control functions	34
5.3.1	Detailed Description	34
5.3.2	Function Documentation	34
5.3.2.1	getLock()	34
5.3.2.2	getLockStatus()	34
5.4	PID control functions	35
5.4.1	Detailed Description	35
5.4.2	Function Documentation	35
5.4.2.1	getPIDconf()	35
5.4.2.2	getPIDstatus()	36
5.4.2.3	getSetpoint()	36
5.4.2.4	getSetpoint2()	36
5.4.2.5	IIRcontrollerCommitParams()	37
5.4.2.6	IIRcontrollerGetParam()	37
5.4.2.7	IIRcontrollerSetParam()	38

5.4.2.8	setCtrlReset()	38
5.4.2.9	setFBenable()	38
5.4.2.10	setPIDconf()	38
5.4.2.11	setSetpoint()	39
5.4.2.12	setSetpoint2()	39
5.5	Function generator functions	40
5.5.1	Detailed Description	40
5.5.2	Function Documentation	40
5.5.2.1	funcGenConfig()	40
5.5.2.2	funcGenEnable()	40
5.6	TetrAMM functions	41
5.6.1	Detailed Description	41
5.6.2	Function Documentation	41
5.6.2.1	getTetramms()	41
5.6.2.2	getTetrammsNumber()	41
5.6.2.3	tetrammHVenable()	42
5.6.2.4	tetrammHVSetVoltage()	42
5.6.2.5	tetrammSetRange()	42
5.7	EnBOX functions	43
5.7.1	Detailed Description	43
5.7.2	Function Documentation	43
5.7.2.1	enboxEncoderSelect()	43
5.7.2.2	getEnboxes()	43
5.7.2.3	getEnboxesNumber()	44
5.8	Misc	44
5.8.1	Detailed Description	44
5.8.2	Function Documentation	44
5.8.2.1	getSFPinfo()	44
5.8.2.2	getSystemVersion()	45
<b>6</b>	<b>Class Documentation</b>	<b>47</b>
6.1	best_buffer_line Struct Reference	47
6.1.1	Detailed Description	47
6.2	best_stats Struct Reference	47
6.2.1	Detailed Description	47
<b>7</b>	<b>File Documentation</b>	<b>49</b>
7.1	best_c_interface.c File Reference	49
7.1.1	Macro Definition Documentation	51
7.1.1.1	debug_print	51
7.1.1.2	LIB_BEST_DEBUG	51
7.1.1.3	VERSION_STRING	51
7.1.1.4	VERSION_STRING_HELPER	51

---

7.1.2 Function Documentation	51
7.1.2.1 __attribute__([1/2])	51
7.1.2.2 __attribute__([2/2])	51
7.1.2.3 getBestLock()	52
7.1.2.4 lock_sigaction()	52
7.1.2.5 releaseBestLock()	52
7.1.3 Variable Documentation	52
7.1.3.1 fd_DMA	52
7.1.3.2 fd_Lock	52
7.1.3.3 fd_Mbox	52
7.1.3.4 in	52
7.1.3.5 LIB_BEST_VERSION	52
7.1.3.6 loginLevel	52
7.1.3.7 out	53
7.1.3.8 p	53
7.1.3.9 SAMP_FREQ	53
7.2 best_c_interface.h File Reference	53
7.2.1 Macro Definition Documentation	57
7.2.1.1 BEST_FILE_CONFIG_INI	57
7.2.1.2 BEST_FILE_DISP_DMA	57
7.2.1.3 BEST_FILE_MAILBOX	57
7.2.1.4 FFT_LEN	57
7.2.1.5 LEN_BBF	57
7.2.1.6 LOCK_FILE	57
7.2.2 Variable Documentation	57
7.2.2.1 LIB_BEST_VERSION	57
7.3 best_c_interface.h	58
7.4 best_c_interface_conf.cpp File Reference	62
7.4.1 Variable Documentation	64
7.4.1.1 fd_DMA	64
7.4.1.2 fd_Mbox	64
7.4.1.3 in	64
7.4.1.4 loginLevel	64
7.4.1.5 out	64
7.4.1.6 p	64
7.4.1.7 SAMP_FREQ	64
7.5 best_c_interface_data.cpp File Reference	65
7.5.1 Function Documentation	66
7.5.1.1 fftw_abs()	66
7.5.2 Variable Documentation	66
7.5.2.1 fd_DMA	66
7.5.2.2 fd_Mbox	66

---

7.5.2.3 in . . . . .	66
7.5.2.4 loginLevel . . . . .	66
7.5.2.5 out . . . . .	66
7.5.2.6 p . . . . .	66
7.5.2.7 SAMP_FREQ . . . . .	67
7.6 best_c_interface_tetramm.cpp File Reference . . . . .	67
7.6.1 Variable Documentation . . . . .	68
7.6.1.1 fd_Mbox . . . . .	68
7.6.1.2 loginLevel . . . . .	68
7.7 changelog.md File Reference . . . . .	68
<b>Index</b>	<b>69</b>



# Chapter 1

## [2.0.0] - 2026/06/15

### 1.0.1 ADDED

- [EXT] First release for Ubuntu 24.04



# Chapter 2

## Topic Index

### 2.1 Topics

Here is a list of all topics with brief descriptions:

- Data functions . . . . . 9
- Configuration functions . . . . . 15
- Access control functions . . . . . 34
- PID control functions . . . . . 35
- Function generator functions . . . . . 40
- TetrAMM functions . . . . . 41
- EnBOX functions . . . . . 43
- Misc . . . . . 44



# Chapter 3

## Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">best_buffer_line</a> . . . . .	47
<a href="#">best_stats</a> . . . . .	47



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">best_c_interface.c</a>	49
<a href="#">best_c_interface.h</a>	53
<a href="#">best_c_interface_conf.cpp</a>	62
<a href="#">best_c_interface_data.cpp</a>	65
<a href="#">best_c_interface_tetramm.cpp</a>	67



# Chapter 5

## Topic Documentation

### 5.1 Data functions

#### Classes

- struct [best\\_buffer\\_line](#)
- struct [best\\_stats](#)

#### Functions

- struct `__attribute__((packed))` [best\\_buffer\\_line](#)
- int [getBuffer](#) (struct [best\\_buffer\\_line](#)[], int length)  
*Gets arrays of "samples" from display DMA.*
- int [getPosArray\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of positions and intensity from display DMA.*
- int [getPosScalar\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of positions and intensity from display DMA.*
- int [getPosArray](#) (double \*posX, double \*posY, double \*I0, unsigned int length)  
*Same as getPosArray\_sel, but only for 1st TetrAMM.*
- int [getVoltageArray](#) (double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages from display DMA.*
- int [getVoltageArray\\_sel](#) (int sel, double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).*
- int [getVoltageScalar\\_sel](#) (int sel, double \*ch0, double \*ch1, double \*ch2, double \*ch3, unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).*
- int [getCurrentArray](#) (double(\*currents)[4], unsigned int length)  
*Gets arrays of currents from display DMA.*
- int [getCurrentArray\\_sel](#) (int sel, double(\*currents)[4], unsigned int length)  
*Gets arrays of currents one of two TetrAMMs.*
- int [getFFT](#) (unsigned int selector, double \*outM, double \*outF, int removeDC)  
*Calculates FFT.*
- int [getPosStats](#) (int selector, double \*meanX, double \*stdX, double \*stdX\_filt, double \*meanY, double \*stdY, double \*stdY\_filt, double \*meanI0, double \*stdI0, double \*stdI0\_filt)  
*Get position statistics.*
- int [setDisplayFreq](#) (int freq, uint8\_t use\_filter)  
*Sets display frequency.*
- int [getDisplayFreq](#) (int \*freq, uint8\_t \*use\_filter)  
*Gets display frequency.*
- struct `__attribute__((packed))` [best\\_stats](#)
- int [getBPMStats](#) ([best\\_stats](#) \*stats)  
*Gets BPM Statistics.*

### 5.1.1 Detailed Description

Functions which read measurement and calculation data from system

### 5.1.2 Function Documentation

#### 5.1.2.1 `__attribute__()` [1/2]

```
struct __attribute__ (
    (__packed__)
```

#### 5.1.2.2 `__attribute__()` [2/2]

```
struct __attribute__ (
    (packed)
```

#### 5.1.2.3 `getBPMStats()`

```
int getBPMStats (
    best_stats * stats)
```

Gets BPM Statistics.

##### Parameters

<code>stats</code>	in <code>best_stats</code> structure format
--------------------	---

##### Returns

0 on success, -1 on fail

#### 5.1.2.4 `getBuffer()`

```
int getBuffer (
    struct best_buffer_line buf[],
    int length)
```

Gets arrays of "samples" from display DMA.

##### Parameters

<code>length</code>	number of samples (each sample is 256 bytes big)
---------------------	--

##### Returns

\_\_\_\_\_ numbers of samples copies (should be equal to length)

#### 5.1.2.5 `getCurrentArray()`

**Parameters**

<i>length</i>	Length of the arrays
---------------	----------------------

**Returns**

numbers of samples copies (should be equal to length)

**5.1.2.6 getCurrentArray\_sel()**

```
int getCurrentArray_sel (
    int sel,
    double(*) currents[4],
    unsigned int length)
```

Gets arrays of currents one of two TetrAMMs.

**Parameters**

<i>sel</i>	TetrAMM selector (0 or 1)
<i>length</i>	Length of the arrays

**Returns**

numbers of samples copies (should be equal to length), or EINVAL if wrong arguments

**5.1.2.7 getDisplayFreq()**

```
int getDisplayFreq (
    int * freq,
    uint8_t * use_filter)
```

Gets display frequency.

**Returns**

0 on success, -1 on fail

**5.1.2.8 getFFT()**

```
int getFFT (
    unsigned int selector,
    double * outM,
    double * outF,
    int removeDC)
```

Calculates FFT.

**Parameters**

<i>selector</i>	Selects source for FFT ( offset_adc0_ch0 = 24, offset_pos0_posX = 0, offset_pos0_posY = 1, offset_pos0_I0 = 2, offset_pos1_posX = 3, offset_pos1_posY = 4, offset_pos1_I0 = 5, offset_adc0_ch0 = 16, offset_adc0_ch1 = 17, offset_adc0_ch2 = 18, offset_adc0_ch3 = 19
<i>outM</i>	magnitude output (size <code>FFT_LEN / 2</code> )
<i>outF</i>	frequency output (size <code>FFT_LEN / 2</code> )
<i>removeDC</i>	removes DC component

**5.1.2.9 getPosArray()**

```
int getPosArray (
    double * posX,
    double * posY,
    double * I0,
    unsigned int length)
```

Same as `getPosArray_sel`, but only for 1st TetrAMM.

**5.1.2.10 getPosArray\_sel()**

```
int getPosArray_sel (
    int selector,
    double * posX,
    double * posY,
    double * I0,
    unsigned int length)
```

Gets arrays of positions and intensity from display DMA.

**Parameters**

<i>selector</i>	Selects BPM (= TetrAMM), should be 0 or 1
<i>posX</i>	Pointer to array where position X will be written
<i>posY</i>	Pointer to array where position Y will be written
<i>I0</i>	Pointer to array where Intensity will be written
<i>length</i>	Length of the arrays

**Returns**

numbers of samples copies (should be equal to length)

**5.1.2.11 getPosScalar\_sel()**

```
int getPosScalar_sel (
    int selector,
    double * posX,
    double * posY,
    double * I0,
    unsigned int length)
```

Gets arrays of positions and intensity from display DMA.

**Parameters**

<i>selector</i>	Selects BPM (= TetrAMM), should be 0 or 1
<i>posX</i>	Pointer to array where position X will be written
<i>posY</i>	Pointer to array where position Y will be written
<i>I0</i>	Pointer to array where Intensity will be written
<i>length</i>	Length of the arrays on which to perform the average

**Returns**

numbers of samples copies (should be equal to length)

**5.1.2.12 getPosStats()**

```
int getPosStats (
    int selector,
    double * meanX,
    double * stdX,
    double * stdX_filt,
    double * meanY,
    double * stdY,
    double * stdY_filt,
    double * meanI0,
    double * stdI0,
    double * stdI0_filt)
```

Get position statistics.

**Parameters**

<i>selector</i>	Selects which BPM
<i>mean</i>	mean value
<i>std</i>	standard deviation value (full bandwidth)
<i>std_filt</i>	standard deviation value (filtered at 100 Hz)

**5.1.2.13 getVoltageArray()**

```
int getVoltageArray (
    double(*) voltages[4],
    unsigned int length)
```

Gets arrays of voltages from display DMA.

**Parameters**

<i>length</i>	Length of the arrays
---------------	----------------------

**Returns**

numbers of samples copies (should be equal to length)

### 5.1.2.14 `getVoltageArray_sel()`

```
int getVoltageArray_sel (
    int sel,
    double(*) voltages[4],
    unsigned int length)
```

Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).

#### Parameters

<i>sel</i>	PreDAC selector (0, for the time beign)
<i>length</i>	Length of the arrays

#### Returns

numbers of samples copies (should be equal to length), or EINVAL if wrong arguments

### 5.1.2.15 `getVoltageScalar_sel()`

```
int getVoltageScalar_sel (
    int sel,
    double * ch0,
    double * ch1,
    double * ch2,
    double * ch3,
    unsigned int length)
```

Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).

#### Parameters

<i>sel</i>	PreDAC selector (0, for the time beign)
<i>length</i>	Length of the arrays on which to perform the average

#### Returns

numbers of samples copies (should be equal to length), or EINVAL if wrong arguments

### 5.1.2.16 `setDisplayFreq()`

```
int setDisplayFreq (
    int freq,
    uint8_t use_filter)
```

Sets display frequency.

**Parameters**

<i>freq</i>	Frequency in hertz
<i>use_filter</i>	enable filter to prevent aliasing

**Returns**

0 on success, -1 on fail

**Note**

Access level must be user

## 5.2 Configuration functions

**Enumerations**

- enum `best_pid_select` { `BEST_PID_SELECT_X` = 0 , `BEST_PID_SELECT_Y` = 1 , `BEST_PID_SELECT_I0` = 2 }

**Functions**

- int `getBPMscaling_sel` (int selector, double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int `getBPMscaling` (double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int `getROCandROI` (double \*roiX, double \*roiY, double \*roi0min, double \*roi0max, double \*roc)  
*Gets RoiX, RoiY, Roi0max, Roi0min and Roc of 1st BPM. Read values from CONFIG\_FILE.*
- int `getROCGenable` (bool \*rocX, bool \*rocY, bool \*rocl0, bool \*rocX2, bool \*rocY2, bool \*rocl02)  
*Gets enables for different ROC checks of 1st nad 2nd BPMs. Read values from FPGA.*
- int `setROCGenable` (bool rocX, bool rocY, bool rocl0, bool rocX2, bool rocY2, bool rocl02)  
*Sets enables for different ROC checks of 1st nad 2nd BPMs. Writes values to FPGA.*
- int `setOffsetSingle` (enum `best_pid_select` pid\_sel, double offset)  
*Sets output offset in [V] (output=this+PID value, updated even when PID is not running). Write offset to FPGA.*
- int `getOffsetSingle` (enum `best_pid_select` pid\_sel, double \*offset)  
*Gets output offset in [V]. Read offset from FPGA.*
- int `setOffset` (double offsetX, double offsetY, double offsetI0)  
*Sets output offsets in [V] (output=this+PID value, updated even when PID is not running).*
- int `setWindAvg` (enum `best_pid_select` pid\_sel, uint32\_t nr\_samp)  
*Sets number of samples for window averaging.*
- int `getWindAvg` (enum `best_pid_select` pid\_sel, uint32\_t \*nr\_samp)  
*Gets number of samples for window averaging. Read number of samples from FPGA.*
- int `setPIDparams` (char pid\_sel, double Kp, double Ki, double Kd, double emin, double lmax, double Omin, double Omax, double Ogain)  
*Sets PID parameters (legacy).*
- int `setPIDparamSingle` (char pid\_sel, int pid\_par, double param\_value)  
*Sets single PID parameter of specific PID. Write PID parameter to FPGA.*
- int `getPIDparamSingle` (char pid\_sel, int pid\_par, double \*param\_value)

- Gets PID parameters. Read PID parameters from FPGA.*

  - int [setPIDOutSel](#) (uint8\_t posX, uint8\_t posY, uint8\_t I0)  
*Sets PID output cross switch. Selects which PID is assigned to which PreDAC output channel. Write PID output cross switch to FPGA.*
  - int [getPIDOutSel](#) (uint8\_t \*posX, uint8\_t \*posY, uint8\_t \*I0)  
*Gets PID output cross switch. Read PID output cross switch from FPGA.*
  - int [setCrossBar](#) (uint8\_t sel, uint8\_t ch0, uint8\_t ch1, uint8\_t ch2, uint8\_t ch3)  
*Sets input cross switch.*
  - int [setBPMorient](#) (uint8\_t sel, uint8\_t is90deg)  
*Sets BPM orientation for diffOverSum.*
  - int [setOutputMux](#) (uint8\_t sel)  
*Sets value of output mux.*
  - int [getOutputMux](#) (uint8\_t \*sel)  
*Gets value of output mux.*
  - int [setBPMscale](#) (double scaleX, double scaleY, int bpm)  
*Sets BPM scaling parameters for position X and Y in [um]. Write BPM scaling parameters to FPGA.*
  - int [getBPMscale](#) (int selector, double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters for position X and Y in [um]. Read BPM scaling parameters from FPGA.*
  - int [setBPMoffset](#) (double offsetX, double offsetY, int bpm)  
*Sets BPM offset in [um] ONLY FOR TETRAMMs. Write BPM offsets to FPGA.*
  - int [setBPMoffsetEnbox](#) (double offsetX, double offsetY, int bpm, int encType)  
*Sets BPM offset in [um] ONLY FOR Enboxes. Write BPM offsets to FPGA. (>= 1.8-16).*
  - int [setBPMdevice](#) (uint8\_t sel, uint8\_t isEnbox)  
*Sets BPM device for diffOverSum. Write to FPGA. (>= 1.8-18).*
  - int [setEnboxType](#) (uint8\_t sel, uint8\_t encType)  
*Sets the EnBOX type (relative or absolute) . Write to FPGA. (>= 1.8-18).*
  - int [setEnboxSumDiff](#) (uint8\_t sel, uint8\_t isSumDiff)  
*Set diffOverSum calculations (for EnBOX). 0:Normal the positions are calculated from enc1 and enc2. 1:SumDiff the positions are calculated from enc1+enc2 and enc1-enc2. Write to FPGA. (>= 1.8-18).*
  - int [getBPMoffset](#) (double \*offsetX, double \*offsetY, int bpm)  
*Gets BPM offsets in [um] ONLY FOR TETRAMMs. Read BPM offsets from FPGA.*
  - int [getBPMoffsetEnbox](#) (double \*offsetX, double \*offsetY, int bpm, int encType)  
*Gets BPM offsets in [um] ONLY FOR Enboxes. Read BPM offsets from FPGA.*
  - int [getBPMorient](#) (uint8\_t sel, uint8\_t \*is90deg)  
*Sets BPM orientation for diffOverSum. Read BPM orientation from FPGA.*
  - int [getBPMselector](#) (int \*posX, int \*posY, int \*I0)  
*Gets BPM sources. Read BPM sources from FPGA.*
  - int [setBPMselector](#) (int posX, int posY, int I0)  
*Sets BPM sources. Write BPM sources to FPGA.*
  - int [getCrossBar](#) (uint8\_t sel, uint8\_t \*ch0, uint8\_t \*ch1, uint8\_t \*ch2, uint8\_t \*ch3)  
*Gets input cross switch. Read input cross switch from FPGA.*
  - int [setROClimits](#) (int sel, double min, double max)  
*Sets ROC min and max values. ROC should be in [um] for positions and in [A] for intensity. Write min and max ROC levels of specific axis to FPGA.*
  - int [getROClimits](#) (int sel, double \*min, double \*max)  
*Gets ROC. ROC values are in [um] for positions and in [A] for intensity. Read min and max ROC levels of specific axis from FPGA.*

## 5.2.1 Detailed Description

Functions which configure system

## 5.2.2 Enumeration Type Documentation

### 5.2.2.1 best\_pid\_select

```
enum best_pid_select
```

#### Enumerator

BEST_PID_SELECT_X	
BEST_PID_SELECT_Y	
BEST_PID_SELECT_I0	

## 5.2.3 Function Documentation

### 5.2.3.1 getBPMoffset()

```
int getBPMoffset (
    double * offsetX,
    double * offsetY,
    int bpm)
```

Gets BPM offsets in [um] ONLY FOR TETRAMMs. Read BPM offsets from FPGA.

#### Parameters

<i>offsetX</i>	in [um]
<i>offsetY</i>	in [um]
<i>bpm</i>	Selects BPM (= TetrAMM), should be 0 or 1

#### Returns

0 on success, negative on fail

### 5.2.3.2 getBPMoffsetEnbox()

```
int getBPMoffsetEnbox (
    double * offsetX,
    double * offsetY,
    int bpm,
    int encType)
```

Gets BPM offsets in [um] ONLY FOR Enboxes. Read BPM offsets from FPGA.

**Parameters**

<i>offsetX</i>	in [um]
<i>offsetY</i>	in [um]
<i>bpm</i>	Selects BPM (= TetrAMM), should be 0 or 1
<i>encType,-</i>	Relative (tonic), 1: Absolute (Resolute)

**Returns**

0 on success, negative on fail

**5.2.3.3 getBPMorient()**

```
int getBPMorient (
    uint8_t sel,
    uint8_t * is90deg)
```

Sets BPM orientation for diffOverSum. Read BPM orientation from FPGA.

**Parameters**

<i>sel</i>	Selects TetrAmm (in case of 2 TetrAmms)
<i>is90deg</i>	(0 = 90deg, 1 = 45deg)

**Returns**

0 on success, negative on fail

**Note**

Access level: all levels

**5.2.3.4 getBPMscale()**

```
int getBPMscale (
    int selector,
    double * scaleX,
    double * scaleY)
```

Gets BPM scaling parameters for position X and Y in [um]. Read BPM scaling parameters from FPGA.

**Parameters**

<i>selector</i>	Selects BPM (= TetrAMM), should be 0 or 1
-----------------	---

**Returns**

0 on success, -1 on fail

**Note**

Access level: all levels

### 5.2.3.5 getBPMscaling()

```
int getBPMscaling (
    double * scaleX,
    double * scaleY)
```

Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG\_FILE.

#### Parameters

<i>scaleX</i>	scaling value for X in [um]
<i>scaleY</i>	scaling value for Y in [um]

#### Returns

0 on success, -1 on fail

### 5.2.3.6 getBPMscaling\_sel()

```
int getBPMscaling_sel (
    int selector,
    double * scaleX,
    double * scaleY)
```

Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG\_FILE.

#### Parameters

<i>selector</i>	Selects BPM (= TetrAMM or EnBOX), 0 for 1st BPM, 1 for 2nd BPM
<i>scaleX</i>	pointer to scaling value for X in [um]
<i>scaleY</i>	pointer to scaling value for Y in [um]

#### Returns

0 on success, -1 on fail

### 5.2.3.7 getBPMselector()

```
int getBPMselector (
    int * posX,
    int * posY,
    int * IO)
```

Gets BPM sources. Read BPM sources from FPGA.

**Parameters**

<i>posX</i>	BPM (= TetrAMM) for posX
<i>posY</i>	BPM (= TetrAMM) for posY
<i>I0</i>	BPM (= TetrAMM) for I0

**Returns**

ACK

**Note**

Access level: all levels

**5.2.3.8 getCrossBar()**

```
int getCrossBar (
    uint8_t sel,
    uint8_t * ch0,
    uint8_t * ch1,
    uint8_t * ch2,
    uint8_t * ch3)
```

Gets input cross switch. Read input cross switch from FPGA.

**Parameters**

<i>sel</i>	Selects TetrAmm (in case of 2 TetrAmms)
<i>ch0</i>	Top / Top-Left
<i>ch1</i>	Right / Top-Right
<i>ch2</i>	Bottom / Bottom-Right
<i>ch3</i>	Left / Bottom-Left

**Returns**

0 on success, negative on fail

**5.2.3.9 getOffsetSingle()**

```
int getOffsetSingle (
    enum best_pid_select pid_sel,
    double * offset)
```

Gets output offset in [V]. Read offset from FPGA.

**Parameters**

<i>pid_sel</i>	PID select enum
<i>offset</i>	offset value in [V]

**Returns**

0 on success, negative on fail

**Note**

Read from FPGA

Access level: all levels

**5.2.3.10 getOutputMux()**

```
int getOutputMux (
    uint8_t * sel)
```

Gets value of output mux.

**Parameters**

<i>sel</i>	Selects between FPGA (value 1) and software (value 0)
------------	---

**Returns**

0 on success, negative on fail

**Note**

Access level: all levels

**5.2.3.11 getPIDoutSel()**

```
int getPIDoutSel (
    uint8_t * posX,
    uint8_t * posY,
    uint8_t * IO)
```

Gets PID output cross switch. Read PID output cross switch from FPGA.

**Parameters**

<i>posX</i>	DAC channel for posX
-------------	----------------------

<i>posY</i>	DAC channel for posY
<i>I0</i>	DAC channel for I0

**Returns**

0 on success, negative on fail

**Note**

Access level: all levels

**5.2.3.12 getPIDparamSingle()**

```
int getPIDparamSingle (
    char pid_sel,
    int pid_par,
    double * param_value)
```

Gets PID parameters. Read PID parameters from FPGA.

**Parameters**

<i>pid_sel</i>	(0 = X, 1 = Y, 2 = I0)
<i>pid_par</i>	(0=Kp, 1=Ki, 2=Kd, 3=emin, 4=lmax, 5=Omin, 6=Omax, 7=Ogain)
<i>param_value</i>	

**Returns**

0 on success, negative on fail

**Note**

Access level: all levels

**5.2.3.13 getROCandROI()**

```
int getROCandROI (
    double * roiX,
    double * roiY,
    double * roiI0min,
    double * roiI0max,
    double * roc)
```

Gets RoiX, RoiY, RoiI0max, RoiI0min and Roc of 1st BPM. Read values from CONFIG\_FILE.

**Parameters**

<i>roiX</i>	double parameter in [um]
<i>roiY</i>	double parameter in [um]
<i>roil0min</i>	double parameter in [A]
<i>roil0max</i>	double parameter in [A]
<i>roc</i>	double parameter in percentage

**Note**

Access level: all levels

**5.2.3.14 getROCEnable()**

```
int getROCEnable (
    bool * rocX,
    bool * rocY,
    bool * rocI0,
    bool * rocX2,
    bool * rocY2,
    bool * rocI02)
```

Gets enables for different ROC checks of 1st nad 2nd BPMs. Read values from FPGA.

**Parameters**

<i>rocX</i>	enable rocX
<i>rocY</i>	enable rocY
<i>rocl0</i>	enable rocl0
<i>rocX2</i>	enable rocX2
<i>rocY2</i>	enable rocY2
<i>rocl02</i>	enable rocl02

**Returns**

0 on success, -1 on fail

**Note**

Access level: all levels

### 5.2.3.15 getROClimits()

```
int getROClimits (
    int sel,
    double * min,
    double * max)
```

Gets ROC. ROC values are in [um] for positions and in [A] for intensity. Read min and max ROC levels of specific axis from FPGA.

#### Parameters

<i>sel</i>	(0 = X, 1 = Y, 2 = I0, 0 = X2, 1 = Y2, 2 = I02)
<i>min</i>	
<i>max</i>	

#### Returns

0 on success, negative on fail

### 5.2.3.16 getWindAvg()

```
int getWindAvg (
    enum best_pid_select pid_sel,
    uint32_t * nr_samp)
```

Gets number of samples for window averaging. Read number of samples from FPGA.

#### Parameters

<i>sel</i>	(0 = X, 1 = Y, 2 = I0)
<i>nr_samp</i>	Number of samples

#### Returns

0 on success, negative on fail

### 5.2.3.17 setBPMdevice()

```
int setBPMdevice (
    uint8_t sel,
    uint8_t isEnbox)
```

Sets BPM device for diffOverSum. Write to FPGA. (>= 1.8-18).

**Parameters**

<i>sel</i>	Selects the BPM
<i>isEnbox</i>	

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.18 setBPMoffset()**

```
int setBPMoffset (
    double offsetX,
    double offsetY,
    int bpm)
```

Sets BPM offset in [um] ONLY FOR TETRAMMs. Write BPM offsets to FPGA.

**Parameters**

<i>offsetX</i>	in [um]
<i>offsetY</i>	in [um]
<i>bpm</i>	Selects BPM (= TetrAMM), should be 0 or 1

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.19 setBPMoffsetEnbox()**

```
int setBPMoffsetEnbox (
    double offsetX,
    double offsetY,
    int bpm,
    int encType)
```

Sets BPM offset in [um] ONLY FOR Enboxes. Write BPM offsets to FPGA. (>= 1.8-16).

**Parameters**

<i>offsetX</i>	in [um]
<i>offsetY</i>	in [um]
<i>bpm</i>	Selects BPM (= Enbox), should be 0 or 1
<i>encType,-</i>	Relative (tonic), 1: Absolute (Resolute)

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.20 setBPMorient()**

```
int setBPMorient (
    uint8_t sel,
    uint8_t is90deg)
```

Sets BPM orientation for diffOverSum.

**Parameters**

<i>sel</i>	Selects TetrAmm (in case of 2 TetrAmms)
<i>is90deg</i>	

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.21 setBPMscale()**

```
int setBPMscale (
    double scaleX,
    double scaleY,
    int bpm)
```

Sets BPM scaling parameters for position X and Y in [um]. Write BPM scaling parameters to FPGA.

**Parameters**

<i>scaleX</i>	in [ $\mu\text{m}$ ]
<i>scaleY</i>	in [ $\mu\text{m}$ ]
<i>bpm</i>	Selects BPM (= TetrAMM), should be 0 or 1

**Returns**

0 on success, -1 on fail

**Note**

Access level must be admin

**5.2.3.22 setBPMselector()**

```
int setBPMselector (
    int posX,
    int posY,
    int IO)
```

Sets BPM sources. Write BPM sources to FPGA.

**Parameters**

<i>posX</i>	BPM (= TetrAMM) for posX
<i>posY</i>	BPM (= TetrAMM) for posY
<i>IO</i>	BPM (= TetrAMM) for IO

**Returns**

ACK

**Note**

Access level: all levels

**5.2.3.23 setCrossBar()**

```
int setCrossBar (
    uint8_t sel,
    uint8_t ch0,
    uint8_t ch1,
    uint8_t ch2,
    uint8_t ch3)
```

**Parameters**

<i>sel</i>	Selects TetrAmm (in case of 2 TetrAmms)
<i>ch0</i>	
<i>ch1</i>	
<i>ch2</i>	
<i>ch3</i>	

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.24 setEnboxSumDiff()**

```
int setEnboxSumDiff (
    uint8_t sel,
    uint8_t isSumDiff)
```

Set diffOverSum calculations (for EnBOX). 0:Normal the positions are calculated from enc1 and enc2. 1:SumDiff the positions are calculated from enc1+enc2 and enc1-enc2. Write to FPGA. (>= 1.8-18).

**Parameters**

<i>sel</i>	Selects the BPM
<i>encType,-</i>	Normal (enc1, enc2), 1: SumDiff (enc1+enc2, enc1-enc2)

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.25 setEnboxType()**

```
int setEnboxType (
    uint8_t sel,
    uint8_t encType)
```

Sets the EnBOX type (relative or absolute) . Write to FPGA. (>= 1.8-18).

**Parameters**

<i>sel</i>	Selects the BPM
<i>encType,-</i>	Relative (tonic), 1: Absolute (Resolute)

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.26 setOffset()**

```
int setOffset (
    double offsetX,
    double offsetY,
    double offsetIO)
```

Sets output offsets in [V] (output=this+PID value, updated even when PID is not running).

**Parameters**

<i>offsetX</i>	in [V]
<i>offsetY</i>	in [V]
<i>offsetIO</i>	in [V]

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.27 setOffsetSingle()**

```
int setOffsetSingle (
    enum best_pid_select pid_sel,
    double offset)
```

Sets output offset in [V] (output=this+PID value, updated even when PID is not running). Write offset to FPGA.

**Parameters**

<i>pid_sel</i>	PID select enum
<i>offset</i>	offset value in [V]

**Returns**

0 on success, negative on fail

**Note**

Write to FPGA

Access level must be admin

**5.2.3.28 setOutputMux()**

```
int setOutputMux (
    uint8_t sel)
```

Sets value of output mux.

**Parameters**

<i>sel</i>	Selects between FPGA (value 1) and software (value 0)
------------	---

**Returns**

0 on success, negative on fail

**Note**

Access level must be user or admin

**5.2.3.29 setPIDoutSel()**

```
int setPIDoutSel (
    uint8_t posX,
    uint8_t posY,
    uint8_t IO)
```

Sets PID output cross switch. Selects which PID is assigned to which PreDAC output channel. Write PID output cross switch to FPGA.

**Parameters**

<i>posX</i>	DAC channel for posX
<i>posY</i>	DAC channel for posY
<i>I0</i>	DAC channel for I0

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

**5.2.3.30 setPIDparams()**

```
int setPIDparams (
    char pid_sel,
    double Kp,
    double Ki,
    double Kd,
    double emin,
    double Imax,
    double Omin,
    double Omax,
    double Ogain)
```

Sets PID parameters (legacy).

**Parameters**

<i>pid_sel</i>	(0 = X, 1 = Y, 2 = I0)
<i>Kp</i>	See BEST User's Manual for detailed explanation
<i>Ki</i>	See BEST User's Manual for detailed explanation
<i>Kd</i>	See BEST User's Manual for detailed explanation
<i>emin</i>	See BEST User's Manual for detailed explanation
<i>Imax</i>	See BEST User's Manual for detailed explanation
<i>Omin</i>	See BEST User's Manual for detailed explanation
<i>Omax</i>	See BEST User's Manual for detailed explanation
<i>Ogain</i>	See BEST User's Manual for detailed explanation

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

### 5.2.3.31 setPIDparamSingle()

```
int setPIDparamSingle (
    char pid_sel,
    int pid_par,
    double param_value)
```

Sets single PID parameter of specific PID. Write PID parameter to FPGA.

#### Parameters

<i>pid_sel</i>	(0 = X, 1 = Y, 2 = I0)
<i>pid_par</i>	(0=Kp, 1=Ki, 2=Kd, 3=emin, 4=lmax, 5=Omin, 6=Omax, 7=Ogain)
<i>param_value</i>	

#### Returns

0 on success, negative on fail

#### Note

Access level must be admin

### 5.2.3.32 setROCEnable()

```
int setROCEnable (
    bool rocX,
    bool rocY,
    bool rocI0,
    bool rocX2,
    bool rocY2,
    bool rocI02)
```

Sets enables for different ROC checks of 1st nad 2nd BPMs. Writes values to FPGA.

#### Parameters

<i>rocX</i>	enable rocX
<i>rocY</i>	enable rocY
<i>rocI0</i>	enable rocl0
<i>rocX2</i>	enable rocX2
<i>rocY2</i>	enable rocY2
<i>rocI02</i>	enable rocl02

#### Returns

0 on success, -1 on fail

#### Note

Access level must be admin

**5.2.3.33 setROClimits()**

```
int setROClimits (
    int sel,
    double min,
    double max)
```

Sets ROC min and max values. ROC should be in [um] for positions and in [A] for intensity. Write min and max ROC levels of specific axis to FPGA.

**Parameters**

<i>sel</i>	(0 = X, 1 = Y, 2 = I0, 3 = X2, 4 = Y2, 5 = I02)
<i>min</i>	
<i>max</i>	

**Returns**

0 on success, negative on fail

**Note**

Access level: access level user or admin

**5.2.3.34 setWindAvg()**

```
int setWindAvg (
    enum best_pid_select pid_sel,
    uint32_t nr_samp)
```

Sets number of samples for window averaging.

**Parameters**

<i>sel</i>	(0 = X, 1 = Y, 2 = I0)
<i>nr_samp</i>	Number of samples

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

## 5.3 Access control functions

### Functions

- int [getLock](#) (const char \*usr, const char \*pass)  
*Acquires lock on /var/lock/best\_lock.*
- int [getLockStatus](#) ()  
*Gets login level, which can be changed with call on [getLock\(\)](#).*

### 5.3.1 Detailed Description

Functions which allow to elevate user access level

### 5.3.2 Function Documentation

#### 5.3.2.1 getLock()

```
int getLock (  
    const char * usr,  
    const char * pass)
```

Acquires lock on /var/lock/best\_lock.

#### Parameters

<i>usr</i>	Username
<i>pass</i>	Password

#### Returns

login level (0=cruise, 1=user, 2=admin)

#### Note

Access level: all levels

#### 5.3.2.2 getLockStatus()

```
int getLockStatus ()
```

Gets login level, which can be changed with call on [getLock\(\)](#).

#### Returns

login level (0=cruise, 1=user, 2=admin)

#### Note

Access level: all levels

## 5.4 PID control functions

### Functions

- int `getPIDstatus` (void)  
*Gets PID status.*
- int `setFBenable` (int enable)  
*Enables or disables feedback.*
- int `setCtrlReset` ()  
*Resets internal states of controller.*
- int `setPIDconf` (char conf)  
*Sets PID controller configuration. Write PID controller configuration to FPGA.*
- int `getPIDconf` (char \*conf)  
*Sets PID controller configuration. Read PID configuration from FPGA.*
- int `setSetpoint` (char sel, double setpt)  
*Sets new setpoint in [um]. Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from CONFIG\_FILE.*
- int `getSetpoint` (char sel, double \*setpt)  
*Gets setpoint in [um]. Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from CONFIG\_FILE.*
- int `setSetpoint2` (char sel, double setpt)  
*Sets new setpoint (without using config file). Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from FPGA.*
- int `getSetpoint2` (char sel, double \*setpt)  
*Gets new setpoint (without using config file). Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from FPGA.*
- int `IIRcontrollerSetParam` (char ctrlr\_sel, char param\_is\_a, char param\_sel, double param)  
*Sets IIR controller parameter.*
- int `IIRcontrollerCommitParams` (char ctrlr\_sel)  
*Commits IIR controller parameters.*
- int `IIRcontrollerGetParam` (char ctrlr\_sel, char param\_is\_a, char param\_sel, double \*param)  
*Reads IIR controller parameter.*

### 5.4.1 Detailed Description

Functions which control PID behaviour

### 5.4.2 Function Documentation

#### 5.4.2.1 `getPIDconf()`

```
int getPIDconf (
    char * conf)
```

Sets PID controller configuration. Read PID configuration from FPGA.

**Parameters**

<i>conf</i>	return PID conf
-------------	-----------------

**Returns**

0 on success, negative on fail

**5.4.2.2 getPIDstatus()**

```
int getPIDstatus (
    void )
```

Gets PID status.

**Returns**

stoppedByUser = 0, stoppedByROC = 1, paused = 2, running = 3

**Note**

Access level: all levels

**5.4.2.3 getSetpoint()**

```
int getSetpoint (
    char sel,
    double * setpt)
```

Gets setpoint in [um]. Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from CONFIG\_FILE.

**Parameters**

<i>sel</i>	Selects which PID ( X = 0, Y = 1, I0 = 2 )
<i>setpt</i>	Setpoint in [um]

**Returns**

0 on success, negative on fail

**Note**

Access level: all levels

**5.4.2.4 getSetpoint2()**

```
int getSetpoint2 (
    char sel,
```

**Parameters**

<i>sel</i>	Selects which PID ( X = 0, Y = 1, I0 = 2 )
<i>setpt</i>	Setpoint in [um]

**Returns**

0 on success, negative on fail

**5.4.2.5 IIRcontrollerCommitParams()**

```
int IIRcontrollerCommitParams (
    char ctrlr_sel)
```

Commits IIR controller parameters.

**Parameters**

<i>ctrlr_sel</i>	Selects controller ( X = 0, Y = 1, I0 = 2 )
------------------	---

**Returns**

0 on success, negative on fail

**5.4.2.6 IIRcontrollerGetParam()**

```
int IIRcontrollerGetParam (
    char ctrlr_sel,
    char param_is_a,
    char param_sel,
    double * param)
```

Reads IIR controller parameter.

**Parameters**

<i>ctrlr_sel</i>	Selects controller ( X = 0, Y = 1, I0 = 2 )
<i>param_is_a</i>	Selects if param read is a (when 1) or b (when 0)
<i>param_sel</i>	Selects parameter (0-9)
<i>param</i>	IIR coefficient

**Returns**

0 on success, negative on fail

This function returns the parameter from read-back storage (the parameters which are actually being used by IIR controller).

### 5.4.2.7 IIRcontrollerSetParam()

```
int IIRcontrollerSetParam (
    char ctrlr_sel,
    char param_is_a,
    char param_sel,
    double param)
```

Sets IIR controller parameter.

#### Parameters

<i>ctrlr_sel</i>	Selects controller ( X = 0, Y = 1, I0 = 2 )
<i>param_is_a</i>	Selects if param written is a (when 1) or b (when 0)
<i>param_sel</i>	Selects parameter (0-9)
<i>param</i>	IIR coefficient

#### Returns

0 on success, negative on fail

This function stores the parameter in the temporary storage. After all parameters have been set use IIRcontrollerCommitParams function to download parameters to controller.

### 5.4.2.8 setCtrlReset()

```
int setCtrlReset ()
```

Resets internal states of controller.

#### Returns

0 on success, negative on fail (e.g. authentication)

#### Note

Access level must be user or admin

### 5.4.2.9 setFBenable()

```
int setFBenable (
    int enable)
```

Enables or disables feedback.

#### Returns

0 on success, negative on fail (e.g. authentication)

#### Note

Access level must be user or admin

**Parameters**

<i>conf</i>	PID conf (X = 0, XY = 1, XIO = 2, Y = 3, YIO = 4, XYIO = 5, IO = 6)
-------------	---

**Returns**

0 on success, negative on fail (e.g. authentication)

**Note**

Access level must be user or admin

**5.4.2.11 setSetpoint()**

```
int setSetpoint (
    char sel,
    double setpt)
```

Sets new setpoint in [um]. Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from CONFIG\_FILE.

**Parameters**

<i>sel</i>	Selects which PID ( X = 0, Y = 1, IO = 2 )
<i>setpt</i>	Setpoint in [um]

**Returns**

0 on success, negative on fail (e.g. authentication)

**Note**

Access level must be user or admin

**5.4.2.12 setSetpoint2()**

```
int setSetpoint2 (
    char sel,
    double setpt)
```

Sets new setpoint (without using config file). Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from FPGA.

**Parameters**

<i>sel</i>	Selects which PID ( X = 0, Y = 1, I0 = 2 )
<i>setpt</i>	Setpoint in [um]

**Returns**

0 on success, negative on fail (e.g. authentication)

**Note**

Access level must be user or admin

## 5.5 Function generator functions

**Functions**

- int [funcGenConfig](#) (float freq, float ampl[4], float offset[4])  
*Configures function generator.*
- int [funcGenEnable](#) (char enable)  
*Enables or disables function generator.*

### 5.5.1 Detailed Description

### 5.5.2 Function Documentation

#### 5.5.2.1 funcGenConfig()

```
int funcGenConfig (
    float freq,
    float ampl[4],
    float offset[4])
```

Configures function generator.

**Parameters**

<i>freq</i>	Frequency (in Hz)
<i>ampl</i>	Amplitude of sine wave
<i>offset</i>	Offset of sine wave

**Returns**

0 on success, negative on fail

configuration is committed when the func gen is enabled. if func gen is already running, just call funcGenEnable function again

#### 5.5.2.2 funcGenEnable()

**Parameters**

<i>enable</i>	0 to disable, 1 to enable
---------------	---------------------------

## 5.6 TetrAMM functions

**Functions**

- int [tetrammSetRange](#) (int range, unsigned char selector)  
*Sets front-end range.*
- int [tetrammHVSetVoltage](#) (float voltage, unsigned char sel)  
*Sets TetrAMM HV voltage.*
- int [tetrammHVenable](#) (int enable, unsigned char sel)  
*Enables or disables TetrAMM HV module.*
- unsigned int [getTetrammsNumber](#) ()  
*Gets number of TetrAMMs.*
- int [getTetramms](#) (int \*pos\_A, int \*pos\_B)  
*Gets TetrAMMs. (>= 1.8-14).*

### 5.6.1 Detailed Description

### 5.6.2 Function Documentation

#### 5.6.2.1 getTetramms()

```
int getTetramms (
    int * pos_A,
    int * pos_B)
```

Gets TetrAMMs. (>= 1.8-14).

**Parameters**

<i>pos</i> <i>_A</i>	on SFP A, 0 = No TetrAMM connected, 1 = TetrAMM connected
<i>pos</i> <i>_A</i>	on SFP B, 0 = No TetrAMM connected, 1 = TetrAMM connected

**Returns**

0 on success, negative on fail

#### 5.6.2.2 getTetrammsNumber()

```
unsigned int getTetrammsNumber ()
```

Gets number of TetrAMMs.

**Returns**

TetrAMMs number

### 5.6.2.3 tetrammHVenable()

```
int tetrammHVenable (
    int enable,
    unsigned char sel)
```

Enables or disables TetrAMM HV module.

#### Parameters

<i>enable</i>	1 to enable, 0 to disable
<i>sel</i>	0 = TetrAMM on SFP A, 1 = TetrAMM on SFP B

#### Note

Access level must be admin

### 5.6.2.4 tetrammHVSetVoltage()

```
int tetrammHVSetVoltage (
    float voltage,
    unsigned char sel)
```

Sets TetrAMM HV voltage.

#### Parameters

<i>voltage</i>	voltage in volts
<i>sel</i>	0 = TetrAMM on SFP A, 1 = TetrAMM on SFP B

#### Note

Access level must be admin

### 5.6.2.5 tetrammSetRange()

```
int tetrammSetRange (
    int range,
    unsigned char selector)
```

Sets front-end range.

#### Parameters

<i>range</i>	0 = RNG0, 1 = RNG1
<i>selector</i>	0 = TetrAMM on SFP A, 1 = TetrAMM on SFP B

#### Note

Access level must be admin

## 5.7 EnBOX functions

### Functions

- int `enboxEncoderSelect` (int `enc_type`, unsigned char `selector`)
- unsigned int `getEnboxesNumber` ()  
*Gets number of Enbox. (>= 1.8-15).*
- int `getEnboxes` (int `*pos_A`, int `*pos_B`)  
*Gets Enbox. (>= 1.8-15).*

### 5.7.1 Detailed Description

### 5.7.2 Function Documentation

#### 5.7.2.1 `enboxEncoderSelect()`

```
int enboxEncoderSelect (
    int enc_type,
    unsigned char selector)
```

#### Parameters

<code>enc_type</code>	0 = Tonic, 1 = Absolute
<code>selector</code>	0 = EnBOX on SFP A, 1 = EnBOX on SFP B

#### Note

Access level must be admin

#### 5.7.2.2 `getEnboxes()`

```
int getEnboxes (
    int * pos_A,
    int * pos_B)
```

Gets Enbox. (>= 1.8-15).

#### Parameters

<code>pos↔ _A</code>	on SFP A, 0 = No Enbox connected, 1 = Enbox connected
<code>pos↔ _B</code>	on SFP B, 0 = No Enbox connected, 1 = Enbox connected

#### Returns

0 on success, negative on fail

### 5.7.2.3 getEnboxesNumber()

```
unsigned int getEnboxesNumber ()
```

Gets number of Enbox. (>= 1.8-15).

#### Returns

Enboxes number

## 5.8 Misc

### Functions

- int [getSFPinfo](#) (int selector, uint16\_t \*id, uint32\_t \*timestamp, uint32\_t \*fw\_ver, uint32\_t \*ser\_nr)  
*Gets SFP info.*
- int [getSystemVersion](#) (uint32\_t \*hwVer, uint32\_t \*swVer, uint32\_t \*ctrlVer)  
*Gets System HW, SW and PID versions.*

### 5.8.1 Detailed Description

### 5.8.2 Function Documentation

#### 5.8.2.1 getSFPinfo()

```
int getSFPinfo (
    int selector,
    uint16_t * id,
    uint32_t * timestamp,
    uint32_t * fw_ver,
    uint32_t * ser_nr)
```

Gets SFP info.

#### Parameters

<i>selector</i>	Selects which SFP port (A=0, B=1, ...)
<i>id</i>	
<i>timestamp</i>	
<i>fw_ver</i>	
<i>ser_nr</i>	

#### Returns

0 on success, -1 on fail

#### Note

Access level: all levels

### 5.8.2.2 `getSystemVersion()`

```
int getSystemVersion (  
    uint32_t * hwVer,  
    uint32_t * swVer,  
    uint32_t * ctrlVer)
```

Gets System HW, SW and PID versions.

#### Parameters

<i>hwVer</i>	hardware version
<i>swVer</i>	software version (main application)
<i>ctrlVer</i>	controller version (pid version)

#### Returns

0 on success, -1 on fail

#### Note

Access level: all levels



# Chapter 6

## Class Documentation

### 6.1 `best_buffer_line` Struct Reference

```
#include <best_c_interface.h>
```

#### 6.1.1 Detailed Description

best DMA buffer struct

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

- [best\\_c\\_interface.h](#)

### 6.2 `best_stats` Struct Reference

```
#include <best_c_interface.h>
```

#### 6.2.1 Detailed Description

best statistics data structure

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

- [best\\_c\\_interface.h](#)



# Chapter 7

## File Documentation

### 7.1 best\_c\_interface.c File Reference

```
#include "best_c_interface.h"
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include "pcie_driver/BEST_PCIE.h"
#include "pcie_mailbox/mailbox_comm_defs.h"
#include <iostream>
#include "inih/cpp/INIReader.h"
#include <complex>
#include <fftw3.h>
#include <signal.h>
```

#### Macros

- #define [VERSION\\_STRING\\_HELPER\(X\)](#)
- #define [VERSION\\_STRING\(X\)](#)
- #define [LIB\\_BEST\\_DEBUG](#) 0
- #define [debug\\_print](#)(fmt, ...)

#### Functions

- int [getBestLock](#) (int fd)  
*!!! All sets should check loginLevel*
- int [releaseBestLock](#) (int fd)
- void [lock\\_sigaction](#) (int sig, siginfo\_t \*siginfo, void \*context)
- void [\\_\\_attribute\\_\\_](#) ((constructor))
- void [\\_\\_attribute\\_\\_](#) ((destructor))

- int [getLock](#) (const char \*usr, const char \*pass)  
*Acquires lock on /var/lock/best\_lock.*
- int [getLockStatus](#) ()  
*Gets login level, which can be changed with call on [getLock\(\)](#).*
- int [setFBenable](#) (int enable)  
*Enables or disables feedback.*
- int [setCtrlReset](#) ()  
*Resets internal states of controller.*
- int [setPIDconf](#) (char conf)  
*Sets PID controller configuration. Write PID controller configuration to FPGA.*
- int [getPIDconf](#) (char \*conf)  
*Sets PID controller configuration. Read PID configuration from FPGA.*
- int [setSetpoint](#) (char sel, double setpt)  
*Sets new setpoint in [um]. Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from CONFIG\_FILE.*
- int [setSetpoint2](#) (char sel, double setpt)  
*Sets new setpoint (without using config file). Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from FPGA.*
- int [getSetpoint](#) (char sel, double \*setpt)  
*Gets setpoint in [um]. Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from CONFIG\_FILE.*
- int [getSetpoint2](#) (char sel, double \*setpt)  
*Gets new setpoint (without using config file). Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from FPGA.*
- int [IIRcontrollerSetParam](#) (char ctrlr\_sel, char param\_is\_a, char param\_sel, double param)  
*Sets IIR controller parameter.*
- int [IIRcontrollerCommitParams](#) (char ctrlr\_sel)  
*Commits IIR controller parameters.*
- int [IIRcontrollerGetParam](#) (char ctrlr\_sel, char param\_is\_a, char param\_sel, double \*param)  
*Reads IIR controller parameter.*
- int [funcGenConfig](#) (float freq, float amp[4], float offset[4])  
*Configures function generator.*
- int [funcGenEnable](#) (char enable)  
*Enables or disables function generator.*
- int [getSFPinfo](#) (int selector, uint16\_t \*id, uint32\_t \*timestamp, uint32\_t \*fw\_ver, uint32\_t \*ser\_nr)  
*Gets SFP info.*
- int [getSystemVersion](#) (uint32\_t \*hwVer, uint32\_t \*swVer, uint32\_t \*ctrlVer)  
*Gets System HW, SW and PID versions.*

## Variables

- double \* [in](#)
- fftw\_complex \* [out](#)
- fftw\_plan [p](#)
- int [fd\\_DMA](#)
- int [fd\\_Mbox](#)
- int [fd\\_Lock](#)
- int [SAMP\\_FREQ](#) = 1000
- int [loginLevel](#) = 0
- char [LIB\\_BEST\\_VERSION](#) [] = [VERSION\\_STRING](#)(FROM\_DEFS\_VERSION)

## 7.1.1 Macro Definition Documentation

### 7.1.1.1 debug\_print

```
#define debug_print(  
    fmt,  
    ...)
```

**Value:**

```
do { if (LIB_BEST_DEBUG) fprintf(stderr, "[libbest] " fmt, __VA_ARGS__); } while (0)
```

### 7.1.1.2 LIB\_BEST\_DEBUG

```
#define LIB_BEST_DEBUG 0
```

### 7.1.1.3 VERSION\_STRING

```
#define VERSION_STRING(  
    X)
```

**Value:**

```
VERSION_STRING_HELPER(X)
```

### 7.1.1.4 VERSION\_STRING\_HELPER

```
#define VERSION_STRING_HELPER(  
    X)
```

**Value:**

```
#X
```

## 7.1.2 Function Documentation

### 7.1.2.1 \_\_attribute\_\_ () [1/2]

```
void __attribute__ (  
    (constructor) )
```

### 7.1.2.2 \_\_attribute\_\_ () [2/2]

```
void __attribute__ (  
    (destructor) )
```

### 7.1.2.3 getBestLock()

```
int getBestLock (  
    int fd)
```

!!! All sets should check loginLevel

### 7.1.2.4 lock\_sigaction()

```
void lock_sigaction (  
    int sig,  
    siginfo_t * siginfo,  
    void * context)
```

### 7.1.2.5 releaseBestLock()

```
int releaseBestLock (  
    int fd)
```

## 7.1.3 Variable Documentation

### 7.1.3.1 fd\_DMA

```
int fd_DMA
```

### 7.1.3.2 fd\_Lock

```
int fd_Lock
```

### 7.1.3.3 fd\_Mbox

```
int fd_Mbox
```

### 7.1.3.4 in

```
double* in
```

### 7.1.3.5 LIB\_BEST\_VERSION

```
char LIB_BEST_VERSION[] = VERSION\_STRING(FROM_DEFS_VERSION)
```

### 7.1.3.6 loginLevel

```
int loginLevel = 0
```

### 7.1.3.7 out

```
fftw_complex* out
```

### 7.1.3.8 p

```
fftw_plan p
```

### 7.1.3.9 SAMP\_FREQ

```
int SAMP_FREQ = 1000
```

## 7.2 best\_c\_interface.h File Reference

```
#include <stdint.h>
```

### Macros

- #define [BEST\\_FILE\\_MAILBOX](#) "/dev/best\_mailbox"
- #define [BEST\\_FILE\\_DISP\\_DMA](#) "/dev/best\_dma\_displ"
- #define [LOCK\\_FILE](#) "/var/lock/best\_lock"
- #define [BEST\\_FILE\\_CONFIG\\_INI](#) "/opt/CAENels/BEST/config.ini"
- #define [FFT\\_LEN](#) (2048)
- #define [LEN\\_BBF](#) (32)

### Enumerations

- enum [best\\_pid\\_select](#) { [BEST\\_PID\\_SELECT\\_X](#) = 0 , [BEST\\_PID\\_SELECT\\_Y](#) = 1 , [BEST\\_PID\\_SELECT\\_I0](#) = 2 }

### Functions

- struct [\\_\\_attribute\\_\\_](#) ((packed)) [best\\_buffer\\_line](#)
- int [getBuffer](#) (struct [best\\_buffer\\_line](#)[], int length)  
*Gets arrays of "samples" from display DMA.*
- int [getPosArray\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of postions and intensity from display DMA.*
- int [getPosScalar\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of postions and intensity from display DMA.*
- int [getPosArray](#) (double \*posX, double \*posY, double \*I0, unsigned int length)  
*Same as getPosArray\_sel, but only for 1st TetrAMM.*
- int [getVoltageArray](#) (double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages from display DMA.*
- int [getVoltageArray\\_sel](#) (int sel, double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).*

- int [getVoltageScalar\\_sel](#) (int sel, double \*ch0, double \*ch1, double \*ch2, double \*ch3, unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for the time beign).*
- int [getCurrentArray](#) (double(\*currents)[4], unsigned int length)  
*Gets arrays of currents from display DMA.*
- int [getCurrentArray\\_sel](#) (int sel, double(\*currents)[4], unsigned int length)  
*Gets arrays of currents one of two TetrAMMs.*
- int [getFFT](#) (unsigned int selector, double \*outM, double \*outF, int removeDC)  
*Calculates FFT.*
- int [getPosStats](#) (int selector, double \*meanX, double \*stdX, double \*stdX\_filt, double \*meanY, double \*stdY, double \*stdY\_filt, double \*meanI0, double \*stdI0, double \*stdI0\_filt)  
*Get position statistics.*
- int [setDisplayFreq](#) (int freq, uint8\_t use\_filter)  
*Sets display frequency.*
- int [getDisplayFreq](#) (int \*freq, uint8\_t \*use\_filter)  
*Gets display frequency.*
- struct [\\_\\_attribute\\_\\_\(\(packed\)\) best\\_stats](#)
- int [getBPMStats](#) (best\_stats \*stats)  
*Gets BPM Statistics.*
- int [getBPMscaling\\_sel](#) (int selector, double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int [getBPMscaling](#) (double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int [getROCandROI](#) (double \*roiX, double \*roiY, double \*roi0min, double \*roi0max, double \*roc)  
*Gets RoiX, RoiY, Roi0max, Roi0min and Roc of 1st BPM. Read values from CONFIG\_FILE.*
- int [getROCGenable](#) (bool \*rocX, bool \*rocY, bool \*rocl0, bool \*rocX2, bool \*rocY2, bool \*rocl02)  
*Gets enables for different ROC checks of 1st nad 2nd BPMs. Read values from FPGA.*
- int [setROCGenable](#) (bool rocX, bool rocY, bool rocl0, bool rocX2, bool rocY2, bool rocl02)  
*Sets enables for different ROC checks of 1st nad 2nd BPMs. Writes values to FPGA.*
- int [setOffsetSingle](#) (enum [best\\_pid\\_select](#) pid\_sel, double offset)  
*Sets output offset in [V] (output=this+PID value, updated even when PID is not running). Write offset to FPGA.*
- int [getOffsetSingle](#) (enum [best\\_pid\\_select](#) pid\_sel, double \*offset)  
*Gets output offset in [V]. Read offset from FPGA.*
- int [setOffset](#) (double offsetX, double offsetY, double offsetI0)  
*Sets output offsets in [V] (output=this+PID value, updated even when PID is not running).*
- int [setWindAvg](#) (enum [best\\_pid\\_select](#) pid\_sel, uint32\_t nr\_samp)  
*Sets number of samples for window averaging.*
- int [getWindAvg](#) (enum [best\\_pid\\_select](#) pid\_sel, uint32\_t \*nr\_samp)  
*Gets number of samples for window averaging. Read number of samples from FPGA.*
- int [setPIDparams](#) (char pid\_sel, double Kp, double Ki, double Kd, double emin, double lmax, double Omin, double Omax, double Ogain)  
*Sets PID parameters (legacy).*
- int [setPIDparamSingle](#) (char pid\_sel, int pid\_par, double param\_value)  
*Sets single PID parameter of specific PID. Write PID parameter to FPGA.*
- int [getPIDparamSingle](#) (char pid\_sel, int pid\_par, double \*param\_value)  
*Gets PID parameters. Read PID parameters from FPGA.*
- int [setPIDOutSel](#) (uint8\_t posX, uint8\_t posY, uint8\_t I0)  
*Sets PID output cross switch. Selects which PID is assigned to which PreDAC output channel. Write PID output cross switch to FPGA.*
- int [getPIDOutSel](#) (uint8\_t \*posX, uint8\_t \*posY, uint8\_t \*I0)  
*Gets PID output cross switch. Read PID output cross switch from FPGA.*
- int [setCrossBar](#) (uint8\_t sel, uint8\_t ch0, uint8\_t ch1, uint8\_t ch2, uint8\_t ch3)

- Sets input cross switch.*

  - int [setBPMorient](#) (uint8\_t sel, uint8\_t is90deg)
    - Sets BPM orientation for diffOverSum.*
  - int [setOutputMux](#) (uint8\_t sel)
    - Sets value of output mux.*
  - int [getOutputMux](#) (uint8\_t \*sel)
    - Gets value of output mux.*
  - int [setBPMscale](#) (double scaleX, double scaleY, int bpm)
    - Sets BPM scaling parameters for position X and Y in [um]. Write BPM scaling parameters to FPGA.*
  - int [getBPMscale](#) (int selector, double \*scaleX, double \*scaleY)
    - Gets BPM scaling parameters for position X and Y in [um]. Read BPM scaling parameters from FPGA.*
  - int [setBPMoffset](#) (double offsetX, double offsetY, int bpm)
    - Sets BPM offset in [um] ONLY FOR TETRAMMs. Write BPM offsets to FPGA.*
  - int [setBPMoffsetEnbox](#) (double offsetX, double offsetY, int bpm, int encType)
    - Sets BPM offset in [um] ONLY FOR Enboxes. Write BPM offsets to FPGA. (>= 1.8-16).*
  - int [setBPMdevice](#) (uint8\_t sel, uint8\_t isEnbox)
    - Sets BPM device for diffOverSum. Write to FPGA. (>= 1.8-18).*
  - int [setEnboxType](#) (uint8\_t sel, uint8\_t encType)
    - Sets the EnBOX type (relative or absolute) . Write to FPGA. (>= 1.8-18).*
  - int [setEnboxSumDiff](#) (uint8\_t sel, uint8\_t isSumDiff)
    - Set diffOverSum calculations (for EnBOX). 0:Normal the positions are calculated from enc1 and enc2. 1:SumDiff the positions are calculated from enc1+enc2 and enc1-enc2. Write to FPGA. (>= 1.8-18).*
  - int [getBPMoffset](#) (double \*offsetX, double \*offsetY, int bpm)
    - Gets BPM offsets in [um] ONLY FOR TETRAMMs. Read BPM offsets from FPGA.*
  - int [getBPMoffsetEnbox](#) (double \*offsetX, double \*offsetY, int bpm, int encType)
    - Gets BPM offsets in [um] ONLY FOR Enboxes. Read BPM offsets from FPGA.*
  - int [getBPMorient](#) (uint8\_t sel, uint8\_t \*is90deg)
    - Sets BPM orientation for diffOverSum. Read BPM orientation from FPGA.*
  - int [getBPMselector](#) (int \*posX, int \*posY, int \*I0)
    - Gets BPM sources. Read BPM sources from FPGA.*
  - int [setBPMselector](#) (int posX, int posY, int I0)
    - Sets BPM sources. Write BPM sources to FPGA.*
  - int [getCrossBar](#) (uint8\_t sel, uint8\_t \*ch0, uint8\_t \*ch1, uint8\_t \*ch2, uint8\_t \*ch3)
    - Gets input cross switch. Read input cross switch from FPGA.*
  - int [setROClimits](#) (int sel, double min, double max)
    - Sets ROC min and max values. ROC should be in [um] for positions and in [A] for intensity. Write min and max ROC levels of specific axis to FPGA.*
  - int [getROClimits](#) (int sel, double \*min, double \*max)
    - Gets ROC. ROC values are in [um] for positions and in [A] for intensity. Read min and max ROC levels of specific axis from FPGA.*
  - int [getLock](#) (const char \*usr, const char \*pass)
    - Acquires lock on /var/lock/best\_lock.*
  - int [getLockStatus](#) ()
    - Gets login level, which can be changed with call on [getLock\(\)](#).*
  - int [getPIDstatus](#) (void)
    - Gets PID status.*
  - int [setFBenable](#) (int enable)
    - Enables or disables feedback.*
  - int [setCtrlReset](#) ()
    - Resets internal states of controller.*
  - int [setPIDconf](#) (char conf)

- Sets PID controller configuration. Write PID controller configuration to FPGA.*

  - int [getPIDconf](#) (char \*conf)

*Sets PID controller configuration. Read PID configuration from FPGA.*

  - int [setSetpoint](#) (char sel, double setpt)

*Sets new setpoint in [um]. Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from CONFIG\_FILE.*

  - int [getSetpoint](#) (char sel, double \*setpt)

*Gets setpoint in [um]. Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from CONFIG\_FILE.*

  - int [setSetpoint2](#) (char sel, double setpt)

*Sets new setpoint (without using config file). Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from FPGA.*

  - int [getSetpoint2](#) (char sel, double \*setpt)

*Gets new setpoint (without using config file). Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from FPGA.*

  - int [IIRcontrollerSetParam](#) (char ctrlr\_sel, char param\_is\_a, char param\_sel, double param)

*Sets IIR controller parameter.*

  - int [IIRcontrollerCommitParams](#) (char ctrlr\_sel)

*Commits IIR controller parameters.*

  - int [IIRcontrollerGetParam](#) (char ctrlr\_sel, char param\_is\_a, char param\_sel, double \*param)

*Reads IIR controller parameter.*

  - int [funcGenConfig](#) (float freq, float ampl[4], float offset[4])

*Configures function generator.*

  - int [funcGenEnable](#) (char enable)

*Enables or disables function generator.*

  - int [tetrammSetRange](#) (int range, unsigned char selector)

*Sets front-end range.*

  - int [tetrammHVSetVoltage](#) (float voltage, unsigned char sel)

*Sets TetrAMM HV voltage.*

  - int [tetrammHVenable](#) (int enable, unsigned char sel)

*Enables or disables TetrAMM HV module.*

  - unsigned int [getTetrammsNumber](#) ()

*Gets number of TetrAMMs.*

  - int [getTetramms](#) (int \*pos\_A, int \*pos\_B)

*Gets TetrAMMs. (>= 1.8-14).*

  - int [enboxEncoderSelect](#) (int enc\_type, unsigned char selector)
  - unsigned int [getEnboxesNumber](#) ()

*Gets number of Enbox. (>= 1.8-15).*

  - int [getEnboxes](#) (int \*pos\_A, int \*pos\_B)

*Gets Enbox. (>= 1.8-15).*

  - int [getSFPinfo](#) (int selector, uint16\_t \*id, uint32\_t \*timestamp, uint32\_t \*fw\_ver, uint32\_t \*ser\_nr)

*Gets SFP info.*

  - int [getSystemVersion](#) (uint32\_t \*hwVer, uint32\_t \*swVer, uint32\_t \*ctrlVer)

*Gets System HW, SW and PID versions.*

## Variables

- char [LIB\\_BEST\\_VERSION](#) []

## 7.2.1 Macro Definition Documentation

### 7.2.1.1 BEST\_FILE\_CONFIG\_INI

```
#define BEST_FILE_CONFIG_INI "/opt/CAENels/BEST/config.ini"
```

### 7.2.1.2 BEST\_FILE\_DISP\_DMA

```
#define BEST_FILE_DISP_DMA "/dev/best_dma_displ"
```

### 7.2.1.3 BEST\_FILE\_MAILBOX

```
#define BEST_FILE_MAILBOX "/dev/best_mailbox"
```

### 7.2.1.4 FFT\_LEN

```
#define FFT_LEN (2048)
```

### 7.2.1.5 LEN\_BBF

```
#define LEN_BBF (32)
```

### 7.2.1.6 LOCK\_FILE

```
#define LOCK_FILE "/var/lock/best_lock"
```

## 7.2.2 Variable Documentation

### 7.2.2.1 LIB\_BEST\_VERSION

```
char LIB_BEST_VERSION[] [extern]
```

## 7.3 best\_c\_interface.h

[Go to the documentation of this file.](#)

```

00001
00002 #ifndef BEST_H_
00003 #define BEST_H_
00004
00005 #include <stdint.h>
00006
00007
00008 #define BEST_FILE_MAILBOX      "/dev/best_mailbox"
00009 #define BEST_FILE_DISP_DMA     "/dev/best_dma_displ"
00010 #define LOCK_FILE              "/var/lock/best_lock"
00011 #define BEST_FILE_CONFIG_INI   "/opt/CAENels/BEST/config.ini"
00012
00013 #define FFT_LEN                (2048)
00014 #define LEN_BBF                (32)          // sizeof(best_buffer_line)/sizeof(double)
00015
00016 #ifdef __cplusplus
00017 extern "C" {
00018 #endif /* __cplusplus */
00019
00020 extern char LIB_BEST_VERSION[];
00021
00022
00023 //=====//
00024 //                                     //
00025 //                                     Data functions                                     //
00026 //                                     //
00027 //=====//
00028
00033
00035
00038 struct __attribute__((packed)) best_buffer_line { //addresses with double pointer *(double)
00039     double bpm0_posX;          // 0
00040     double bpm0_posY;          // 1
00041     double bpm0_i0;           // 2
00042     double bpm0_rsvd;         // 3
00043     double bpml_posX;         // 4
00044     double bpml_posY;         // 5
00045     double bpml_i0;           // 6
00046     uint32_t sin;             // 7
00047     uint32_t cos;             // 7
00048     double dac[4];            // 8
00049     uint32_t rsvd0[8];        // 8+4=12
00050     uint32_t rsvd1[4];        // 12+8/2=16
00051     float setpts[3];          // 16+4/2=18
00052     uint32_t status;          // 19
00053     uint32_t rsvd2[7];        // 18+(3+1)/2=20
00054     uint32_t timestamp;        // 23
00055     double tetramm0[4];        // 20+(7+1)/2=24
00056     double tetramm1[4];        // 24+4=28
00057 };
00058
00063 int getBuffer(struct best_buffer_line[], int length);
00064
00073 int getPosArray_sel(
00074     int selector,
00075     double* posX,
00076     double* posY,
00077     double* I0,
00078     unsigned int length
00079 );
00080
00089 int getPosScalar_sel(
00090     int selector,
00091     double* posX,
00092     double* posY,
00093     double* I0,
00094     unsigned int length
00095 );
00096
00099 int getPosArray(double* posX, double* posY, double* I0, unsigned int length);
00100
00101
00106 int getVoltageArray(double (*voltages)[4], unsigned int length);
00107
00115 int getVoltageArray_sel(int sel, double (*voltages)[4], unsigned int length);
00116
00124 int getVoltageScalar_sel(int sel, double* ch0, double* ch1, double* ch2, double* ch3, unsigned int
length);
00125
00130 int getCurrentArray(double (*currents)[4], unsigned int length);
00131
00138 int getCurrentArray_sel(int sel, double (*currents)[4], unsigned int length);

```

```

00139
00140
00158 int getFFT(unsigned int selector, double *outM, double *outF, int removedDC);
00159
00160
00167 int getPosStats(int selector,
00168                 double* meanX, double* stdX, double* stdX_filt,
00169                 double* meanY, double* stdY, double* stdY_filt,
00170                 double* meanI0, double* stdI0, double* stdI0_filt);
00171
00178 int setDisplayFreq(int freq, uint8_t use_filter);
00179
00183 int getDisplayFreq(int* freq, uint8_t* use_filter);
00184
00186
00189 struct __attribute__((__packed__)) best_stats {
00190     double filt_freq;
00191     struct __attribute__((__packed__)) {
00192         double mean;
00193         double stdev;
00194         double stdev_lim;
00195     } stat[6];
00196 };
00197
00202 int getBPMStats(best_stats* stats);
00203
00204 // end of dataFunc
00206
00207
00208 //=====//
00209 //                                                     //
00210 //                         Configuration functions           //
00211 //                                                     //
00212 //=====//
00213
00218
00219
00228 int getBPMscaling_sel(
00229     int selector,
00230     double* scaleX,
00231     double* scaleY
00232 );
00233
00240 int getBPMscaling(double* scaleX, double* scaleY);
00241
00242
00252 int getROCanROI( double *roiX,
00253                 double *roiY,
00254                 double *roiI0min,
00255                 double *roiI0max,
00256                 double *roc );
00257
00269 int getROCanable(bool *rocX,
00270                 bool *rocY,
00271                 bool *rocI0,
00272                 bool *rocX2,
00273                 bool *rocY2,
00274                 bool *rocI02);
00275
00287 int setROCanable(bool rocX,
00288                 bool rocY,
00289                 bool rocI0,
00290                 bool rocX2,
00291                 bool rocY2,
00292                 bool rocI02);
00293
00294 //
00295 enum best_pid_select {
00296     BEST_PID_SELECT_X = 0,
00297     BEST_PID_SELECT_Y = 1,
00298     BEST_PID_SELECT_I0 = 2
00299 };
00300
00309 int setOffsetSingle(enum best_pid_select pid_sel, double offset);
00310
00319 int getOffsetSingle(enum best_pid_select pid_sel, double* offset);
00320
00329 int setOffset(double offsetX, double offsetY, double offsetI0);
00330
00337 int setWindAvg(enum best_pid_select pid_sel, uint32_t nr_samp);
00338
00345 int getWindAvg(enum best_pid_select pid_sel, uint32_t* nr_samp);
00346
00360 int setPIDparams( char pid_sel,
00361                 double Kp,
00362                 double Ki,
00363                 double Kd,

```

```

00364         double emin,
00365         double Imax,
00366         double Omin,
00367         double Omax,
00368         double Ogain);
00369
00378 int setPIDparamSingle(char pid_sel, int pid_par, double param_value);
00379
00388 int getPIDparamSingle(char pid_sel, int pid_par, double* param_value);
00389
00398 int setPIDoutSel(uint8_t posX, uint8_t posY, uint8_t IO);
00399
00408 int getPIDoutSel(uint8_t *posX, uint8_t *posY, uint8_t *IO);
00409
00419 int setCrossBar(uint8_t sel, uint8_t ch0, uint8_t ch1, uint8_t ch2, uint8_t ch3);
00420
00427 int setBPMorient(uint8_t sel, uint8_t is90deg);
00428
00434 int setOutputMux(uint8_t sel);
00435
00441 int getOutputMux(uint8_t* sel);
00442
00451 int setBPMscale(double scaleX, double scaleY, int bpm);
00452
00459 int getBPMscale(int selector, double* scaleX, double* scaleY);
00460
00469 int setBPMoffset(double offsetX, double offsetY, int bpm);
00470
00480 int setBPMoffsetEnbox(double offsetX, double offsetY, int bpm, int encType);
00481
00489 int setBPMdevice(uint8_t sel, uint8_t isEnbox);
00490
00498 int setEnboxType(uint8_t sel, uint8_t encType);
00499
00509 int setEnboxSumDiff(uint8_t sel, uint8_t isSumDiff);
00510
00518 int getBPMoffset(double *offsetX, double *offsetY, int bpm);
00519
00528 int getBPMoffsetEnbox(double *offsetX, double *offsetY, int bpm, int encType);
00529
00537 int getBPMorient(uint8_t sel, uint8_t* is90deg);
00538
00547 int getBPMselector(int *posX, int *posY, int *IO);
00548
00557 int setBPMselector(int posX, int posY, int IO);
00558
00568 int getCrossBar(uint8_t sel, uint8_t* ch0, uint8_t* ch1, uint8_t* ch2, uint8_t* ch3);
00569
00579 int setROClimits(int sel, double min, double max);
00580
00589 int getROClimits(int sel, double* min, double* max);
00590 // end of configFunc
00592
00593
00594
00595 //=====//
00596 //                                     //
00597 //                                     Access control functions //
00598 //                                     //
00599 //=====//
00600
00605
00612 int getLock(const char *usr, const char *pass);
00613
00618 int getLockStatus();
00619 // end of loginFunc
00621
00622
00623 //=====//
00624 //                                     //
00625 //                                     PID control functions //
00626 //                                     //
00627 //=====//
00628
00633
00638 int getPIDstatus(void);
00639
00644 int setFBenable(int enable);
00645
00650 int setCtrlReset();
00651
00658 int setPIDconf(char conf);
00659
00665 int getPIDconf(char* conf);
00666
00676 int setSetpoint(char sel, double setpt);
00677

```

```

00687 int getSetpoint(char sel, double* setpt);
00688
00698 int setSetpoint2(char sel, double setpt);
00699
00708 int getSetpoint2(char sel, double* setpt);
00709
00721 int IIRcontrollerSetParam(char ctrlr_sel, char param_is_a, char param_sel, double param);
00722
00727 int IIRcontrollerCommitParams(char ctrlr_sel);
00728
00739 int IIRcontrollerGetParam(char ctrlr_sel, char param_is_a, char param_sel, double* param);
00740 // end of PIDfunc
00742
00743
00744 //=====//
00745 //                                     //
00746 //                               Function gen control functions //
00747 //                                     //
00748 //=====//
00749
00753
00754
00764 int funcGenConfig(float freq, float ampl[4], float offset[4]);
00765
00769 int funcGenEnable(char enable);
00770
00771 // end of FuncGenfunc
00773
00774
00775 //=====//
00776 //                                     //
00777 //                               TetrAMM functions //
00778 //                                     //
00779 //=====//
00780
00784
00790 int tetrammSetRange(int range, unsigned char selector);
00791
00792
00798 int tetrammHVSetVoltage(float voltage, unsigned char sel);
00799
00800
00806 int tetrammHVenable(int enable, unsigned char sel);
00807
00811 unsigned int getTetrammsNumber();
00812
00818 int getTetramms(int* pos_A, int* pos_B);
00819 // end of Tetrammfunc
00821
00822
00823 //=====//
00824 //                                     //
00825 //                               EnBOX functions //
00826 //                                     //
00827 //=====//
00828
00832
00834 // \brief EnBOX Select ENCODER.
00838 int enboxEncoderSelect(int enc_type, unsigned char selector);
00839
00843 unsigned int getEnboxesNumber();
00844
00850 int getEnboxes(int* pos_A, int* pos_B);
00851
00852
00853 // end of EnBOXfunc
00855
00856 //=====//
00857 //                                     //
00858 //                               Misc functions //
00859 //                                     //
00860 //=====//
00861
00865
00875 int getSFPinfo(int selector,
00876                uint16_t *id,
00877                uint32_t *timestamp,
00878                uint32_t *fw_ver,
00879                uint32_t *ser_nr);
00880
00888 int getSystemVersion(uint32_t* hwVer, uint32_t* swVer, uint32_t* ctrlVer);
00889
00890 // end of Miscfunc
00892
00893 #ifndef __cplusplus
00894 }
00895 #endif /* __cplusplus */

```

```
00896
00897 #endif /* BEST_H_ */
```

## 7.4 best\_c\_interface\_conf.cpp File Reference

```
#include "best_c_interface.h"
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include "pcie_driver/BEST_PCIE.h"
#include "pcie_mailbox/mailbox_comm_defs.h"
#include <iostream>
#include "inih/cpp/INIReader.h"
#include <complex>
#include <fftw3.h>
#include <signal.h>
```

### Functions

- int [getPIDstatus](#) (void)  
*Gets PID status.*
- int [getPosStats](#) (int selector, double \*meanX, double \*stdX, double \*stdX\_filt, double \*meanY, double \*stdY, double \*stdY\_filt, double \*meanI0, double \*stdI0, double \*stdI0\_filt)  
*Get position statistics.*
- int [getBPMscaling\\_sel](#) (int selector, double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int [getBPMscaling](#) (double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG\_FILE.*
- int [getBPMscale](#) (int selector, double \*scaleX, double \*scaleY)  
*Gets BPM scaling parameters for position X and Y in [um]. Read BPM scaling parameters from FPGA.*
- int [getROCandROI](#) (double \*roiX, double \*roiY, double \*roiI0min, double \*roiI0max, double \*roc)  
*Gets RoiX, RoiY, RoiI0max, RoiI0min and Roc of 1st BPM. Read values from CONFIG\_FILE.*
- int [getROCEnable](#) (bool \*rocX, bool \*rocY, bool \*rocl0, bool \*rocX2, bool \*rocY2, bool \*rocl02)  
*Gets enables for different ROC checks of 1st nad 2nd BPMs. Read values from FPGA.*
- int [setROCEnable](#) (bool rocX, bool rocY, bool rocl0, bool rocX2, bool rocY2, bool rocl02)  
*Sets enables for different ROC checks of 1st nad 2nd BPMs. Writes values to FPGA.*
- int [setOffsetSingle](#) (enum [best\\_pid\\_select](#) pid\_sel, double offset)  
*Sets output offset in [V] (output=this+PID value, updated even when PID is not running). Write offset to FPGA.*
- int [getOffsetSingle](#) (enum [best\\_pid\\_select](#) pid\_sel, double \*offset)  
*Gets output offset in [V]. Read offset from FPGA.*
- int [setOffset](#) (double offsetX, double offsetY, double offsetI0)  
*Sets output offsets in [V] (output=this+PID value, updated even when PID is not running).*
- int [setWindAvg](#) (enum [best\\_pid\\_select](#) pid\_sel, uint32\_t nr\_samp)  
*Sets number of samples for window averaging.*

- int [getWindAvg](#) (enum [best\\_pid\\_select](#) pid\_sel, uint32\_t \*nr\_samp)  
*Gets number of samples for window averaging. Read number of samples from FPGA.*
- int [setPIDparams](#) (char pid\_sel, double Kp, double Ki, double Kd, double emin, double lmax, double Omin, double Omax, double Ogain)  
*Sets PID parameters (legacy).*
- int [setPIDparamSingle](#) (char pid\_sel, int pid\_par, double param\_value)  
*Sets single PID parameter of specific PID. Write PID parameter to FPGA.*
- int [getPIDparamSingle](#) (char pid\_sel, int pid\_par, double \*param\_value)  
*Gets PID parameters. Read PID parameters from FPGA.*
- int [setPIDOutSel](#) (uint8\_t posX, uint8\_t posY, uint8\_t I0)  
*Sets PID output cross switch. Selects which PID is assigned to which PreDAC output channel. Write PID output cross switch to FPGA.*
- int [getPIDOutSel](#) (uint8\_t \*posX, uint8\_t \*posY, uint8\_t \*I0)  
*Gets PID output cross switch. Read PID output cross switch from FPGA.*
- int [setCrossBar](#) (uint8\_t sel, uint8\_t ch0, uint8\_t ch1, uint8\_t ch2, uint8\_t ch3)  
*Sets input cross switch.*
- int [setBPMorient](#) (uint8\_t sel, uint8\_t is90deg)  
*Sets BPM orientation for diffOverSum.*
- int [getBPMorient](#) (uint8\_t sel, uint8\_t \*is90deg)  
*Sets BPM orientation for diffOverSum. Read BPM orientation from FPGA.*
- int [setOutputMux](#) (uint8\_t sel)  
*Sets value of output mux.*
- int [getOutputMux](#) (uint8\_t \*sel)  
*Gets value of output mux.*
- int [setBPMscale](#) (double scaleX, double scaleY, int bpm)  
*Sets BPM scaling parameters for position X and Y in [um]. Write BPM scaling parameters to FPGA.*
- int [setBPMoffset](#) (double offsetX, double offsetY, int bpm)  
*Sets BPM offset in [um] ONLY FOR TETRAMMs. Write BPM offsets to FPGA.*
- int [setBPMoffsetEnbox](#) (double offsetX, double offsetY, int bpm, int encType)  
*Sets BPM offset in [um] ONLY FOR Enboxes. Write BPM offsets to FPGA. (>= 1.8-16).*
- int [setBPMdevice](#) (uint8\_t sel, uint8\_t isEnbox)  
*Sets BPM device for diffOverSum. Write to FPGA. (>= 1.8-18).*
- int [setEnboxType](#) (uint8\_t sel, uint8\_t encType)  
*Sets the EnBOX type (relative or absolute) . Write to FPGA. (>= 1.8-18).*
- int [setEnboxSumDiff](#) (uint8\_t sel, uint8\_t isSumDiff)  
*Set diffOverSum calculations (for EnBOX). 0:Normal the positions are calculated from enc1 and enc2. 1:SumDiff the positions are calculated from enc1+enc2 and enc1-enc2. Write to FPGA. (>= 1.8-18).*
- int [getBPMoffset](#) (double \*offsetX, double \*offsetY, int bpm)  
*Gets BPM offsets in [um] ONLY FOR TETRAMMs. Read BPM offsets from FPGA.*
- int [getBPMoffsetEnbox](#) (double \*offsetX, double \*offsetY, int bpm, int encType)  
*Gets BPM offsets in [um] ONLY FOR Enboxes. Read BPM offsets from FPGA.*
- int [getBPMselector](#) (int \*posX, int \*posY, int \*I0)  
*Gets BPM sources. Read BPM sources from FPGA.*
- int [setBPMselector](#) (int posX, int posY, int I0)  
*Sets BPM sources. Write BPM sources to FPGA.*
- int [getCrossBar](#) (uint8\_t sel, uint8\_t \*ch0, uint8\_t \*ch1, uint8\_t \*ch2, uint8\_t \*ch3)  
*Gets input cross switch. Read input cross switch from FPGA.*
- int [setROClimits](#) (int sel, double min, double max)  
*Sets ROC min and max values. ROC should be in [um] for positions and in [A] for intensity. Write min and max ROC levels of specific axis to FPGA.*
- int [getROClimits](#) (int sel, double \*min, double \*max)  
*Gets ROC. ROC values are in [um] for positions and in [A] for intensity. Read min and max ROC levels of specific axis from FPGA.*

## Variables

- int [loginLevel](#)
- int [fd\\_Mbox](#)
- int [fd\\_DMA](#)
- double \* [in](#)
- fftw\_complex \* [out](#)
- fftw\_plan [p](#)
- int [SAMP\\_FREQ](#)

## 7.4.1 Variable Documentation

### 7.4.1.1 fd\_DMA

```
int fd_DMA [extern]
```

### 7.4.1.2 fd\_Mbox

```
int fd_Mbox [extern]
```

### 7.4.1.3 in

```
double* in [extern]
```

### 7.4.1.4 loginLevel

```
int loginLevel [extern]
```

### 7.4.1.5 out

```
fftw_complex* out [extern]
```

### 7.4.1.6 p

```
fftw_plan p [extern]
```

### 7.4.1.7 SAMP\_FREQ

```
int SAMP_FREQ [extern]
```

## 7.5 best\_c\_interface\_data.cpp File Reference

```
#include "best_c_interface.h"
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include "pcie_driver/BEST_PCIE.h"
#include "pcie_mailbox/mailbox_comm_defs.h"
#include <iostream>
#include "inih/cpp/INIReader.h"
#include <complex>
#include <fftw3.h>
#include <signal.h>
```

### Functions

- int [getBuffer](#) (struct [best\\_buffer\\_line](#) buf[], int length)  
*Gets arrays of "samples" from display DMA.*
- int [getPosArray\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of postions and intensity from display DMA.*
- int [getPosScalar\\_sel](#) (int selector, double \*posX, double \*posY, double \*I0, unsigned int length)  
*Gets arrays of postions and intensity from display DMA.*
- int [getPosArray](#) (double \*posX, double \*posY, double \*I0, unsigned int length)  
*Same as getPosArray\_sel, but only for 1st TetrAMM.*
- int [getVoltageArray\\_sel](#) (int sel, double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).*
- int [getVoltageArray](#) (double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages from display DMA.*
- int [getVoltageScalar\\_sel](#) (int sel, double \*ch0, double \*ch1, double \*ch2, double \*ch3, unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).*
- int [getCurrentArray\\_sel](#) (int sel, double(\*currents)[4], unsigned int length)  
*Gets arrays of currents one of two TetrAMMs.*
- int [getCurrentArray](#) (double(\*currents)[4], unsigned int length)  
*Gets arrays of currents from display DMA.*
- double [fftw\\_abs](#) (fftw\_complex f)
- int [getFFT](#) (unsigned int selector, double \*outM, double \*outF, int removedDC)  
*Calculates FFT.*
- int [setDisplayFreq](#) (int freq, uint8\_t use\_filter)  
*Sets display frequency.*
- int [getDisplayFreq](#) (int \*freq, uint8\_t \*use\_filter)  
*Gets display frequency.*
- int [getBPMStats](#) ([best\\_stats](#) \*stats)  
*Gets BPM Statistics.*

## Variables

- int `loginLevel`
- int `fd_Mbox`
- int `fd_DMA`
- double \* `in`
- `fftw_complex` \* `out`
- `fftw_plan` `p`
- int `SAMP_FREQ`

## 7.5.1 Function Documentation

### 7.5.1.1 `fftw_abs()`

```
double fftw_abs (  
    fftw_complex f)
```

## 7.5.2 Variable Documentation

### 7.5.2.1 `fd_DMA`

```
int fd_DMA [extern]
```

### 7.5.2.2 `fd_Mbox`

```
int fd_Mbox [extern]
```

### 7.5.2.3 `in`

```
double* in [extern]
```

### 7.5.2.4 `loginLevel`

```
int loginLevel [extern]
```

### 7.5.2.5 `out`

```
fftw_complex* out [extern]
```

### 7.5.2.6 `p`

```
fftw_plan p [extern]
```

### 7.5.2.7 SAMP\_FREQ

```
int SAMP_FREQ [extern]
```

## 7.6 best\_c\_interface\_tetramm.cpp File Reference

```
#include "best_c_interface.h"  
#include <stdlib.h>  
#include <time.h>  
#include <stdint.h>  
#include <string.h>  
#include <errno.h>  
#include <unistd.h>  
#include <fcntl.h>  
#include <sys/stat.h>  
#include <sys/ioctl.h>  
#include "pcie_driver/BEST_PCIE.h"  
#include "pcie_mailbox/mailbox_comm_defs.h"  
#include <iostream>  
#include "inih/cpp/INIReader.h"  
#include <complex>  
#include <fftw3.h>  
#include <signal.h>
```

### Functions

- int [tetrammSetRange](#) (int range, unsigned char selector)  
*Sets front-end range.*
- int [tetrammHVSetVoltage](#) (float voltage, unsigned char sel)  
*Sets TetrAMM HV voltage.*
- int [tetrammHVenable](#) (int enable, unsigned char sel)  
*Enables or disables TetrAMM HV module.*
- unsigned int [getTetrammsNumber](#) ()  
*Gets number of TetrAMMs.*
- unsigned int [getEnboxesNumber](#) ()  
*Gets number of Enbox. (>= 1.8-15).*
- int [getTetramms](#) (int \*pos\_A, int \*pos\_B)  
*Gets TetrAMMs. (>= 1.8-14).*
- int [getEnboxes](#) (int \*pos\_A, int \*pos\_B)  
*Gets Enbox. (>= 1.8-15).*
- int [enboxEncoderSelect](#) (int enc\_type, unsigned char selector)

### Variables

- int [loginLevel](#)
- int [fd\\_Mbox](#)

## 7.6.1 Variable Documentation

### 7.6.1.1 fd\_Mbox

```
int fd_Mbox [extern]
```

### 7.6.1.2 loginLevel

```
int loginLevel [extern]
```

## 7.7 changelog.md File Reference

# Index

[2.0.0] - 2026/06/15, 1

\_\_attribute\_\_

best\_c\_interface.c, 51

Data functions, 10

Access control functions, 34

getLock, 34

getLockStatus, 34

best\_buffer\_line, 47

best\_c\_interface.c, 49

\_\_attribute\_\_, 51

debug\_print, 51

fd\_DMA, 52

fd\_Lock, 52

fd\_Mbox, 52

getBestLock, 51

in, 52

LIB\_BEST\_DEBUG, 51

LIB\_BEST\_VERSION, 52

lock\_sigaction, 52

loginLevel, 52

out, 52

p, 53

releaseBestLock, 52

SAMP\_FREQ, 53

VERSION\_STRING, 51

VERSION\_STRING\_HELPER, 51

best\_c\_interface.h, 53

BEST\_FILE\_CONFIG\_INI, 57

BEST\_FILE\_DISP\_DMA, 57

BEST\_FILE\_MAILBOX, 57

FFT\_LEN, 57

LEN\_BBF, 57

LIB\_BEST\_VERSION, 57

LOCK\_FILE, 57

best\_c\_interface\_conf.cpp, 62

fd\_DMA, 64

fd\_Mbox, 64

in, 64

loginLevel, 64

out, 64

p, 64

SAMP\_FREQ, 64

best\_c\_interface\_data.cpp, 65

fd\_DMA, 66

fd\_Mbox, 66

fftw\_abs, 66

in, 66

loginLevel, 66

out, 66

p, 66

SAMP\_FREQ, 66

best\_c\_interface\_tetramm.cpp, 67

fd\_Mbox, 68

loginLevel, 68

BEST\_FILE\_CONFIG\_INI

best\_c\_interface.h, 57

BEST\_FILE\_DISP\_DMA

best\_c\_interface.h, 57

BEST\_FILE\_MAILBOX

best\_c\_interface.h, 57

best\_pid\_select

Configuration functions, 17

BEST\_PID\_SELECT\_I0

Configuration functions, 17

BEST\_PID\_SELECT\_X

Configuration functions, 17

BEST\_PID\_SELECT\_Y

Configuration functions, 17

best\_stats, 47

changelog.md, 68

Configuration functions, 15

best\_pid\_select, 17

BEST\_PID\_SELECT\_I0, 17

BEST\_PID\_SELECT\_X, 17

BEST\_PID\_SELECT\_Y, 17

getBPMoffset, 17

getBPMoffsetEnbox, 17

getBPMorient, 18

getBPMscale, 18

getBPMscaling, 18

getBPMscaling\_sel, 19

getBPMselector, 19

getCrossBar, 20

getOffsetSingle, 20

getOutputMux, 21

getPIDoutSel, 21

getPIDparamSingle, 22

getROCandROI, 22

getROCanable, 23

getROClimits, 23

getWindAvg, 24

setBPMdevice, 24

setBPMoffset, 25

setBPMoffsetEnbox, 25

setBPMorient, 26

setBPMscale, 26

setBPMselector, 27

- setCrossBar, 27
- setEnboxSumDiff, 28
- setEnboxType, 28
- setOffset, 29
- setOffsetSingle, 29
- setOutputMux, 30
- setPIDoutSel, 30
- setPIDparams, 31
- setPIDparamSingle, 31
- setROCEnable, 32
- setROClimits, 32
- setWindAvg, 33
- Data functions, 9
  - \_\_attribute\_\_, 10
  - getBPMStats, 10
  - getBuffer, 10
  - getCurrentArray, 10
  - getCurrentArray\_sel, 11
  - getDisplayFreq, 11
  - getFFT, 11
  - getPosArray, 12
  - getPosArray\_sel, 12
  - getPosScalar\_sel, 12
  - getPosStats, 13
  - getVoltageArray, 13
  - getVoltageArray\_sel, 13
  - getVoltageScalar\_sel, 14
  - setDisplayFreq, 14
- debug\_print
  - best\_c\_interface.c, 51
- EnBOX functions, 43
  - enboxEncoderSelect, 43
  - getEnboxes, 43
  - getEnboxesNumber, 43
- enboxEncoderSelect
  - EnBOX functions, 43
- fd\_DMA
  - best\_c\_interface.c, 52
  - best\_c\_interface\_conf.cpp, 64
  - best\_c\_interface\_data.cpp, 66
- fd\_Lock
  - best\_c\_interface.c, 52
- fd\_Mbox
  - best\_c\_interface.c, 52
  - best\_c\_interface\_conf.cpp, 64
  - best\_c\_interface\_data.cpp, 66
  - best\_c\_interface\_tetramm.cpp, 68
- FFT\_LEN
  - best\_c\_interface.h, 57
- fftw\_abs
  - best\_c\_interface\_data.cpp, 66
- funcGenConfig
  - Function generator functions, 40
- funcGenEnable
  - Function generator functions, 40
- Function generator functions, 40
  - funcGenConfig, 40
  - funcGenEnable, 40
- getBestLock
  - best\_c\_interface.c, 51
- getBPMoffset
  - Configuration functions, 17
- getBPMoffsetEnbox
  - Configuration functions, 17
- getBPMorient
  - Configuration functions, 18
- getBPMscale
  - Configuration functions, 18
- getBPMscaling
  - Configuration functions, 18
- getBPMscaling\_sel
  - Configuration functions, 19
- getBPMselector
  - Configuration functions, 19
- getBPMStats
  - Data functions, 10
- getBuffer
  - Data functions, 10
- getCrossBar
  - Configuration functions, 20
- getCurrentArray
  - Data functions, 10
- getCurrentArray\_sel
  - Data functions, 11
- getDisplayFreq
  - Data functions, 11
- getEnboxes
  - EnBOX functions, 43
- getEnboxesNumber
  - EnBOX functions, 43
- getFFT
  - Data functions, 11
- getLock
  - Access control functions, 34
- getLockStatus
  - Access control functions, 34
- getOffsetSingle
  - Configuration functions, 20
- getOutputMux
  - Configuration functions, 21
- getPIDconf
  - PID control functions, 35
- getPIDoutSel
  - Configuration functions, 21
- getPIDparamSingle
  - Configuration functions, 22
- getPIDstatus
  - PID control functions, 36
- getPosArray
  - Data functions, 12
- getPosArray\_sel
  - Data functions, 12
- getPosScalar\_sel
  - Data functions, 12

- getPosStats
  - Data functions, [13](#)
- getROcandROI
  - Configuration functions, [22](#)
- getROcenable
  - Configuration functions, [23](#)
- getROclimits
  - Configuration functions, [23](#)
- getSetpoint
  - PID control functions, [36](#)
- getSetpoint2
  - PID control functions, [36](#)
- getSFPinfo
  - Misc, [44](#)
- getSystemVersion
  - Misc, [44](#)
- getTetramms
  - TetrAMM functions, [41](#)
- getTetrammsNumber
  - TetrAMM functions, [41](#)
- getVoltageArray
  - Data functions, [13](#)
- getVoltageArray\_sel
  - Data functions, [13](#)
- getVoltageScalar\_sel
  - Data functions, [14](#)
- getWindAvg
  - Configuration functions, [24](#)
  
- IIRcontrollerCommitParams
  - PID control functions, [37](#)
- IIRcontrollerGetParam
  - PID control functions, [37](#)
- IIRcontrollerSetParam
  - PID control functions, [37](#)
- in
  - [best\\_c\\_interface.c](#), [52](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [64](#)
  - [best\\_c\\_interface\\_data.cpp](#), [66](#)
  
- LEN\_BBF
  - [best\\_c\\_interface.h](#), [57](#)
- LIB\_BEST\_DEBUG
  - [best\\_c\\_interface.c](#), [51](#)
- LIB\_BEST\_VERSION
  - [best\\_c\\_interface.c](#), [52](#)
  - [best\\_c\\_interface.h](#), [57](#)
- LOCK\_FILE
  - [best\\_c\\_interface.h](#), [57](#)
- lock\_sigaction
  - [best\\_c\\_interface.c](#), [52](#)
- loginLevel
  - [best\\_c\\_interface.c](#), [52](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [64](#)
  - [best\\_c\\_interface\\_data.cpp](#), [66](#)
  - [best\\_c\\_interface\\_tetramm.cpp](#), [68](#)
  
- Misc, [44](#)
  - [getSFPinfo](#), [44](#)
  - [getSystemVersion](#), [44](#)
  
- out
  - [best\\_c\\_interface.c](#), [52](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [64](#)
  - [best\\_c\\_interface\\_data.cpp](#), [66](#)
  
- P
  - [best\\_c\\_interface.c](#), [53](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [64](#)
  - [best\\_c\\_interface\\_data.cpp](#), [66](#)
- PID control functions, [35](#)
  - [getPIDconf](#), [35](#)
  - [getPIDstatus](#), [36](#)
  - [getSetpoint](#), [36](#)
  - [getSetpoint2](#), [36](#)
  - [IIRcontrollerCommitParams](#), [37](#)
  - [IIRcontrollerGetParam](#), [37](#)
  - [IIRcontrollerSetParam](#), [37](#)
  - [setCtrlReset](#), [38](#)
  - [setFBenable](#), [38](#)
  - [setPIDconf](#), [38](#)
  - [setSetpoint](#), [39](#)
  - [setSetpoint2](#), [39](#)
  
- releaseBestLock
  - [best\\_c\\_interface.c](#), [52](#)
  
- SAMP\_FREQ
  - [best\\_c\\_interface.c](#), [53](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [64](#)
  - [best\\_c\\_interface\\_data.cpp](#), [66](#)
- setBPMdevice
  - Configuration functions, [24](#)
- setBPMoffset
  - Configuration functions, [25](#)
- setBPMoffsetEnbox
  - Configuration functions, [25](#)
- setBPMorient
  - Configuration functions, [26](#)
- setBPMscale
  - Configuration functions, [26](#)
- setBPMselector
  - Configuration functions, [27](#)
- setCrossBar
  - Configuration functions, [27](#)
- setCtrlReset
  - PID control functions, [38](#)
- setDisplayFreq
  - Data functions, [14](#)
- setEnboxSumDiff
  - Configuration functions, [28](#)
- setEnboxType
  - Configuration functions, [28](#)
- setFBenable
  - PID control functions, [38](#)
- setOffset
  - Configuration functions, [29](#)
- setOffsetSingle

- Configuration functions, [29](#)
- setOutputMux
  - Configuration functions, [30](#)
- setPIDconf
  - PID control functions, [38](#)
- setPIDoutSel
  - Configuration functions, [30](#)
- setPIDparams
  - Configuration functions, [31](#)
- setPIDparamSingle
  - Configuration functions, [31](#)
- setROCenable
  - Configuration functions, [32](#)
- setROClimits
  - Configuration functions, [32](#)
- setSetpoint
  - PID control functions, [39](#)
- setSetpoint2
  - PID control functions, [39](#)
- setWindAvg
  - Configuration functions, [33](#)
- TetrAMM functions, [41](#)
  - getTetramms, [41](#)
  - getTetrammsNumber, [41](#)
  - tetrammHVenable, [41](#)
  - tetrammHVSetVoltage, [42](#)
  - tetrammSetRange, [42](#)
- tetrammHVenable
  - TetrAMM functions, [41](#)
- tetrammHVSetVoltage
  - TetrAMM functions, [42](#)
- tetrammSetRange
  - TetrAMM functions, [42](#)
- VERSION\_STRING
  - best\_c\_interface.c, [51](#)
- VERSION\_STRING\_HELPER
  - best\_c\_interface.c, [51](#)