

## BIBEST Interface

Generated by Doxygen 1.8.17



<b>1 libbibest</b>	<b>1</b>
1.0.1 changelog version	1
1.1 Before building	1
1.2 Generated files	1
1.2.1 Build issues:	1
1.3 CHANGELOG	2
1.3.1 1.2.7	2
1.3.2 1.2.6	2
1.3.3 1.2.5	2
1.3.4 1.2.4	2
1.3.5 1.2.3	2
<b>2 Module Index</b>	<b>3</b>
2.1 Modules	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 Module 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()	11
5.1.2.6 getCurrentArray_sel()	11
5.1.2.7 getDisplayFreq()	12
5.1.2.8 getFFT()	12
5.1.2.9 getPosArray()	12
5.1.2.10 getPosArray_sel()	13
5.1.2.11 getPosStats()	13
5.1.2.12 getVoltageArray()	13
5.1.2.13 getVoltageArray_sel()	14
5.1.2.14 setDisplayFreq()	14
5.2 Configuration functions	16
5.2.1 Detailed Description	17
5.2.2 Enumeration Type Documentation	17
5.2.2.1 best_pid_select	17
5.2.3 Function Documentation	18

---

5.2.3.1	getBPMoffset()	18
5.2.3.2	getBPMorient()	18
5.2.3.3	getBPMscale()	19
5.2.3.4	getBPMscaling()	19
5.2.3.5	getBPMscaling_sel()	19
5.2.3.6	getBPMselector()	20
5.2.3.7	getCrossBar()	20
5.2.3.8	getOffsetSingle()	21
5.2.3.9	getOutputMux()	21
5.2.3.10	getPIDoutSel()	22
5.2.3.11	getPIDparamSingle()	22
5.2.3.12	getROCandROI()	23
5.2.3.13	getROCenable()	23
5.2.3.14	getROClimits()	24
5.2.3.15	getWindAvg()	24
5.2.3.16	setBPMoffset()	24
5.2.3.17	setBPMorient()	25
5.2.3.18	setBPMscale()	25
5.2.3.19	setBPMselector()	26
5.2.3.20	setCrossBar()	26
5.2.3.21	setOffset()	28
5.2.3.22	setOffsetSingle()	29
5.2.3.23	setOutputMux()	30
5.2.3.24	setPIDoutSel()	30
5.2.3.25	setPIDparams()	31
5.2.3.26	setPIDparamSingle()	32
5.2.3.27	setROCenable()	32
5.2.3.28	setROClimits()	33
5.2.3.29	setWindAvg()	33
5.3	Access control functions	35
5.3.1	Detailed Description	35
5.3.2	Function Documentation	35
5.3.2.1	getLock()	35
5.3.2.2	getLockStatus()	35
5.4	PID control functions	37
5.4.1	Detailed Description	37
5.4.2	Function Documentation	38
5.4.2.1	commitFiltCoef()	38
5.4.2.2	getFiltCoef()	38
5.4.2.3	getFiltControl()	38
5.4.2.4	getPIDconf()	39
5.4.2.5	getPIDstatus()	39

5.4.2.6	getSetpoint()	40
5.4.2.7	getSetpoint2()	40
5.4.2.8	IIRcontrollerCommitParams()	40
5.4.2.9	IIRcontrollerGetParam()	41
5.4.2.10	IIRcontrollerSetParam()	41
5.4.2.11	setCtrlReset()	42
5.4.2.12	setFBenable()	42
5.4.2.13	setFiltCoef()	42
5.4.2.14	setFiltControl()	43
5.4.2.15	setPIDconf()	43
5.4.2.16	setSetpoint()	44
5.4.2.17	setSetpoint2()	44
5.5	Function generator functions	46
5.5.1	Detailed Description	46
5.5.2	Function Documentation	46
5.5.2.1	funcGenConfig()	46
5.5.2.2	funcGenEnable()	46
5.6	TetrAMM functions	47
5.6.1	Detailed Description	47
5.6.2	Function Documentation	47
5.6.2.1	getTetramms()	47
5.6.2.2	getTetrammsNumber()	47
5.6.2.3	tetrammHVenable()	48
5.6.2.4	tetrammHVSetVoltage()	48
5.6.2.5	tetrammSetRange()	48
5.7	EnBOX functions	50
5.7.1	Detailed Description	50
5.7.2	Function Documentation	50
5.7.2.1	enboxEncoderSelect()	50
5.7.2.2	enboxMathSelect()	50
5.8	Misc	51
5.8.1	Detailed Description	51
5.8.2	Function Documentation	51
5.8.2.1	getSFPinfo()	51
5.8.2.2	getSystemVersion()	51
<b>6</b>	<b>Class Documentation</b>	<b>55</b>
6.1	best_buffer_line Struct Reference	55
6.1.1	Detailed Description	55
6.2	best_stats Struct Reference	55
6.2.1	Detailed Description	55
<b>7</b>	<b>File Documentation</b>	<b>57</b>

---

7.1 best_c_interface.c File Reference	57
7.1.1 Macro Definition Documentation	59
7.1.1.1 debug_print	59
7.1.1.2 LIB_BEST_DEBUG	59
7.1.1.3 VERSION_STRING	59
7.1.1.4 VERSION_STRING_HELPER	59
7.1.2 Function Documentation	59
7.1.2.1 __attribute__([1/2])	60
7.1.2.2 __attribute__([2/2])	60
7.1.2.3 getBestLock()	60
7.1.2.4 lock_sigaction()	60
7.1.2.5 releaseBestLock()	60
7.1.3 Variable Documentation	60
7.1.3.1 fd_DMA	60
7.1.3.2 fd_Lock	61
7.1.3.3 fd_Mbox	61
7.1.3.4 in	61
7.1.3.5 LIB_BEST_VERSION	61
7.1.3.6 loginLevel	61
7.1.3.7 out	61
7.1.3.8 p	61
7.1.3.9 SAMP_FREQ	61
7.2 best_c_interface.h File Reference	62
7.2.1 Macro Definition Documentation	65
7.2.1.1 BEST_FILE_CONFIG_INI	65
7.2.1.2 BEST_FILE_DISP_DMA	65
7.2.1.3 BEST_FILE_MAILBOX	65
7.2.1.4 FFT_LEN	66
7.2.1.5 LEN_BBF	66
7.2.1.6 LOCK_FILE	66
7.2.2 Variable Documentation	66
7.2.2.1 LIB_BEST_VERSION	66
7.3 best_c_interface_conf.cpp File Reference	66
7.3.1 Variable Documentation	68
7.3.1.1 fd_DMA	68
7.3.1.2 fd_Mbox	68
7.3.1.3 in	69
7.3.1.4 loginLevel	69
7.3.1.5 out	69
7.3.1.6 p	69
7.3.1.7 SAMP_FREQ	69
7.4 best_c_interface_data.cpp File Reference	69

---

---

7.4.1 Function Documentation . . . . .	70
7.4.1.1 fftw_abs() . . . . .	70
7.4.2 Variable Documentation . . . . .	70
7.4.2.1 fd_DMA . . . . .	71
7.4.2.2 fd_Mbox . . . . .	71
7.4.2.3 in . . . . .	71
7.4.2.4 loginLevel . . . . .	71
7.4.2.5 out . . . . .	71
7.4.2.6 p . . . . .	71
7.4.2.7 SAMP_FREQ . . . . .	71
7.5 best_c_interface_tetramm.cpp File Reference . . . . .	72
7.5.1 Variable Documentation . . . . .	72
7.5.1.1 fd_Mbox . . . . .	72
7.5.1.2 loginLevel . . . . .	73
7.6 README.md File Reference . . . . .	73
<b>Index</b>	<b>75</b>





# Chapter 1

## libbibest

Library to interface with BIBEST Central Unit FPGA.

### 1.0.1 changelog version

The last version in `debian/changelog` should be equal to the one in files `version` and in `build_number`. After succesfull compilation the `build_number` and the updated `debian/changelog` should be checked in the version control.

## 1.1 Before building

- update `.version` and `.buildnumber`
- update debian files
  - update changelog with `dch -i` accordingly to version and buildnumber
  - check `debian/control` file (if changes required)
- update changelog here in the [README.md](#)
- commit the changes
- `make publish`

## 1.2 Generated files

Generated `.deb` files are stored in `pkg` folder.

### 1.2.1 Build issues:

```
sudo apt install libfftw3-dev
```

## 1.3 CHANGELOG

### 1.3.1 1.2.7

- Update documentation

### 1.3.2 1.2.6

- Updated pcie\_mailbox submodule

### 1.3.3 1.2.5

- Added compatibility with Ubuntu 20.04 and 14.04
- Fixed set/get PIDparamSingle
- Removed setROC
- Fixed set/get BPMoffset
- Fixed set/get ROClimits
- Removed getROCandROI\_sel
- Added getTetrammsNumber

### 1.3.4 1.2.4

- Added getBPMoffset that reads from HW

### 1.3.5 1.2.3

- Added getROClimits, getWindAvg getROCandROI\_sel get BPMorient getCrossBar getBPMoffset
- Added getBPMscale function
- addde python script to test all function (~/.test/scanAllGetFunctions.py)
- updated sin/cos

# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

Data functions . . . . .	9
Configuration functions . . . . .	16
Access control functions . . . . .	35
PID control functions . . . . .	37
Function generator functions . . . . .	46
TetrAMM functions . . . . .	47
EnBOX functions . . . . .	50
Misc . . . . .	51



# 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> . . . . .	55
<a href="#">best_stats</a> . . . . .	55



# 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>	57
<a href="#">best_c_interface.h</a>	62
<a href="#">best_c_interface_conf.cpp</a>	66
<a href="#">best_c_interface_data.cpp</a>	69
<a href="#">best_c_interface_tetramm.cpp</a>	72





# Chapter 5

## Module Documentation

### 5.1 Data functions

Functions which read measurement and calculation data from system.

#### 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 [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 the time being)*
- 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 \*meanY, double \*stdY, double \*meanI0, double \*stdI0)  
*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 [],  
    int length )
```

Gets arrays of "samples" from display DMA.

**Parameters**

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

**Returns**

numbers of samples copies (should be equal to length)

**5.1.2.5 getCurrentArray()**

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

Gets arrays of currents from display DMA.

**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_dac0_ch0 = 16, offset_dac0_ch1 = 17, offset_dac0_ch2 = 18, offset_dac0_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 * IO,
    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>IO</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 getPosStats()**

```
int getPosStats (
    int selector,
    double * meanX,
    double * stdX,
    double * meanY,
    double * stdY,
    double * meanIO,
    double * stdIO )
```

Get position statistics.

**Parameters**

<i>selector</i>	Selects which TetrAmm
-----------------	-----------------------

**5.1.2.12 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)

**5.1.2.13 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

**5.1.2.14 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

Functions which configure system.

### Enumerations

- enum `best_pid_select` { `BEST_PID_SELECT_0` = 0, `BEST_PID_SELECT_1` = 1, `BEST_PID_SELECT_2` = 2, `BEST_PID_SELECT_3` = 3 }

### 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 \*roil0min, double \*roil0max, double \*roc)  
*Gets RoiX, RoiY, Roil0max, Roil0min 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 offsetXlf, double offsetXhf, double offsetYlf, double offsetYhf)  
*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 posXlf, uint8\_t posXhf, uint8\_t posYlf, uint8\_t posYhf)  
*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 \*posXlf, uint8\_t \*posXhf, uint8\_t \*posYlf, uint8\_t \*posYhf)  
*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]. Write BPM offsets to FPGA.*
- int `getBPMoffset` (double \*offsetX, double \*offsetY, int bpm)  
*Gets BPM offsets in [um]. 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 \*posXlf, int \*posXhf, int \*posYlf, int \*posYhf)  
*Gets BPM sources. Read BPM sources from FPGA.*
- int `setBPMselector` (int posXlf, int posXhf, int posYlf, int posYhf)  
*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↵ _0	
BEST_PID_SELECT↵ _1	
BEST_PID_SELECT↵ _2	
BEST_PID_SELECT↵ _3	

## 5.2.3 Function Documentation

### 5.2.3.1 getBPMoffset()

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

Gets BPM offsets in [um]. 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 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

### 5.2.3.3 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

### 5.2.3.4 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

### 5.2.3.5 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

5.2.3.6 `getBPMselector()`

```
int getBPMselector (
    int * posXlf,
    int * posXhf,
    int * posYlf,
    int * posYhf )
```

Gets BPM sources. Read BPM sources from FPGA.

## Parameters

<i>posXlf</i>	BPM (= TetrAMM) for posXlf
<i>posXhf</i>	BPM (= TetrAMM) for posXhf
<i>posYlf</i>	BPM (= TetrAMM) for posYlf
<i>posYhf</i>	BPM (= TetrAMM) for posYhf

## Returns

ACK

5.2.3.7 `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

**5.2.3.8 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

**5.2.3.9 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

### 5.2.3.10 getPIDoutSel()

```
int getPIDoutSel (
    uint8_t * posXlf,
    uint8_t * posXhf,
    uint8_t * posYlf,
    uint8_t * posYhf )
```

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

#### Parameters

<i>posXlf</i>	DAC channel for posXlf
<i>posXhf</i>	DAC channel for posXhf
<i>posYlf</i>	DAC channel for posYlf
<i>posYhf</i>	DAC channel for posYhf

#### Returns

0 on success, negative on fail

#### Note

Access level: all levels

### 5.2.3.11 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 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
<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

5.2.3.12 `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>roiI0min</i>	double parameter in [A]
<i>roiI0max</i>	double parameter in [A]
<i>roc</i>	double parameter in percentage

## Note

Access level: all levels

5.2.3.13 `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

**5.2.3.14 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

**5.2.3.15 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

**5.2.3.16 setBPMoffset()**

```
int setBPMoffset (
    double offsetX,
```



```
double offsetY,
int bpm )
```

Sets BPM offset in [um]. 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

#### 5.2.3.17 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

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

**Returns**

0 on success, -1 on fail

**Note**

Access level must be admin

**5.2.3.19 setBPMselector()**

```
int setBPMselector (
    int posXlf,
    int posXhf,
    int posYlf,
    int posYhf )
```

Sets BPM sources. Write BPM sources to FPGA.

**Parameters**

<i>posXlf</i>	BPM (= TetrAMM) for posXlf
<i>posXhf</i>	BPM (= TetrAMM) for posXhf
<i>posYlf</i>	BPM (= TetrAMM) for posYlf
<i>posYhf</i>	BPM (= TetrAMM) for posYhf

**Returns**

ACK

**Note**

Access level: all levels

**5.2.3.20 setCrossBar()**

```
int setCrossBar (
    uint8_t sel,
    uint8_t ch0,
```

```
uint8_t ch1,  
uint8_t ch2,  
uint8_t ch3 )
```

Sets input cross switch.

**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

**Note**

Access level must be admin

**5.2.3.21 setOffset()**

```
int setOffset (
    double offsetXlf,
    double offsetXhf,
    double offsetYlf,
    double offsetYhf )
```

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

**Parameters**

<i>offsetXlf</i>	in [V]
<i>offsetXhf</i>	in [V]
<i>offsetYlf</i>	in [V]
<i>offsetYhf</i>	in [V]

**Returns**

0 on success, negative on fail

**Note**

Access level must be admin

### 5.2.3.2 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

**5.2.3.23 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

**5.2.3.24 setPIDOutSel()**

```
int setPIDOutSel (
    uint8_t posXlf,
    uint8_t posXhf,
    uint8_t posYlf,
    uint8_t posYhf )
```

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

## Parameters

<i>posXlf</i>	DAC channel for posXlf
<i>posXhf</i>	DAC channel for posXhf
<i>posYlf</i>	DAC channel for posYlf
<i>posYhf</i>	DAC channel for posYhf

## Returns

0 on success, negative on fail

## Note

Access level must be admin

## 5.2.3.25 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 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
<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

**5.2.3.26 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 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
<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

**5.2.3.27 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>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 must be admin

**5.2.3.28 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

**5.2.3.29 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 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
<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 which allow to elevate user access level.

### 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 which control PID behaviour.

### Functions

- unsigned 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.*
- 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 [setFiltControl](#) (char filt\_sel, char pos\_sel, char enable)  
*Sets IIR controller parameter.*
- int [getFiltControl](#) (char filt\_sel, char pos\_sel, char \*enable)  
*Sets IIR controller parameter.*
- int [setFiltCoef](#) (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double coef)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int [commitFiltCoef](#) (char filt\_sel, char pos\_sel)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int [getFiltCoef](#) (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double \*coef)  
*Gets Filter (IIR) coefficients (IIR filter on PosX and PosY)*

### 5.4.1 Detailed Description

Functions which control PID behaviour.

## 5.4.2 Function Documentation

### 5.4.2.1 commitFiltCoef()

```
int commitFiltCoef (
    char filt_sel,
    char pos_sel )
```

Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)

#### Parameters

<i>filt_sel</i>	Selects filter ( 0 = lowpass, 1 = band pass )
<i>pos_sel</i>	Selects position (0 = X, 1 = Y)

#### Returns

0 on success, negative on fail

This function commits the temporary IIR coefficients

### 5.4.2.2 getFiltCoef()

```
int getFiltCoef (
    char filt_sel,
    char pos_sel,
    char param_is_a,
    char coef_sel,
    double * coef )
```

Gets Filter (IIR) coefficients (IIR filter on PosX and PosY)

#### Parameters

<i>filt_sel</i>	Selects filter ( 0 = lowpass, 1 = band pass )
<i>pos_sel</i>	Selects position (0 = X, 1 = Y)
<i>param_is_a</i>	Selects if param read is a (when 1) or b (when 0)
<i>coef_sel</i>	Selects coeff (0-4)
<i>coef</i>	coeff value in double

This function gets the IIR coefficients, one at the time

### 5.4.2.3 getFiltControl()

```
int getFiltControl (
    char filt_sel,
```

```
char pos_sel,
char * enable )
```

Sets IIR controller parameter.

#### Parameters

<i>filt_sel</i>	0 = Low Pass
<i>pos_sel</i>	Selects position (0 = X, 1 = Y)
<i>enable</i>	Selects filter ( 0 = lowpass, 1 = full band )

#### Returns

0 on success, negative on fail

This function gets the type of filter applied on position X>If and Y>If

#### 5.4.2.4 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

#### 5.4.2.5 getPIDstatus()

```
unsigned int getPIDstatus (
void )
```

Gets PID status.

#### Returns

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

#### Note

Access level: all levels

### 5.4.2.6 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 ( Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3 )
<i>setpt</i>	Setpoint in [um]

#### Returns

0 on success, negative on fail

#### Note

Access level: all levels

### 5.4.2.7 getSetpoint2()

```
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.

#### Parameters

<i>sel</i>	Selects which PID ( Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3 )
<i>setpt</i>	Setpoint in [um]

#### Returns

0 on success, negative on fail

### 5.4.2.8 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

## 5.4.2.9 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).

## 5.4.2.10 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 IIRcontroller↔CommitParams function to download parameters to controller.

**5.4.2.11 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

**5.4.2.12 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

**5.4.2.13 setFiltCoef()**

```
int setFiltCoef (
    char filt_sel,
    char pos_sel,
    char param_is_a,
    char coef_sel,
    double coef )
```

Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)

## Parameters

<i>filt_sel</i>	Selects filter ( 0 = lowpass, 1 = band pass )
<i>pos_sel</i>	Selects position (0 = X, 1 = Y)
<i>param_is_a</i>	Selects if param read is a (when 1) or b (when 0)
<i>coef_sel</i>	Selects coeff (0-4)
<i>coef</i>	coeff value in double

## Returns

0 on success, negative on fail

This function sets the IIR coefficients, one at the time

**5.4.2.14 setFiltControl()**

```
int setFiltControl (
    char filt_sel,
    char pos_sel,
    char enable )
```

Sets IIR controller parameter.

## Parameters

<i>filt_sel</i>	0 = Low Pass
<i>pos_sel</i>	Selects position (0 = X, 1 = Y)
<i>enable</i>	Selects filter ( 0 = lowpass, 1 = full band )

## Returns

0 on success, negative on fail

This function sets the type of filter applied on position X\_if and Y\_if

**5.4.2.15 setPIDconf()**

```
int setPIDconf (
    char conf )
```

Sets PID controller configuration.

## Parameters

<i>conf</i>	PID conf 0 = 0 1 = 01 2 = 02 3 = 03 4 = 012 5 = 013 6 = 023 7 = 0123 8 = 1 9 = 12 10 = 13 11 = 123 12 = 2 13 = 23 14 = 3
-------------	--

**Returns**

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

**Note**

Access level must be user or admin

**5.4.2.16 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 ( Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3 )
<i>setpt</i>	Setpoint in [um]

**Returns**

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

**Note**

Access level must be user or admin

**5.4.2.17 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 ( Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3 )
<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()

```
int funcGenEnable (  
    char enable )
```

Enables or disables function generator.

#### 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.*

### 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.

#### Parameters

<code>pos_A</code>	on SFP A, 0 = No TetrAMM connected, 1 = TetrAMM connected
<code>pos_B</code>	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*)
- int [enboxMathSelect](#) (int *math\_func*, unsigned char *selector*)

### 5.7.1 Detailed Description

### 5.7.2 Function Documentation

#### 5.7.2.1 enboxEncoderSelect()

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

##### Parameters

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

##### Note

Access level must be admin

#### 5.7.2.2 enboxMathSelect()

```
int enboxMathSelect (
    int math_func,
    unsigned char selector )
```

##### Parameters

<i>math_func</i>	0 = normal, 1 = sum-diff between encoder0 and encoder1 on the same bpm
<i>selector</i>	0 = EnBOX on SFP A, 1 = EnBOX on SFP B

##### Note

Access level must be admin

## 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) #X`
- #define `VERSION_STRING(X) VERSION_STRING_HELPER(X)`
- #define `LIB_BEST_DEBUG 0`
- #define `debug_print(fmt, ...)` do { if (`LIB_BEST_DEBUG`) fprintf(stderr, "[libbest] " fmt, \_\_VA\_ARGS\_\_); } while (0)

#### 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.*
- 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 `setFiltControl` (char filt\_sel, char pos\_sel, char enable)  
*Sets IIR controller parameter.*
- int `getFiltControl` (char filt\_sel, char pos\_sel, char \*enable)  
*Sets IIR controller parameter.*
- int `setFiltCoef` (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double coef)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int `commitFiltCoef` (char filt\_sel, char pos\_sel)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int `getFiltCoef` (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double \*coef)  
*Gets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int `funcGenConfig` (float freq, float ampl[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,  
    ... ) 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 ) VERSION\_STRING\_HELPER(X)
```

### 7.1.1.4 VERSION\_STRING\_HELPER

```
#define VERSION_STRING_HELPER(  
    X ) #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/BIBEST/config_bibest.ini"`
- #define `FFT_LEN` (2048)
- #define `LEN_BBF` (32)

### Enumerations

- enum `best_pid_select` { `BEST_PID_SELECT_0` = 0, `BEST_PID_SELECT_1` = 1, `BEST_PID_SELECT_2` = 2, `BEST_PID_SELECT_3` = 3 }

### 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 `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 `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 \*meanY, double \*stdY, double \*meanI0, double \*stdI0)  
*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 \*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 offsetXlf, double offsetXhf, double offsetYlf, double offsetYhf)

*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 posXlf, uint8\_t posXhf, uint8\_t posYlf, uint8\_t posYhf)

*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 \*posXlf, uint8\_t \*posXhf, uint8\_t \*posYlf, uint8\_t \*posYhf)

*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]. Write BPM offsets to FPGA.*
- int [getBPMoffset](#) (double \*offsetX, double \*offsetY, int bpm)

*Gets BPM offsets in [um]. 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 \*posXlf, int \*posXhf, int \*posYlf, int \*posYhf)

*Gets BPM sources. Read BPM sources from FPGA.*

- int [setBPMselector](#) (int posXlf, int posXhf, int posYlf, int posYhf)  
*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\(\)](#).*
- unsigned 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.*
- 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 [setFiltControl](#) (char filt\_sel, char pos\_sel, char enable)  
*Sets IIR controller parameter.*
- int [getFiltControl](#) (char filt\_sel, char pos\_sel, char \*enable)  
*Sets IIR controller parameter.*
- int [setFiltCoef](#) (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double coef)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int [commitFiltCoef](#) (char filt\_sel, char pos\_sel)  
*Sets Filter (IIR) coefficients (IIR filter on PosX and PosY)*
- int [getFiltCoef](#) (char filt\_sel, char pos\_sel, char param\_is\_a, char coef\_sel, double \*coef)  
*Gets Filter (IIR) coefficients (IIR filter on PosX and PosY)*



- int [funcGenConfig](#) (float freq, float amp[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.*
- int [enboxEncoderSelect](#) (int enc\_type, unsigned char selector)
- int [enboxMathSelect](#) (int math\_func, unsigned char selector)
- 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/BIBEST/config_bibest.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 Variable Documentation

#### 7.2.2.1 LIB\_BEST\_VERSION

```
char LIB_BEST_VERSION[ ]
```

## 7.3 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

- unsigned int [getPIDstatus](#) (void)  
*Gets PID status.*
- int [getPosStats](#) (int selector, double \*meanX, double \*stdX, double \*meanY, double \*stdY, double \*meanI0, double \*stdI0)  
*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 offsetXlf, double offsetXhf, double offsetYlf, double offsetYhf)  
*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 posXlf, uint8\_t posXhf, uint8\_t posYlf, uint8\_t posYhf)  
*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 \*posXlf, uint8\_t \*posXhf, uint8\_t \*posYlf, uint8\_t \*posYhf)  
*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]. Write BPM offsets to FPGA.*
- int [getBPMoffset](#) (double \*offsetX, double \*offsetY, int bpm)  
*Gets BPM offsets in [um]. Read BPM offsets from FPGA.*
- int [getBPMselector](#) (int \*posXlf, int \*posXhf, int \*posYlf, int \*posYhf)  
*Gets BPM sources. Read BPM sources from FPGA.*
- int [setBPMselector](#) (int posXlf, int posXhf, int posYlf, int posYhf)  
*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](#)

### 7.3.1 Variable Documentation

#### 7.3.1.1 fd\_DMA

```
int fd_DMA
```

#### 7.3.1.2 fd\_Mbox

```
int fd_Mbox
```

### 7.3.1.3 in

```
double* in
```

### 7.3.1.4 loginLevel

```
int loginLevel
```

### 7.3.1.5 out

```
fftw_complex* out
```

### 7.3.1.6 p

```
fftw_plan p
```

### 7.3.1.7 SAMP\_FREQ

```
int SAMP_FREQ
```

## 7.4 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 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\\_sel](#) (int sel, double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for the time beign)*
- int [getVoltageArray](#) (double(\*voltages)[4], unsigned int length)  
*Gets arrays of voltages from display DMA.*
- 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 removeDC)  
*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](#)

### 7.4.1 Function Documentation

#### 7.4.1.1 [fftw\\_abs\(\)](#)

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

### 7.4.2 Variable Documentation

#### 7.4.2.1 fd\_DMA

```
int fd_DMA
```

#### 7.4.2.2 fd\_Mbox

```
int fd_Mbox
```

#### 7.4.2.3 in

```
double* in
```

#### 7.4.2.4 loginLevel

```
int loginLevel
```

#### 7.4.2.5 out

```
fftw_complex* out
```

#### 7.4.2.6 p

```
fftw_plan p
```

#### 7.4.2.7 SAMP\_FREQ

```
int SAMP_FREQ
```

## 7.5 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.*
- int [getTetramms](#) (int \*pos\_A, int \*pos\_B)  
*Gets TetrAMMs.*
- int [enboxEncoderSelect](#) (int enc\_type, unsigned char selector)
- int [enboxMathSelect](#) (int math\_func, unsigned char selector)

### Variables

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

#### 7.5.1 Variable Documentation

##### 7.5.1.1 fd\_Mbox

```
int fd_Mbox
```



### 7.5.1.2 loginLevel

```
int loginLevel
```

## 7.6 README.md File Reference



# Index

- `__attribute__`
  - `best_c_interface.c`, 59, 60
  - Data functions, 10
- Access control functions, 35
  - `getLock`, 35
  - `getLockStatus`, 35
- `best_buffer_line`, 55
- `best_c_interface.c`, 57
  - `__attribute__`, 59, 60
  - `debug_print`, 59
  - `fd_DMA`, 60
  - `fd_Lock`, 60
  - `fd_Mbox`, 61
  - `getBestLock`, 60
  - in, 61
  - `LIB_BEST_DEBUG`, 59
  - `LIB_BEST_VERSION`, 61
  - `lock_sigaction`, 60
  - `loginLevel`, 61
  - out, 61
  - p, 61
  - `releaseBestLock`, 60
  - `SAMP_FREQ`, 61
  - `VERSION_STRING`, 59
  - `VERSION_STRING_HELPER`, 59
- `best_c_interface.h`, 62
  - `BEST_FILE_CONFIG_INI`, 65
  - `BEST_FILE_DISP_DMA`, 65
  - `BEST_FILE_MAILBOX`, 65
  - `FFT_LEN`, 65
  - `LEN_BBF`, 66
  - `LIB_BEST_VERSION`, 66
  - `LOCK_FILE`, 66
- `best_c_interface_conf.cpp`, 66
  - `fd_DMA`, 68
  - `fd_Mbox`, 68
  - in, 68
  - `loginLevel`, 69
  - out, 69
  - p, 69
  - `SAMP_FREQ`, 69
- `best_c_interface_data.cpp`, 69
  - `fd_DMA`, 70
  - `fd_Mbox`, 71
  - `fftw_abs`, 70
  - in, 71
  - `loginLevel`, 71
  - out, 71
  - p, 71
  - `SAMP_FREQ`, 71
- `best_c_interface_tetramm.cpp`, 72
  - `fd_Mbox`, 72
  - `loginLevel`, 72
- `BEST_FILE_CONFIG_INI`
  - `best_c_interface.h`, 65
- `BEST_FILE_DISP_DMA`
  - `best_c_interface.h`, 65
- `BEST_FILE_MAILBOX`
  - `best_c_interface.h`, 65
- `best_pid_select`
  - Configuration functions, 17
- `BEST_PID_SELECT_0`
  - Configuration functions, 17
- `BEST_PID_SELECT_1`
  - Configuration functions, 17
- `BEST_PID_SELECT_2`
  - Configuration functions, 17
- `BEST_PID_SELECT_3`
  - Configuration functions, 17
- `best_stats`, 55
- `commitFiltCoef`
  - PID control functions, 38
- Configuration functions, 16
  - `best_pid_select`, 17
  - `BEST_PID_SELECT_0`, 17
  - `BEST_PID_SELECT_1`, 17
  - `BEST_PID_SELECT_2`, 17
  - `BEST_PID_SELECT_3`, 17
  - `getBPMoffset`, 18
  - `getBPMorient`, 18
  - `getBPMscale`, 18
  - `getBPMscaling`, 19
  - `getBPMscaling_sel`, 19
  - `getBPMselector`, 20
  - `getCrossBar`, 20
  - `getOffsetSingle`, 21
  - `getOutputMux`, 21
  - `getPIDoutSel`, 21
  - `getPIDparamSingle`, 22
  - `getROCandROI`, 22
  - `getROCenable`, 23
  - `getROClimits`, 24
  - `getWindAvg`, 24
  - `setBPMoffset`, 24
  - `setBPMorient`, 25
  - `setBPMscale`, 25
  - `setBPMselector`, 26

- setCrossBar, 26
- setOffset, 28
- setOffsetSingle, 28
- setOutputMux, 30
- setPIDoutSel, 30
- setPIDparams, 31
- setPIDparamSingle, 32
- setROCEnable, 32
- setROCLimits, 33
- setWindAvg, 33
- Data functions, 9
  - \_\_attribute\_\_, 10
  - getBPMStats, 10
  - getBuffer, 10
  - getCurrentArray, 11
  - getCurrentArray\_sel, 11
  - getDisplayFreq, 11
  - getFFT, 12
  - getPosArray, 12
  - getPosArray\_sel, 12
  - getPosStats, 13
  - getVoltageArray, 13
  - getVoltageArray\_sel, 14
  - setDisplayFreq, 14
- debug\_print
  - best\_c\_interface.c, 59
- EnBOX functions, 50
  - enboxEncoderSelect, 50
  - enboxMathSelect, 50
- enboxEncoderSelect
  - EnBOX functions, 50
- enboxMathSelect
  - EnBOX functions, 50
- fd\_DMA
  - best\_c\_interface.c, 60
  - best\_c\_interface\_conf.cpp, 68
  - best\_c\_interface\_data.cpp, 70
- fd\_Lock
  - best\_c\_interface.c, 60
- fd\_Mbox
  - best\_c\_interface.c, 61
  - best\_c\_interface\_conf.cpp, 68
  - best\_c\_interface\_data.cpp, 71
  - best\_c\_interface\_tetramm.cpp, 72
- FFT\_LEN
  - best\_c\_interface.h, 65
- fftw\_abs
  - best\_c\_interface\_data.cpp, 70
- funcGenConfig
  - Function generator functions, 46
- funcGenEnable
  - Function generator functions, 46
- Function generator functions, 46
  - funcGenConfig, 46
  - funcGenEnable, 46
- getBestLock
  - best\_c\_interface.c, 60
- getBPMoffset
  - Configuration functions, 18
- getBPMorient
  - Configuration functions, 18
- getBPMscale
  - Configuration functions, 18
- getBPMscaling
  - Configuration functions, 19
- getBPMscaling\_sel
  - Configuration functions, 19
- getBPMselector
  - Configuration functions, 20
- getBPMStats
  - Data functions, 10
- getBuffer
  - Data functions, 10
- getCrossBar
  - Configuration functions, 20
- getCurrentArray
  - Data functions, 11
- getCurrentArray\_sel
  - Data functions, 11
- getDisplayFreq
  - Data functions, 11
- getFFT
  - Data functions, 12
- getFiltCoef
  - PID control functions, 38
- getFiltControl
  - PID control functions, 38
- getLock
  - Access control functions, 35
- getLockStatus
  - Access control functions, 35
- getOffsetSingle
  - Configuration functions, 21
- getOutputMux
  - Configuration functions, 21
- getPIDconf
  - PID control functions, 39
- getPIDoutSel
  - Configuration functions, 21
- getPIDparamSingle
  - Configuration functions, 22
- getPIDstatus
  - PID control functions, 39
- getPosArray
  - Data functions, 12
- getPosArray\_sel
  - Data functions, 12
- getPosStats
  - Data functions, 13
- getROCandROI
  - Configuration functions, 22
- getROCEnable
  - Configuration functions, 23

- getROClimits
  - Configuration functions, [24](#)
- getSetpoint
  - PID control functions, [39](#)
- getSetpoint2
  - PID control functions, [40](#)
- getSFPinfo
  - Misc, [51](#)
- getSystemVersion
  - Misc, [51](#)
- getTetramms
  - TetrAMM functions, [47](#)
- getTetrammsNumber
  - TetrAMM functions, [47](#)
- getVoltageArray
  - Data functions, [13](#)
- getVoltageArray\_sel
  - Data functions, [14](#)
- getWindAvg
  - Configuration functions, [24](#)
  
- IIRcontrollerCommitParams
  - PID control functions, [40](#)
- IIRcontrollerGetParam
  - PID control functions, [41](#)
- IIRcontrollerSetParam
  - PID control functions, [41](#)
- in
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [68](#)
  - [best\\_c\\_interface\\_data.cpp](#), [71](#)
  
- LEN\_BBF
  - [best\\_c\\_interface.h](#), [66](#)
- LIB\_BEST\_DEBUG
  - [best\\_c\\_interface.c](#), [59](#)
- LIB\_BEST\_VERSION
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface.h](#), [66](#)
- LOCK\_FILE
  - [best\\_c\\_interface.h](#), [66](#)
- lock\_sigaction
  - [best\\_c\\_interface.c](#), [60](#)
- loginLevel
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [69](#)
  - [best\\_c\\_interface\\_data.cpp](#), [71](#)
  - [best\\_c\\_interface\\_tetramm.cpp](#), [72](#)
  
- Misc, [51](#)
  - [getSFPinfo](#), [51](#)
  - [getSystemVersion](#), [51](#)
  
- out
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [69](#)
  - [best\\_c\\_interface\\_data.cpp](#), [71](#)
  
- p
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [69](#)
  - [best\\_c\\_interface\\_data.cpp](#), [71](#)
  
- PID control functions, [37](#)
  - [commitFiltCoef](#), [38](#)
  - [getFiltCoef](#), [38](#)
  - [getFiltControl](#), [38](#)
  - [getPIDconf](#), [39](#)
  - [getPIDstatus](#), [39](#)
  - [getSetpoint](#), [39](#)
  - [getSetpoint2](#), [40](#)
  - [IIRcontrollerCommitParams](#), [40](#)
  - [IIRcontrollerGetParam](#), [41](#)
  - [IIRcontrollerSetParam](#), [41](#)
  - [setCtrlReset](#), [42](#)
  - [setFBenable](#), [42](#)
  - [setFiltCoef](#), [42](#)
  - [setFiltControl](#), [43](#)
  - [setPIDconf](#), [43](#)
  - [setSetpoint](#), [44](#)
  - [setSetpoint2](#), [44](#)
  
- README.md, [73](#)
- releaseBestLock
  - [best\\_c\\_interface.c](#), [60](#)
  
- SAMP\_FREQ
  - [best\\_c\\_interface.c](#), [61](#)
  - [best\\_c\\_interface\\_conf.cpp](#), [69](#)
  - [best\\_c\\_interface\\_data.cpp](#), [71](#)
- setBPMoffset
  - Configuration functions, [24](#)
- setBPMorient
  - Configuration functions, [25](#)
- setBPMscale
  - Configuration functions, [25](#)
- setBPMselector
  - Configuration functions, [26](#)
- setCrossBar
  - Configuration functions, [26](#)
- setCtrlReset
  - PID control functions, [42](#)
- setDisplayFreq
  - Data functions, [14](#)
- setFBenable
  - PID control functions, [42](#)
- setFiltCoef
  - PID control functions, [42](#)
- setFiltControl
  - PID control functions, [43](#)
- setOffset
  - Configuration functions, [28](#)
- setOffsetSingle
  - Configuration functions, [28](#)
- setOutputMux
  - Configuration functions, [30](#)
- setPIDconf
  - PID control functions, [43](#)
- setPIDoutSel

- Configuration functions, [30](#)
- setPIDparams
  - Configuration functions, [31](#)
- setPIDparamSingle
  - Configuration functions, [32](#)
- setROCenable
  - Configuration functions, [32](#)
- setROClimits
  - Configuration functions, [33](#)
- setSetpoint
  - PID control functions, [44](#)
- setSetpoint2
  - PID control functions, [44](#)
- setWindAvg
  - Configuration functions, [33](#)
  
- TetrAMM functions, [47](#)
  - getTetramms, [47](#)
  - getTetrammsNumber, [47](#)
  - tetrammHVenable, [48](#)
  - tetrammHVSetVoltage, [48](#)
  - tetrammSetRange, [48](#)
- tetrammHVenable
  - TetrAMM functions, [48](#)
- tetrammHVSetVoltage
  - TetrAMM functions, [48](#)
- tetrammSetRange
  - TetrAMM functions, [48](#)
  
- VERSION\_STRING
  - best\_c\_interface.c, [59](#)
- VERSION\_STRING\_HELPER
  - best\_c\_interface.c, [59](#)