

BEST Interface

Generated by Doxygen 1.8.17

1 libbest	1
1.0.1 changelog version	1
1.1 Before building	1
1.2 Generated files	1
1.2.1 Build issues:	1
1.3 TEST	2
1.4 Changelog	2
1.4.1 1.8-18	2
1.4.2 1.8-17	2
1.4.3 1.8-16	2
1.4.4 1.8-15	2
1.4.5 1.8-14	2
1.4.6 1.8-13	2
1.4.7 1.8-12	2
1.4.8 1.8-11	3
1.4.9 1.8-10	3
1.4.10 1.8-9	3
1.4.11 1.8-8	3
1.4.12 1.8-7	3
1.4.13 1.8-6	3
1.4.14 1.8-5	3
1.5 Funtion List	3
1.5.1 Data functions	3
1.5.1.1 getBuffer	4
1.5.1.2 getPosArray_sel	4
1.5.1.3 getPosArray	4
1.5.1.4 getVoltageArray	4
1.5.1.5 getVoltageArray_sel	4
1.5.1.6 getCurrentArray	4
1.5.1.7 getCurrentArray_sel	4
1.5.1.8 getFFT	4
1.5.1.9 getPosStats	4
1.5.1.10 setDisplayFreq	4
1.5.1.11 getDisplayFreq	4
1.5.1.12 getBPMStats	4
1.5.2 Configuration functions	4
1.5.2.1 setOffsetSingle	4
1.5.2.2 getOffsetSingle	4
1.5.2.3 setOffset	4
1.5.2.4 setWindAvg	4
1.5.2.5 getWindAvg (added in v1.8-9)	4
1.5.2.6 setPIDparams	4

1.5.2.7 setPIDparamSingle	4
1.5.2.8 getPIDparamSingle	4
1.5.2.9 setPIDoutSel	4
1.5.2.10 getPIDoutSel	4
1.5.2.11 setCrossBar	5
1.5.2.12 getCrossBar (1.8-6)	5
1.5.2.13 setBPMorient	5
1.5.2.14 getBPMorient	5
1.5.2.15 setOutputMux	5
1.5.2.16 getOutputMux	5
1.5.2.17 setBPMscale	5
1.5.2.18 getBPMscale	5
1.5.2.19 getBPMscaling_sel	5
1.5.2.20 getBPMscaling	5
1.5.2.21 setBPMoffset	5
1.5.2.22 getBPMoffset (1.8-8)	5
1.5.2.23 getBPMselector (1.8-5)	5
1.5.2.24 setBPMselector	5
1.5.2.25 getROCandROI	5
1.5.2.26 getROCenable	5
1.5.2.27 setROCenable	6
1.5.2.28 setROC	6
1.5.2.29 setROClimits	6
1.5.2.30 getROClimits (1.8-7)	6
1.5.2.31 getROCandROI_sel (TODO)	6
1.5.3 Access control functions	6
1.5.3.1 getLock	6
1.5.3.2 getLockStatus	6
1.5.4 PID control functions	6
1.5.4.1 getPIDstatus	6
1.5.4.2 setFBenable	6
1.5.4.3 setCtrlReset	6
1.5.4.4 setPIDconf	6
1.5.4.5 getPIDconf	6
1.5.4.6 setSetpoint	6
1.5.4.7 getSetpoint	7
1.5.4.8 setSetpoint2 (1.8-8)	7
1.5.4.9 getSetpoint2 (1.8-8)	7
1.5.4.10 IIRcontrollerSetParam	7
1.5.4.11 IIRcontrollerCommitParams	7
1.5.4.12 IIRcontrollerGetParam	7
1.5.5 Function gen control functions	7

1.5.5.1 funcGenConfig	7
1.5.5.2 funcGenEnable	7
1.5.6 TetrAMM functions	7
1.5.6.1 tetrammHVSetVoltage	7
1.5.6.2 tetrammSetRange	7
1.5.6.3 tetrammHVenable	7
1.5.7 EnBOX functions	7
1.5.7.1 enboxEncoderSelect	7
1.5.7.2 enboxMathSelect	7
1.5.8 Misc functions	7
1.5.8.1 getSystemVersion	7
1.5.8.2 getSFPinfo	7
2 Module Index	9
2.1 Modules	9
3 Class Index	11
3.1 Class List	11
4 File Index	13
4.1 File List	13
5 Module Documentation	15
5.1 Data functions	15
5.1.1 Detailed Description	16
5.1.2 Function Documentation	16
5.1.2.1 __attribute__([1/2])	16
5.1.2.2 __attribute__([2/2])	16
5.1.2.3 getBPMStats()	16
5.1.2.4 getBuffer()	16
5.1.2.5 getCurrentArray()	16
5.1.2.6 getCurrentArray_sel()	17
5.1.2.7 getDisplayFreq()	17
5.1.2.8 getFFT()	17
5.1.2.9 getPosArray()	18
5.1.2.10 getPosArray_sel()	18
5.1.2.11 getPosStats()	18
5.1.2.12 getVoltageArray()	18
5.1.2.13 getVoltageArray_sel()	19
5.1.2.14 setDisplayFreq()	19
5.2 Configuration functions	20
5.2.1 Detailed Description	21
5.2.2 Enumeration Type Documentation	21
5.2.2.1 best_pid_select	21

5.2.3 Function Documentation	22
5.2.3.1 getBPMoffset()	22
5.2.3.2 getBPMoffsetEnbox()	22
5.2.3.3 getBPMorient()	22
5.2.3.4 getBPMscale()	23
5.2.3.5 getBPMscaling()	23
5.2.3.6 getBPMscaling_sel()	23
5.2.3.7 getBPMselector()	24
5.2.3.8 getCrossBar()	24
5.2.3.9 getOffsetSingle()	25
5.2.3.10 getOutputMux()	25
5.2.3.11 getPIDoutSel()	25
5.2.3.12 getPIDparamSingle()	26
5.2.3.13 getROCandROI()	26
5.2.3.14 getROCenable()	27
5.2.3.15 getROClimits()	27
5.2.3.16 getWindAvg()	27
5.2.3.17 setBPMdevice()	28
5.2.3.18 setBPMoffset()	28
5.2.3.19 setBPMoffsetEnbox()	29
5.2.3.20 setBPMorient()	29
5.2.3.21 setBPMscale()	29
5.2.3.22 setBPMselector()	31
5.2.3.23 setCrossBar()	31
5.2.3.24 setEnboxSumDiff()	32
5.2.3.25 setEnboxType()	32
5.2.3.26 setOffset()	32
5.2.3.27 setOffsetSingle()	33
5.2.3.28 setOutputMux()	33
5.2.3.29 setPIDoutSel()	34
5.2.3.30 setPIDparams()	34
5.2.3.31 setPIDparamSingle()	35
5.2.3.32 setROCenable()	35
5.2.3.33 setROClimits()	36
5.2.3.34 setWindAvg()	36
5.3 Access control functions	37
5.3.1 Detailed Description	37
5.3.2 Function Documentation	37
5.3.2.1 getLock()	37
5.3.2.2 getLockStatus()	37
5.4 PID control functions	38
5.4.1 Detailed Description	38

5.4.2 Function Documentation	38
5.4.2.1 getPIDconf()	38
5.4.2.2 getPIDstatus()	39
5.4.2.3 getSetpoint()	39
5.4.2.4 getSetpoint2()	39
5.4.2.5 IIRcontrollerCommitParams()	40
5.4.2.6 IIRcontrollerGetParam()	40
5.4.2.7 IIRcontrollerSetParam()	40
5.4.2.8 setCtrlReset()	41
5.4.2.9 setFBenable()	41
5.4.2.10 setPIDconf()	41
5.4.2.11 setSetpoint()	42
5.4.2.12 setSetpoint2()	42
5.5 Function generator functions	43
5.5.1 Detailed Description	43
5.5.2 Function Documentation	43
5.5.2.1 funcGenConfig()	43
5.5.2.2 funcGenEnable()	43
5.6 TetrAMM functions	44
5.6.1 Detailed Description	44
5.6.2 Function Documentation	44
5.6.2.1 getTetramms()	44
5.6.2.2 getTetrammsNumber()	44
5.6.2.3 tetrammHVenable()	44
5.6.2.4 tetrammHVSetVoltage()	45
5.6.2.5 tetrammSetRange()	45
5.7 EnBOX functions	46
5.7.1 Detailed Description	46
5.7.2 Function Documentation	46
5.7.2.1 enboxEncoderSelect()	46
5.7.2.2 getEnboxes()	46
5.7.2.3 getEnboxesNumber()	46
5.8 Misc	48
5.8.1 Detailed Description	48
5.8.2 Function Documentation	48
5.8.2.1 getSFPinfo()	48
5.8.2.2 getSystemVersion()	48
6 Class Documentation	51
6.1 best_buffer_line Struct Reference	51
6.1.1 Detailed Description	51
6.2 best_stats Struct Reference	51

6.2.1 Detailed Description	51
7 File Documentation	53
7.1 best_c_interface.c File Reference	53
7.1.1 Macro Definition Documentation	54
7.1.1.1 debug_print	55
7.1.1.2 LIB_BEST_DEBUG	55
7.1.1.3 VERSION_STRING	55
7.1.1.4 VERSION_STRING_HELPER	55
7.1.2 Function Documentation	55
7.1.2.1 __attribute__([1/2])	55
7.1.2.2 __attribute__([2/2])	55
7.1.2.3 getBestLock()	55
7.1.2.4 lock_sigaction()	55
7.1.2.5 releaseBestLock()	55
7.1.3 Variable Documentation	56
7.1.3.1 fd_DMA	56
7.1.3.2 fd_Lock	56
7.1.3.3 fd_Mbox	56
7.1.3.4 in	56
7.1.3.5 LIB_BEST_VERSION	56
7.1.3.6 loginLevel	56
7.1.3.7 out	56
7.1.3.8 p	56
7.1.3.9 SAMP_FREQ	56
7.2 best_c_interface.h File Reference	56
7.2.1 Macro Definition Documentation	60
7.2.1.1 BEST_FILE_CONFIG_INI	60
7.2.1.2 BEST_FILE_DISP_DMA	60
7.2.1.3 BEST_FILE_MAILBOX	60
7.2.1.4 FFT_LEN	60
7.2.1.5 LEN_BBF	60
7.2.1.6 LOCK_FILE	60
7.2.2 Variable Documentation	60
7.2.2.1 LIB_BEST_VERSION	60
7.3 best_c_interface_conf.cpp File Reference	61
7.3.1 Variable Documentation	63
7.3.1.1 fd_DMA	63
7.3.1.2 fd_Mbox	63
7.3.1.3 in	63
7.3.1.4 loginLevel	63
7.3.1.5 out	63

7.3.1.6 p	63
7.3.1.7 SAMP_FREQ	63
7.4 best_c_interface_data.cpp File Reference	63
7.4.1 Function Documentation	64
7.4.1.1 fftw_abs()	64
7.4.2 Variable Documentation	65
7.4.2.1 fd_DMA	65
7.4.2.2 fd_Mbox	65
7.4.2.3 in	65
7.4.2.4 loginLevel	65
7.4.2.5 out	65
7.4.2.6 p	65
7.4.2.7 SAMP_FREQ	65
7.5 best_c_interface_tetramm.cpp File Reference	65
7.5.1 Variable Documentation	66
7.5.1.1 fd_Mbox	66
7.5.1.2 loginLevel	66
7.6 README.md File Reference	66
Index	67

Chapter 1

libbest

Library to interface with BEST 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 [Readme.md](#) -> Changelog and FunctionList
- commit the changes
- `make publish`
- `doxy Doxyfile` to update the documentation

1.2 Generated files

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

1.2.1 Build issues:

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

1.3 TEST

A python script (scanAllGetFunctions.py) can be used to have an idea of who is changing the BEST internal variables

1.4 Changelog

1.4.1 1.8-18

- Fixed setBPMoffsetEnbox compatible with firmware 1.2.20
- Added setBPMdevice, setEnboxType, setEnboxSumDiff to FPGA for Difference Over Sum Calculations
- Removed enboxMathSelect

1.4.2 1.8-17

- Added additional channel for func_gen() for frequency analysis with PreDAC (before was used the 4-th channel of the predac internally)

1.4.3 1.8-16

- Added [setBPMoffsetEnbox\(\)](#): to be used to set the BPM offset in [um] only with BEST-ENC

1.4.4 1.8-15

- Added [getEnboxes\(\)](#) [getEnboxesNumber\(\)](#) functions

1.4.5 1.8-14

- Added getTetramms function

1.4.6 1.8-13

- Replaced getBPMscaling_sel with getBPMscale in getPosArray_sel and getFFT

1.4.7 1.8-12

- Compatibility with Best firmware $\geq 1.2.12$
- Fixed set/get PidParamSingle

1.4.8 1.8-11

- Replaced get getBPMscaling with getBPMscale in getPosStats
- Added function getTetrammsNumber

1.4.9 1.8-10

- Removed setROC function
- Added scaling to set/getBPMoffset and set/getROClimits

1.4.10 1.8-9

- Added [getWindAvg\(\)](#) -> read PID update frequency from FPGA

1.4.11 1.8-8

- Added comments and documentation
- Added [setSetpoint2\(\)](#) -> set new setpoint reading scaling paramenters from FPGA
- Added [getSetpoint2\(\)](#) -> get new setpoint reading scaling paramenters from FPGA
- Added [getBPMoffset\(\)](#) -> read BPM offset from FPGA

1.4.12 1.8-7

- Added [getROClimits\(\)](#) -> read ROC limits from FPGA

1.4.13 1.8-6

- Added [getCrossBar\(\)](#) -> read Crossbar configuration from FPGA

1.4.14 1.8-5

- fixed [getBPMselector\(\)](#) -> read BPM selector configuration from FPGA

1.5 Funtion List

1.5.1 Data functions

List of all functions that access real-time data.

1.5.1.1 `getBuffer`

1.5.1.2 `getPosArray_sel`

1.5.1.3 `getPosArray`

1.5.1.4 `getVoltageArray`

1.5.1.5 `getVoltageArray_sel`

1.5.1.6 `getCurrentArray`

1.5.1.7 `getCurrentArray_sel`

1.5.1.8 `getFFT`

1.5.1.9 `getPosStats`

1.5.1.10 `setDisplayFreq`

1.5.1.11 `getDisplayFreq`

1.5.1.12 `getBPMStats`

1.5.2 Configuration functions

List of all configuration functions.

1.5.2.1 `setOffsetSingle`

write single PID offset to FPGA (level 2)

1.5.2.2 `getOffsetSingle`

read single PID offset from FPGA (no level)

1.5.2.3 `setOffset`

write all 3 PID offsets to FPGA (level 2)

1.5.2.4 `setWindAvg`

write average window samples of specific PID to FPGA (level 2)

1.5.2.5 `getWindAvg` (added in v1.8-9)

read average window samples of specific PID from FPGA (no level)

1.5.2.6 `setPIDparams`

write all the PID parameters of a specific PID to FPGA (level 2)

1.5.2.7 `setPIDparamSingle`

write single PID parameter of a specific PID to FPGA (level 2)

1.5.2.8 `getPIDparamSingle`

read single PID parameter of a specific PID from FPGA (no level)

1.5.2.9 `setPIDoutSel`

write PID output channel selector to FPGA (level 2) select which PID is assigned to which PreDAC output channel.

1.5.2.10 `getPIDoutSel`

read PID output channel selector from FPGA (no level)

1.5.2.11 setCrossBar

write TetrAMM crossbar configuration for BPM0 or BPM1 to FPGA (level 2)

1.5.2.12 getCrossBar (1.8-6)

read TetrAMM crossbar configuration for BPM0 or BPM1 from FPGA (no level)

1.5.2.13 setBPMorient

write BPM orientation for BPM0 or BPM1 to FPGA (level 2)

1.5.2.14 getBPMorient

read BPM orientation for BPM0 or BPM1 from FPGA (no level)

1.5.2.15 setOutputMux

write Output Mux configuration to FPGA (level 1, 2)

1.5.2.16 getOutputMux

read Output Mux configuration from FPGA (no level)

1.5.2.17 setBPMscale

write BPM0 or BPM1 scaling parameters to FPGA (level 2)

1.5.2.18 getBPMscale

read BPM0 or BPM1 scaling parameters from FPGA (no level)

1.5.2.19 getBPMscaling_sel

read BPM0 or BPM1 scaling parameters from CONFIG_FILE (no level)

1.5.2.20 getBPMscaling

read BPM0 scaling parameters from CONFIG_FILE (no level)

1.5.2.21 setBPMoffset

write offsetX and offsetY of BPM0 or BPM1 to FPGA (level 2)

1.5.2.22 getBPMoffset (1.8-8)

read offsetX and offsetY of BPM0 or BPM1 from FPGA (no level)

1.5.2.23 getBPMselector (1.8-5)

read BPM selector configuration from FPGA (no level)

1.5.2.24 setBPMselector

write BPM selector configuration to FPGA (no level)!!!

1.5.2.25 getROCandROI

read RoiX, RoiY, Roi0max, Roi0min and Roc of BPM0 from CONFIG_FILE (no level)

1.5.2.26 getROCenable

read ROC configuration of BPM0 and BPM1 from FPGA (no level)

1.5.2.27 setROCenable

write ROC configuration of BPM0 and BPM1 to FPGA (level 2)

1.5.2.28 setROC

write min and max ROC levels of specific axis to FPGA (level 2)

1.5.2.29 setROClimits

write min and max ROC levels of specific axis to FPGA (no level)

1.5.2.30 getROClimits (1.8-7)

read min and max ROC levels of specific axis from FPGA (no level)

1.5.2.31 getROCandROI_sel (TODO)

read RoiX, RoiY, Roi0max, Roi0min and Roc of BPM0 or BPM1 from CONFIG_FILE (no level)

1.5.3 Access control functions

List of all control functions.

1.5.3.1 getLock

Login function, inputs username and password, output login level

1.5.3.2 getLockStatus

returns the actual login level

1.5.4 PID control functions**1.5.4.1 getPIDstatus**

read feedback status from FPGA (no level)

1.5.4.2 setFBenable

enable/disable feedback. Write to FPGA (level 1, 2)

1.5.4.3 setCtrlReset

reset feedback controller. Write to FPGA (level 1, 2)

1.5.4.4 setPIDconf

write PID configuration to FPGA (level 1, 2)

1.5.4.5 getPIDconf

read PID configuration from FPGA (no level)

1.5.4.6 setSetpoint

reads RoiX, RoiY, Roi0max, Roi0min and Roc of BPM0 from CONFIG_FILE, read BPM-selected scaling parameters (from CONFIG_FILE), divides the user setpoint by the scaling parameters (from CONFIG_FILE), sends the unscaled-setpoint to the FPGA (level 1, 2)

1.5.4.7 getSetpoint

reads the setpoint from FPGA, reads RoiX, RoiY, RoiI0max, RoiI0min and Roc of BPM0 from CONFIG_FILE, read BPM-selected scaling parameters (from CONFIG_FILE), multiplied the setpoint by the scaling parameters returns the scaled-setpoints to user space

1.5.4.8 setSetpoint2 (1.8-8)

read BPM selector configuration from FPGA, read BPM-selected scaling parameters (from FPGA), divides the user setpoint by the scaling parameters, sends the unscaled-setpoint to the FPGA (level 1, 2)

1.5.4.9 getSetpoint2 (1.8-8)

reads the setpoint from FPGA, read BPM selector configuration from FPGA, read BPM-selected scaling parameters (from FPGA), multiplied the setpoint by the scaling parameters returns the scaled-setpoints to user space

1.5.4.10 IIRcontrollerSetParam

write IIR coefficients to FPGA temporarily (level 2)

1.5.4.11 IIRcontrollerCommitParams

apply IIR coefficients written to FPGA (level 2)

1.5.4.12 IIRcontrollerGetParam

read IIR coefficients from FPGA (no level)

1.5.5 Function gen control functions

List of all functions to generate sinusoidal signal with PreDAC.

1.5.5.1 funcGenConfig

1.5.5.2 funcGenEnable

1.5.6 TetrAMM functions

List of all functions to communicate with TetrAMM.

1.5.6.1 tetrammHVSetVoltage

1.5.6.2 tetrammSetRange

1.5.6.3 tetrammHVenable

1.5.7 EnBOX functions

List of all functions to communicate with EnBOX.

1.5.7.1 enboxEncoderSelect

1.5.7.2 enboxMathSelect

1.5.8 Misc functions

1.5.8.1 getSystemVersion

1.5.8.2 getSFPinfo

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Data functions	15
Configuration functions	20
Access control functions	37
PID control functions	38
Function generator functions	43
TetrAMM functions	44
EnBOX functions	46
Misc	48

Chapter 3

Class Index

3.1 Class List

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

best_buffer_line	51
best_stats	51

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

best_c_interface.c	53
best_c_interface.h	56
best_c_interface_conf.cpp	61
best_c_interface_data.cpp	63
best_c_interface_tetramm.cpp	65

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 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 removedDC)
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

<i>stats</i>	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 (<code>offset_adc0_ch0 = 24</code> , <code>offset_pos0_posX = 0</code> , <code>offset_pos0_posY = 1</code> , <code>offset_pos0_I0 = 2</code> , <code>offset_pos1_posX = 3</code> , <code>offset_pos1_posY = 4</code> , <code>offset_pos1_I0 = 5</code> , <code>offset_dac0_ch0 = 16</code> , <code>offset_dac0_ch1 = 17</code> , <code>offset_dac0_ch2 = 18</code> , <code>offset_dac0_ch3 = 19</code>)
<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 * IO,
    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_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 `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 `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,0</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>IO</i>	BPM (= TetrAMM) for IO

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
-------------	----------------------

Parameters

<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>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

5.2.3.14 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,0</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 [μ m]
<i>scaleY</i>	in [μ 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 )
```

Sets input cross switch.

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,0</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,0</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 I0 )
```

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=Imax, 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 rocI0
<i>rocX2</i>	enable rocX2
<i>rocY2</i>	enable rocY2
<i>rocI02</i>	enable rocI02

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 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. 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()

```
unsigned 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,
    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 (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)

Parameters

<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.

5.4.2.8 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

5.4.2.10 setPIDconf()

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

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

Parameters

<i>conf</i>	PID conf (X = 0, XY = 1, XI0 = 2, Y = 3, YI0 = 4, XYI0 = 5, I0 = 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, 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

5.4.2.12 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()`

```
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. (>= 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> ↔ _A	on SFP A, 0 = No TetrAMM connected, 1 = TetrAMM connected
<i>pos</i> ↔ _A	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

<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 `getEnboxes()`

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

Gets Enbox. (>= 1.8-15).

Parameters

<i>pos↔ _A</i>	on SFP A, 0 = No Enbox connected, 1 = Enbox connected
<i>pos↔ _B</i>	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) #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. 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 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/CAENets/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 `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 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 l0)
 - 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 *l0)
 - 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 *l0)
 - Gets BPM sources. Read BPM sources from FPGA.*
 - int [setBPMselector](#) (int posX, int posY, int l0)
 - 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. 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 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. (>= 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[]
```

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 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](#)

7.3.1 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 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 [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.
- 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](#)

7.5.1 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`, 55
 - Data functions, 16
- Access control functions, 37
 - `getLock`, 37
 - `getLockStatus`, 37
- `best_buffer_line`, 51
- `best_c_interface.c`, 53
 - `__attribute__`, 55
 - `debug_print`, 54
 - `fd_DMA`, 56
 - `fd_Lock`, 56
 - `fd_Mbox`, 56
 - `getBestLock`, 55
 - in, 56
 - `LIB_BEST_DEBUG`, 55
 - `LIB_BEST_VERSION`, 56
 - `lock_sigaction`, 55
 - `loginLevel`, 56
 - out, 56
 - p, 56
 - `releaseBestLock`, 55
 - `SAMP_FREQ`, 56
 - `VERSION_STRING`, 55
 - `VERSION_STRING_HELPER`, 55
- `best_c_interface.h`, 56
 - `BEST_FILE_CONFIG_INI`, 60
 - `BEST_FILE_DISP_DMA`, 60
 - `BEST_FILE_MAILBOX`, 60
 - `FFT_LEN`, 60
 - `LEN_BBF`, 60
 - `LIB_BEST_VERSION`, 60
 - `LOCK_FILE`, 60
- `best_c_interface_conf.cpp`, 61
 - `fd_DMA`, 63
 - `fd_Mbox`, 63
 - in, 63
 - `loginLevel`, 63
 - out, 63
 - p, 63
 - `SAMP_FREQ`, 63
- `best_c_interface_data.cpp`, 63
 - `fd_DMA`, 65
 - `fd_Mbox`, 65
 - `ftw_abs`, 64
 - in, 65
 - `loginLevel`, 65
 - out, 65
 - p, 65
 - `SAMP_FREQ`, 65
- `best_c_interface_tetramm.cpp`, 65
 - `fd_Mbox`, 66
 - `loginLevel`, 66
- `BEST_FILE_CONFIG_INI`
 - `best_c_interface.h`, 60
- `BEST_FILE_DISP_DMA`
 - `best_c_interface.h`, 60
- `BEST_FILE_MAILBOX`
 - `best_c_interface.h`, 60
- `best_pid_select`
 - Configuration functions, 21
- `BEST_PID_SELECT_I0`
 - Configuration functions, 21
- `BEST_PID_SELECT_X`
 - Configuration functions, 21
- `BEST_PID_SELECT_Y`
 - Configuration functions, 21
- `best_stats`, 51
- Configuration functions, 20
 - `best_pid_select`, 21
 - `BEST_PID_SELECT_I0`, 21
 - `BEST_PID_SELECT_X`, 21
 - `BEST_PID_SELECT_Y`, 21
 - `getBPMoffset`, 22
 - `getBPMoffsetEnbox`, 22
 - `getBPMorient`, 22
 - `getBPMscale`, 23
 - `getBPMscaling`, 23
 - `getBPMscaling_sel`, 23
 - `getBPMselector`, 24
 - `getCrossBar`, 24
 - `getOffsetSingle`, 25
 - `getOutputMux`, 25
 - `getPIDoutSel`, 25
 - `getPIDparamSingle`, 26
 - `getROCandROI`, 26
 - `getROCanable`, 27
 - `getROClimits`, 27
 - `getWindAvg`, 27
 - `setBPMdevice`, 28
 - `setBPMoffset`, 28
 - `setBPMoffsetEnbox`, 29
 - `setBPMorient`, 29
 - `setBPMscale`, 29
 - `setBPMselector`, 31
 - `setCrossBar`, 31
 - `setEnboxSumDiff`, 32

- setEnboxType, 32
- setOffset, 32
- setOffsetSingle, 33
- setOutputMux, 33
- setPIDoutSel, 34
- setPIDparams, 34
- setPIDparamSingle, 35
- setROCEnable, 35
- setROClimits, 36
- setWindAvg, 36
- Data functions, 15
 - __attribute__, 16
 - getBPMStats, 16
 - getBuffer, 16
 - getCurrentArray, 16
 - getCurrentArray_sel, 17
 - getDisplayFreq, 17
 - getFFT, 17
 - getPosArray, 18
 - getPosArray_sel, 18
 - getPosStats, 18
 - getVoltageArray, 18
 - getVoltageArray_sel, 19
 - setDisplayFreq, 19
- debug_print
 - best_c_interface.c, 54
- EnBOX functions, 46
 - enboxEncoderSelect, 46
 - getEnboxes, 46
 - getEnboxesNumber, 46
- enboxEncoderSelect
 - EnBOX functions, 46
- fd_DMA
 - best_c_interface.c, 56
 - best_c_interface_conf.cpp, 63
 - best_c_interface_data.cpp, 65
- fd_Lock
 - best_c_interface.c, 56
- fd_Mbox
 - best_c_interface.c, 56
 - best_c_interface_conf.cpp, 63
 - best_c_interface_data.cpp, 65
 - best_c_interface_tetramm.cpp, 66
- FFT_LEN
 - best_c_interface.h, 60
- fftw_abs
 - best_c_interface_data.cpp, 64
- funcGenConfig
 - Function generator functions, 43
- funcGenEnable
 - Function generator functions, 43
- Function generator functions, 43
 - funcGenConfig, 43
 - funcGenEnable, 43
- getBestLock
 - best_c_interface.c, 55
- getBPMoffset
 - Configuration functions, 22
- getBPMoffsetEnbox
 - Configuration functions, 22
- getBPMorient
 - Configuration functions, 22
- getBPMscale
 - Configuration functions, 23
- getBPMscaling
 - Configuration functions, 23
- getBPMscaling_sel
 - Configuration functions, 23
- getBPMselector
 - Configuration functions, 24
- getBPMStats
 - Data functions, 16
- getBuffer
 - Data functions, 16
- getCrossBar
 - Configuration functions, 24
- getCurrentArray
 - Data functions, 16
- getCurrentArray_sel
 - Data functions, 17
- getDisplayFreq
 - Data functions, 17
- getEnboxes
 - EnBOX functions, 46
- getEnboxesNumber
 - EnBOX functions, 46
- getFFT
 - Data functions, 17
- getLock
 - Access control functions, 37
- getLockStatus
 - Access control functions, 37
- getOffsetSingle
 - Configuration functions, 25
- getOutputMux
 - Configuration functions, 25
- getPIDconf
 - PID control functions, 38
- getPIDoutSel
 - Configuration functions, 25
- getPIDparamSingle
 - Configuration functions, 26
- getPIDstatus
 - PID control functions, 39
- getPosArray
 - Data functions, 18
- getPosArray_sel
 - Data functions, 18
- getPosStats
 - Data functions, 18
- getROCandROI
 - Configuration functions, 26
- getROCEnable

- Configuration functions, [27](#)
- getROClimits
 - Configuration functions, [27](#)
- getSetpoint
 - PID control functions, [39](#)
- getSetpoint2
 - PID control functions, [39](#)
- getSFPinfo
 - Misc, [48](#)
- getSystemVersion
 - Misc, [48](#)
- getTetramms
 - TetrAMM functions, [44](#)
- getTetrammsNumber
 - TetrAMM functions, [44](#)
- getVoltageArray
 - Data functions, [18](#)
- getVoltageArray_sel
 - Data functions, [19](#)
- getWindAvg
 - Configuration functions, [27](#)
- IIRcontrollerCommitParams
 - PID control functions, [40](#)
- IIRcontrollerGetParam
 - PID control functions, [40](#)
- IIRcontrollerSetParam
 - PID control functions, [40](#)
- in
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface_conf.cpp](#), [63](#)
 - [best_c_interface_data.cpp](#), [65](#)
- LEN_BBF
 - [best_c_interface.h](#), [60](#)
- LIB_BEST_DEBUG
 - [best_c_interface.c](#), [55](#)
- LIB_BEST_VERSION
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface.h](#), [60](#)
- LOCK_FILE
 - [best_c_interface.h](#), [60](#)
- lock_sigaction
 - [best_c_interface.c](#), [55](#)
- loginLevel
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface_conf.cpp](#), [63](#)
 - [best_c_interface_data.cpp](#), [65](#)
 - [best_c_interface_tetramm.cpp](#), [66](#)
- Misc, [48](#)
 - [getSFPinfo](#), [48](#)
 - [getSystemVersion](#), [48](#)
- out
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface_conf.cpp](#), [63](#)
 - [best_c_interface_data.cpp](#), [65](#)
- p
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface_conf.cpp](#), [63](#)
 - [best_c_interface_data.cpp](#), [65](#)
- PID control functions, [38](#)
 - [getPIDconf](#), [38](#)
 - [getPIDstatus](#), [39](#)
 - [getSetpoint](#), [39](#)
 - [getSetpoint2](#), [39](#)
 - [IIRcontrollerCommitParams](#), [40](#)
 - [IIRcontrollerGetParam](#), [40](#)
 - [IIRcontrollerSetParam](#), [40](#)
 - [setCtrlReset](#), [41](#)
 - [setFBenable](#), [41](#)
 - [setPIDconf](#), [41](#)
 - [setSetpoint](#), [41](#)
 - [setSetpoint2](#), [42](#)
- README.md, [66](#)
- releaseBestLock
 - [best_c_interface.c](#), [55](#)
- SAMP_FREQ
 - [best_c_interface.c](#), [56](#)
 - [best_c_interface_conf.cpp](#), [63](#)
 - [best_c_interface_data.cpp](#), [65](#)
- setBPMdevice
 - Configuration functions, [28](#)
- setBPMoffset
 - Configuration functions, [28](#)
- setBPMoffsetEnbox
 - Configuration functions, [29](#)
- setBPMorient
 - Configuration functions, [29](#)
- setBPMscale
 - Configuration functions, [29](#)
- setBPMselector
 - Configuration functions, [31](#)
- setCrossBar
 - Configuration functions, [31](#)
- setCtrlReset
 - PID control functions, [41](#)
- setDisplayFreq
 - Data functions, [19](#)
- setEnboxSumDiff
 - Configuration functions, [32](#)
- setEnboxType
 - Configuration functions, [32](#)
- setFBenable
 - PID control functions, [41](#)
- setOffset
 - Configuration functions, [32](#)
- setOffsetSingle
 - Configuration functions, [33](#)
- setOutputMux
 - Configuration functions, [33](#)
- setPIDconf
 - PID control functions, [41](#)
- setPIDoutSel
 - Configuration functions, [34](#)

- setPIDparams
 - Configuration functions, [34](#)
- setPIDparamSingle
 - Configuration functions, [35](#)
- setROCenable
 - Configuration functions, [35](#)
- setROClimits
 - Configuration functions, [36](#)
- setSetpoint
 - PID control functions, [41](#)
- setSetpoint2
 - PID control functions, [42](#)
- setWindAvg
 - Configuration functions, [36](#)

- TetrAMM functions, [44](#)
 - getTetramms, [44](#)
 - getTetrammsNumber, [44](#)
 - tetrammHVenable, [44](#)
 - tetrammHVSetVoltage, [45](#)
 - tetrammSetRange, [45](#)
- tetrammHVenable
 - TetrAMM functions, [44](#)
- tetrammHVSetVoltage
 - TetrAMM functions, [45](#)
- tetrammSetRange
 - TetrAMM functions, [45](#)

- VERSION_STRING
 - best_c_interface.c, [55](#)
- VERSION_STRING_HELPER
 - best_c_interface.c, [55](#)