BIBEST Interface

Generated by Doxygen 1.8.17

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 Data functions

Functions which read measurement and calulcation 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 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)

  Calcualtes 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 calulcation 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

stats	in best_stats 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

length	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

length	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

sel	TetrAMM selector (0 or 1)
length	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 )
```

Calcualtes FFT.

Parameters

selector	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
outM	magnitude output (size FFT_LEN / 2)
outF	frequency output (size FFT_LEN / 2)
removeDC	removes DC component

5.1.2.9 getPosArray()

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

Same as getPosArray_sel, but only for 1st TetrAMM.

5.1.2.10 getPosArray_sel()

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

Gets arrays of postions and intensity from display DMA.

Parameters

selector	Selects BPM (= TetrAMM), should be 0 or 1
posX	Pointer to array where position X will be written
posY	Pointer to array where position Y will be written
I0	Pointer to array where Intensity will be written
length	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 * meanI0,
            double * stdI0 )
```

Get position statistics.

Parameters

selector	Selects which TetrAmm

5.1.2.12 getVoltageArray()

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

Gets arrays of voltages from display DMA.

Parameters

length	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

sel	PreDAC selector (0, for the time beign)
length	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

freq	Frequency in hertz
use_filter	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 ∗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 ∗rocI0, bool ∗rocX2, bool ∗rocY2, bool ∗rocI02)

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

offsetX	in [um]
offsetY	in [um]
bpm	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

sel	Selects TetrAmm (in case of 2 TetrAmms)
is90deg	(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

selector	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

scaleX	scaling value for X in [um]
scaleY	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

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

posXlf	BPM (= TetrAMM) for posXlf
posXhf	BPM (= TetrAMM) for posXhf
posYlf	BPM (= TetrAMM) for posYlf
posYhf	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

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

pid_sel	PID select enum
offset	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

sel	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

posXlf	DAC channel for posXlf
posXhf	DAC channel for posXhf
posYlf	DAC channel for posYlf
posYhf	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

pid_sel	(0 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
pid_par	(0=Kp, 1=Ki, 2=Kd, 3=emin, 4=Imax, 5=Omin, 6=Omax, 7=Ogain)
param_value	

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

roiX	double parameter in [um]
roiY	double parameter in [um]
roiI0min	double parameter in [A]
roiI0max	double parameter in [A]
roc	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

rocX	enable rocX
rocY	enable rocY
rocI0	enable rocI0
rocX2	enable rocX2
rocY2	enable rocY2
rocI02	enable rocI02

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

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

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

sel	(0 = X, 1 = Y, 2 = I0)
nr_samp	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

offsetX	in [um]
offsetY	in [um]
bpm	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

sel	Selects TetrAmm (in case of 2 TetrAmms)
is90deg	

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

scaleX	in [um]
scaleY	in [um]
bpm	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

posXlf	BPM (= TetrAMM) for posXlf
posXhf	BPM (= TetrAMM) for posXhf
posYlf	BPM (= TetrAMM) for posYlf
posYhf	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

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

offsetXlf	in [V]
offsetXhf	in [V]
offsetYlf	in [V]
offsetYhf	in [V]

Returns

0 on success, negative on fail

Note

Access level must be admin

5.2.3.22 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

pid_sel	PID select enum
offset	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

sel	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

posXlf	DAC channel for posXlf
posXhf	DAC channel for posXhf
posYlf	DAC channel for posYlf
posYhf	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

pid_sel	(0 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
Kp	See BEST User's Manual for detailed explanation
Ki	See BEST User's Manual for detailed explanation
Kd	See BEST User's Manual for detailed explanation
emin	See BEST User's Manual for detailed explanation
Imax	See BEST User's Manual for detailed explanation
Omin	See BEST User's Manual for detailed explanation
Omax	See BEST User's Manual for detailed explanation
Ogain	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

pid_sel	(0 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
pid_par	(0=Kp, 1=Ki, 2=Kd, 3=emin, 4=Imax, 5=Omin, 6=Omax, 7=Ogain)
param_value	

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

rocX	enable rocX
rocY	enable rocY
rocI0	enable rocI0
rocX2	enable rocX2
rocY2	enable rocY2
rocI02	enable rocI02

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

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

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

sel	(0 = Xlf, 1 = Xhf, 2 = Ylf, 3 = Yhf)
nr_samp	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

usr	Username
pass	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

filt_sel	Selects filter (0 = lowpass, 1 = band pass)
pos_sel	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

filt_sel	Selects filter (0 = lowpass, 1 = band pass)
pos_sel	Selects position (0 = X, 1 = Y)
param_is↩ _a	Selects if param read is a (when 1) or b (when 0)
coef_sel	Selects coeff (0-4)
coef	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

filt_sel	0 = Low Pass
pos_sel	Selects position (0 = X, 1 = Y)
enable	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_lf and Y_lf

5.4.2.4 getPIDconf()

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

Sets PID controller configuration. Read PID configuration from FPGA.

Parameters

conf	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

sel	Selects which PID (Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3)
setpt	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

sel	Selects which PID (Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3)
setpt	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

ctrlr_sel	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

ctrlr_sel	Selects controller (X = 0, Y = 1, I0 = 2)
param_is↩ _a	Selects if param read is a (when 1) or b (when 0)
param_sel	Selects parameter (0-9)
param	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

ctrlr_sel	Selects controller (X = 0, Y = 1, I0 = 2)
param_is↩ _a	Selects if param written is a (when 1) or b (when 0)
param_sel	Selects parameter (0-9)
param	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

filt_sel	Selects filter (0 = lowpass, 1 = band pass)
pos_sel	Selects position (0 = X, 1 = Y)
param_is↩ _a	Selects if param read is a (when 1) or b (when 0)
coef_sel	Selects coeff (0-4)
coef	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

filt_sel	0 = Low Pass
pos_sel	Selects position (0 = X, 1 = Y)
enable	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_lf and Y_lf

5.4.2.15 setPIDconf()

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

Sets PID controller configuration.

Parameters

conf	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

sel	Selects which PID (Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3)
setpt	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

sel	Selects which PID (Xlf = 0, Xhf = 1, Ylf = 2, Yhf = 3)
setpt	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

freq	Frequency (in Hz)
ampl	Amplitude of sine wave
offset	Offset of sine wave

Returns

0 on success, negative on fail

configuration is commited 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

enable	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

pos↵_A	on SFP A, 0 = No TetrAMM connected, 1 = TetrAMM connected
pos↵_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

enable	1 to enable, 0 to disable
sel	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

voltage	voltage in volts
sel	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

range	0 = RNG0, 1 = RNG1
selector	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

enc_type	0 = Tonic, 1 = Absolute
selector	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

math_func	0 = normal, 1 = sum-diff between encoder0 and encoder1 on the same bpm
selector	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

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

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

hwVer	hardware version
swVer	software version (main application)
ctrlVer	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, "[libbibest] " 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, "[libbibest] " 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)

  Calcualtes 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 ∗rocI0, bool ∗rocX2, bool ∗rocY2, bool ∗rocI02)

  Gets enables for different ROC checks of 1st nad 2nd BPMs. Read values from FPGA.
- 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.
- 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 Imax, 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 ampl[4], float offset[4])

  Configures function generator.
- int funcGenEnable (char enable)

  Enables or disables function generator.
- int tetrammSetRange (int range, unsigned char selector)

  Sets front-end range.
- int tetrammHVSetVoltage (float voltage, unsigned char sel)

  Sets TetrAMM HV voltage.
- int tetrammHVenable (int enable, unsigned char sel)

  Enables or disables TetrAMM HV module.
- unsigned int getTetrammsNumber ()

  Gets number of TetrAMMs.
- int getTetramms (int ∗pos_A, int ∗pos_B)

  Gets TetrAMMs.
- 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"
```

7.2.2 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 ∗rocI0, bool ∗rocX2, bool ∗rocY2, bool ∗rocI02)

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

  Calcualtes 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