BEST Interface

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

Tue Jun 23 2026 12:20:27

Chapter 1

[2.0.0] - 2026/06/15

1.0.1 ADDED

- [EXT] First release for Ubuntu 24.04

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

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

Topic Documentation

5.1 Data functions

Classes

- struct best_buffer_line
- struct best_stats

Functions

- struct __attribute__ ((packed)) best_buffer_line
- int getBuffer (struct best_buffer_line[ ], int length)

  Gets arrays of "samples" from display DMA.
- int getPosArray_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosScalar_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosArray (double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Same as getPosArray_sel, but only for 1st TetrAMM.
- int getVoltageArray (double(∗voltages)[4], unsigned int length)

  Gets arrays of voltages from display DMA.
- int getVoltageArray_sel (int sel, double(∗voltages)[4], unsigned int length)

  Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).
- int getVoltageScalar_sel (int sel, double ∗ch0, double ∗ch1, double ∗ch2, double ∗ch3, unsigned int length)

  Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for 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 ∗stdX_filt, double ∗meanY, double ∗stdY, double ∗stdY_filt, double ∗meanI0, double ∗stdI0, double ∗stdI0_filt)

  Get position statistics.
- int setDisplayFreq (int freq, uint8_t use_filter)

  Sets display frequency.
- int getDisplayFreq (int ∗freq, uint8_t ∗use_filter)

  Gets display frequency.
- struct __attribute__ ((__packed__)) best_stats
- int getBPMStats (best_stats ∗stats)

  Gets BPM Statistics.

5.1.1 Detailed Description

Functions which read measurement and 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 buf[],
            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()

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 getPosScalar_sel()

```
int getPosScalar_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 on which to perform the average

Returns

numbers of samples copies (should be equal to length)

5.1.2.12 getPosStats()

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

Get position statistics.

Parameters

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

5.1.2.13 getVoltageArray()

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

Gets arrays of voltages from display DMA.

Parameters

length	Length of the arrays

Returns

numbers of samples copies (should be equal to length)

5.1.2.14 getVoltageArray_sel()

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

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

Parameters

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

5.1.2.15 getVoltageScalar_sel()

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

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

Parameters

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

Returns

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

5.1.2.16 setDisplayFreq()

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

Sets display frequency.

Parameters

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

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 ∗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 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 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 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.

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

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 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

offsetX	in [um]
offsetY	in [um]
bpm	Selects BPM (= TetrAMM), should be 0 or 1
encType,-	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

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

5.2.3.4 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

selector	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

scaleX	scaling value for X in [um]
scaleY	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

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

5.2.3.7 getBPMselector()

```
int getBPMselector (
            int * posX,
            int * posY,
            int * I0)
```

Gets BPM sources. Read BPM sources from FPGA.

Parameters

posX	BPM (= TetrAMM) for posX
posY	BPM (= TetrAMM) for posY
I0	BPM (= TetrAMM) for I0

Returns

ACK

Note

Access level: all levels

5.2.3.8 getCrossBar()

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

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

Parameters

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

5.2.3.9 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

5.2.3.10 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

5.2.3.11 getPIDoutSel()

```
int getPIDoutSel (
            uint8_t * posX,
            uint8_t * posY,
            uint8_t * I0)
```

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

Parameters

posX	DAC channel for posX

posY	DAC channel for posY
I0	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

pid_sel	(0 = X, 1 = Y, 2 = I0)
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

5.2.3.13 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

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

5.2.3.14 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

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

5.2.3.15 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

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

Returns

0 on success, negative on fail

5.2.3.16 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

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

sel	Selects the BPM
isEnbox	

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

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

5.2.3.19 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

offsetX	in [um]
offsetY	in [um]
bpm	Selects BPM (= Enbox), should be 0 or 1
encType,-	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

sel	Selects TetrAmm (in case of 2 TetrAmms)
is90deg	

Returns

0 on success, negative on fail

Note

Access level must be admin

5.2.3.21 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

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

5.2.3.22 setBPMselector()

```
int setBPMselector (
            int posX,
            int posY,
            int I0)
```

Sets BPM sources. Write BPM sources to FPGA.

Parameters

posX	BPM (= TetrAMM) for posX
posY	BPM (= TetrAMM) for posY
I0	BPM (= TetrAMM) for I0

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

sel	Selects TetrAmm (in case of 2 TetrAmms)
ch0	
ch1	
ch2	
ch3	

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

sel	Selects the BPM
encType,-	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

sel	Selects the BPM
encType,-	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 offsetI0)
```

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

Parameters

offsetX	in [V]
offsetY	in [V]
offset↩ I0	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

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

5.2.3.28 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

5.2.3.29 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

posX	DAC channel for posX
posY	DAC channel for posY
I0	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

pid_sel	(0 = X, 1 = Y, 2 = I0)
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

5.2.3.31 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

pid_sel	(0 = X, 1 = Y, 2 = I0)
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

5.2.3.32 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

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

5.2.3.33 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

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

5.2.3.34 setWindAvg()

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

Sets number of samples for window averaging.

Parameters

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

Returns

0 on success, negative on fail

Note

Access level must be admin

5.3 Access control functions

Functions

- int getLock (const char ∗usr, const char ∗pass)

  Acquires lock on /var/lock/best_lock.
- int getLockStatus ()

  Gets login level, which can be changed with call on getLock().

5.3.1 Detailed Description

Functions which allow to elevate user access level

5.3.2 Function Documentation

5.3.2.1 getLock()

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

Acquires lock on /var/lock/best_lock.

Parameters

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

- 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

conf	return PID conf

Returns

0 on success, negative on fail

5.4.2.2 getPIDstatus()

```
int getPIDstatus (
            void )
```

Gets PID status.

Returns

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

Note

Access level: all levels

5.4.2.3 getSetpoint()

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

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

Parameters

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

Returns

0 on success, negative on fail

Note

Access level: all levels

5.4.2.4 getSetpoint2()


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

Parameters

sel	Selects which PID (X = 0, Y = 1, I0 = 2)
setpt	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

ctrlr_sel	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

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).

5.4.2.7 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.

5.4.2.8 setCtrlReset()

```
int setCtrlReset ()
```

Resets internal states of controller.

Returns

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

Note

Access level must be user or admin

5.4.2.9 setFBenable()

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

Enables or disables feedback.

Returns

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

Note

Access level must be user or admin

Parameters

conf	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

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

Returns

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

Note

Access level must be user or admin

5.4.2.12 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

sel	Selects which PID (X = 0, Y = 1, I0 = 2)
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

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. (>= 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

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)
- 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

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 getEnboxes()

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

Gets Enbox. (>= 1.8-15).

Parameters

pos_A	on SFP A, 0 = No Enbox connected, 1 = Enbox connected
pos_A	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

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)
- #define VERSION_STRING(X)
- #define LIB_BEST_DEBUG 0
- #define debug_print(fmt, ...)

Functions

- int getBestLock (int fd)

  !!! All sets should check loginLevel
- int releaseBestLock (int fd)
- void lock_sigaction (int sig, siginfo_t *siginfo, void *context)
- void __attribute__ ((constructor))
- void __attribute__ ((destructor))

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

  Acquires lock on /var/lock/best_lock.
- int getLockStatus ()

  Gets login level, which can be changed with call on getLock().
- int setFBenable (int enable)

  Enables or disables feedback.
- int setCtrlReset ()

  Resets internal states of controller.
- int setPIDconf (char conf)

  Sets PID controller configuration. Write PID controller configuration to FPGA.
- int getPIDconf (char *conf)

  Sets PID controller configuration. Read PID configuration from FPGA.
- int setSetpoint (char sel, double setpt)

  Sets new setpoint in [um]. Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from CONFIG_FILE.
- int setSetpoint2 (char sel, double setpt)

  Sets new setpoint (without using config file). Write setpoint to FPGA. Before writing to FPGA the setpoint is divided by scaling parameter of 1st BPM. Scaling parameters are read from FPGA.
- int getSetpoint (char sel, double *setpt)

  Gets setpoint in [um]. Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from CONFIG_FILE.
- int getSetpoint2 (char sel, double *setpt)

  Gets new setpoint (without using config file). Read setpoint from FPGA. After reading from FPGA the setpoint is multiplied by scaling parameter. Scaling parameters are read from FPGA.
- int IIRcontrollerSetParam (char ctrlr_sel, char param_is_a, char param_sel, double param)

  Sets IIR controller parameter.
- int IIRcontrollerCommitParams (char ctrlr_sel)

  Commits IIR controller parameters.
- int IIRcontrollerGetParam (char ctrlr_sel, char param_is_a, char param_sel, double *param)

  Reads IIR controller parameter.
- int funcGenConfig (float freq, float 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,
                ...)
```

Value:

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

7.1.1.2 LIB_BEST_DEBUG

```
#define LIB_BEST_DEBUG 0
```

7.1.1.3 VERSION_STRING

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

Value:

```
VERSION_STRING_HELPER(X)
```

7.1.1.4 VERSION_STRING_HELPER

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

Value:

```
#X
```

7.1.2 Function Documentation

7.1.2.1 __attribute__() [1/2]

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

7.1.2.2 __attribute__() [2/2]

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

7.1.2.3 getBestLock()

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

!!! All sets should check loginLevel

7.1.2.4 lock_sigaction()

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

7.1.2.5 releaseBestLock()

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

7.1.3 Variable Documentation

7.1.3.1 fd_DMA

```
int fd_DMA
```

7.1.3.2 fd_Lock

```
int fd_Lock
```

7.1.3.3 fd_Mbox

```
int fd_Mbox
```

7.1.3.4 in

```
double* in
```

7.1.3.5 LIB_BEST_VERSION

```
char LIB_BEST_VERSION[] = VERSION_STRING(FROM_DEFS_VERSION)
```

7.1.3.6 loginLevel

```
int loginLevel = 0
```

7.1.3.7 out

```
fftw_complex* out
```

7.1.3.8 p

```
fftw_plan p
```

7.1.3.9 SAMP_FREQ

```
int SAMP_FREQ = 1000
```

7.2 best_c_interface.h File Reference

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

Macros

- #define BEST_FILE_MAILBOX "/dev/best_mailbox"
- #define BEST_FILE_DISP_DMA "/dev/best_dma_displ"
- #define LOCK_FILE "/var/lock/best_lock"
- #define BEST_FILE_CONFIG_INI "/opt/CAENels/BEST/config.ini"
- #define FFT_LEN (2048)
- #define LEN_BBF (32)

Enumerations

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

Functions

- struct __attribute__ ((packed)) best_buffer_line
- int getBuffer (struct best_buffer_line[ ], int length)

  Gets arrays of "samples" from display DMA.
- int getPosArray_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosScalar_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosArray (double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Same as getPosArray_sel, but only for 1st TetrAMM.
- int getVoltageArray (double(∗voltages)[4], unsigned int length)

  Gets arrays of voltages from display DMA.
- int getVoltageArray_sel (int sel, double(∗voltages)[4], unsigned int length)

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

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

  Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for 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 *stdX_filt, double *meanY, double *stdY, double *stdY_filt, double *meanI0, double *stdI0, double *stdI0_filt)

  Get position statistics.
- int setDisplayFreq (int freq, uint8_t use_filter)

  Sets display frequency.
- int getDisplayFreq (int *freq, uint8_t *use_filter)

  Gets display frequency.
- struct __attribute__ ((__packed__)) best_stats
- int getBPMStats (best_stats *stats)

  Gets BPM Statistics.
- int getBPMscaling_sel (int selector, double *scaleX, double *scaleY)

  Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG_FILE.
- int getBPMscaling (double *scaleX, double *scaleY)

  Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG_FILE.
- int getROCandROI (double *roiX, double *roiY, double *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 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 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 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 *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().
- int getPIDstatus (void)

  Gets PID status.
- int setFBenable (int enable)

  Enables or disables feedback.
- int setCtrlReset ()

  Resets internal states of controller.
- int setPIDconf (char conf)

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

- int getPIDconf (char *conf)

  Sets PID controller configuration. Read PID configuration from FPGA.

- int setSetpoint (char sel, double setpt)

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

- int getSetpoint (char sel, double *setpt)

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

- int setSetpoint2 (char sel, double setpt)

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

- int getSetpoint2 (char sel, double *setpt)

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

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

  Sets IIR controller parameter.

- int IIRcontrollerCommitParams (char ctrlr_sel)

  Commits IIR controller parameters.

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

  Reads IIR controller parameter.

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

  Configures function generator.

- int funcGenEnable (char enable)

  Enables or disables function generator.

- int tetrammSetRange (int range, unsigned char selector)

  Sets front-end range.

- int tetrammHVSetVoltage (float voltage, unsigned char sel)

  Sets TetrAMM HV voltage.

- int tetrammHVenable (int enable, unsigned char sel)

  Enables or disables TetrAMM HV module.

- unsigned int getTetrammsNumber ()

  Gets number of TetrAMMs.

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

  Gets TetrAMMs. (>= 1.8-14).

- int enboxEncoderSelect (int enc_type, unsigned char selector)

- unsigned int getEnboxesNumber ()

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

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

  Gets Enbox. (>= 1.8-15).

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

  Gets SFP info.

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

  Gets System HW, SW and PID versions.

Variables

- char LIB_BEST_VERSION [ ]

7.2.1 Macro Definition Documentation

7.2.1.1 BEST_FILE_CONFIG_INI

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

7.2.1.2 BEST_FILE_DISP_DMA

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

7.2.1.3 BEST_FILE_MAILBOX

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

7.2.1.4 FFT_LEN

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

7.2.1.5 LEN_BBF

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

7.2.1.6 LOCK_FILE

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

7.2.2 Variable Documentation

7.2.2.1 LIB_BEST_VERSION

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

7.3 best_c_interface.h

Go to the documentation of this file.

```
00001
00002 #ifndef BEST_H_
00003 #define BEST_H_
00004
00005 #include <stdint.h>
00006
00007
00008 #define BEST_FILE_MAILBOX       "/dev/best_mailbox"
00009 #define BEST_FILE_DISP_DMA      "/dev/best_dma_displ"
00010 #define LOCK_FILE               "/var/lock/best_lock"
00011 #define BEST_FILE_CONFIG_INI    "/opt/CAENels/BEST/config.ini"
00012
00013 #define FFT_LEN                 (2048)
00014 #define LEN_BBF                 (32)        // sizeof(best_buffer_line)/sizeof(double)
00015
00016 #ifdef __cplusplus
00017 extern "C" {
00018 #endif /* __cplusplus */
00019
00020 extern char LIB_BEST_VERSION[];
00021
00022
00023 //============================================================================//
00024 //                                                                            //
00025 //                            Data functions                                  //
00026 //                                                                            //
00027 //============================================================================//
00028
00033
00035
00038 struct __attribute__((packed)) best_buffer_line {  //addresses with double pointer  *(double)
00039     double bpm0_posX;       // 0
00040     double bpm0_posY;       // 1
00041     double bpm0_i0;         // 2
00042     double bpm0_rsvd;       // 3
00043     double bpm1_posX;       // 4
00044     double bpm1_posY;       // 5
00045     double bpm1_i0;         // 6
00046     uint32_t sin;           // 7
00047     uint32_t cos;           // 7
00048     double dac[4];          // 8
00049     uint32_t rsvd0[8];      // 8+4=12
00050     uint32_t rsvd1[4];      // 12+8/2=16
00051     float setpts[3];        // 16+4/2=18
00052     uint32_t status;        // 19
00053     uint32_t rsvd2[7];      // 18+(3+1)/2=20
00054     uint32_t timestamp;     // 23
00055     double tetramm0[4];     // 20+(7+1)/2=24
00056     double tetramm1[4];     // 24+4=28
00057 };
00058
00063 int getBuffer(struct best_buffer_line[], int length);
00064
00073 int getPosArray_sel(
00074     int selector,
00075     double* posX,
00076     double* posY,
00077     double* I0,
00078     unsigned int length
00079 );
00080
00089 int getPosScalar_sel(
00090     int selector,
00091     double* posX,
00092     double* posY,
00093     double* I0,
00094     unsigned int length
00095 );
00096
00099 int getPosArray(double* posX, double* posY, double* I0, unsigned int length);
00100
00101
00106 int getVoltageArray(double (*voltages)[4], unsigned int length);
00107
00115 int getVoltageArray_sel(int sel, double (*voltages)[4], unsigned int length);
00116
00124 int getVoltageScalar_sel(int sel, double* ch0, double* ch1, double* ch2, double* ch3, unsigned int
      length);
00125
00130 int getCurrentArray(double (*currents)[4], unsigned int length);
00131
00138 int getCurrentArray_sel(int sel, double (*currents)[4], unsigned int length);
```

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

```
00364                     double emin,
00365                     double Imax,
00366                     double Omin,
00367                     double Omax,
00368                     double Ogain);
00369
00378 int setPIDparamSingle(char pid_sel, int pid_par, double param_value);
00379
00388 int getPIDparamSingle(char pid_sel, int pid_par, double* param_value);
00389
00398 int setPIDoutSel(uint8_t posX, uint8_t posY, uint8_t I0);
00399
00408 int getPIDoutSel(uint8_t *posX, uint8_t *posY, uint8_t *I0);
00409
00419 int setCrossBar(uint8_t sel, uint8_t ch0, uint8_t ch1, uint8_t ch2, uint8_t ch3);
00420
00427 int setBPMorient(uint8_t sel, uint8_t is90deg);
00428
00434 int setOutputMux(uint8_t sel);
00435
00441 int getOutputMux(uint8_t* sel);
00442
00451 int setBPMscale(double scaleX, double scaleY, int bpm);
00452
00459 int getBPMscale(int selector, double* scaleX, double* scaleY);
00460
00469 int setBPMoffset(double offsetX, double offsetY, int bpm);
00470
00480 int setBPMoffsetEnbox(double offsetX, double offsetY, int bpm, int encType);
00481
00489 int setBPMdevice(uint8_t sel, uint8_t isEnbox);
00490
00498 int setEnboxType(uint8_t sel, uint8_t encType);
00499
00509 int setEnboxSumDiff(uint8_t sel, uint8_t isSumDiff);
00510
00518 int getBPMoffset(double *offsetX, double *offsetY, int bpm);
00519
00528 int getBPMoffsetEnbox(double *offsetX, double *offsetY, int bpm, int encType);
00529
00537 int getBPMorient(uint8_t sel, uint8_t* is90deg);
00538
00547 int getBPMselector(int *posX, int *posY, int *I0);
00548
00557 int setBPMselector(int posX, int posY, int I0);
00558
00568 int getCrossBar(uint8_t sel, uint8_t* ch0, uint8_t* ch1, uint8_t* ch2, uint8_t* ch3);
00569
00579 int setROClimits(int sel, double min, double max);
00580
00589 int getROClimits(int sel, double* min, double* max);
00590  // end of configFunc
00592
00593
00594
00595 //=============================================================================//
00596 //                                                                             //
00597 //                        Access control functions                             //
00598 //                                                                             //
00599 //=============================================================================//
00600
00605
00612 int getLock(const char *usr, const char *pass);
00613
00618 int getLockStatus();
00619  // end of loginFunc
00621
00622
00623 //=============================================================================//
00624 //                                                                             //
00625 //                          PID control functions                              //
00626 //                                                                             //
00627 //=============================================================================//
00628
00633
00638 int getPIDstatus(void);
00639
00644 int setFBenable(int enable);
00645
00650 int setCtrlReset();
00651
00658 int setPIDconf(char conf);
00659
00665 int getPIDconf(char* conf);
00666
00676 int setSetpoint(char sel, double setpt);
00677
```

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

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

7.4 best_c_interface_conf.cpp File Reference

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

Functions

- int getPIDstatus (void)

  Gets PID status.
- int getPosStats (int selector, double ∗meanX, double ∗stdX, double ∗stdX_filt, double ∗meanY, double ∗stdY, double ∗stdY_filt, double ∗meanI0, double ∗stdI0, double ∗stdI0_filt)

  Get position statistics.
- int getBPMscaling_sel (int selector, double ∗scaleX, double ∗scaleY)

  Gets BPM scaling parameters for position X and Y in [um]. Read values from CONFIG_FILE.
- int getBPMscaling (double ∗scaleX, double ∗scaleY)

  Gets BPM scaling parameters of 1st BPM for position X and Y in [um]. Read values from CONFIG_FILE.
- int getBPMscale (int selector, double ∗scaleX, double ∗scaleY)

  Gets BPM scaling parameters for position X and Y in [um]. Read BPM scaling parameters from FPGA.
- int getROCandROI (double ∗roiX, double ∗roiY, double ∗roiI0min, double ∗roiI0max, double ∗roc)

  Gets RoiX, RoiY, RoiI0max, RoiI0min and Roc of 1st BPM. Read values from CONFIG_FILE.
- int getROCenable (bool ∗rocX, bool ∗rocY, bool ∗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 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 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 posX, uint8_t posY, uint8_t I0)

  Sets PID output cross switch. Selects which PID is assigned to which PreDAC output channel. Write PID output cross switch to FPGA.
- int getPIDoutSel (uint8_t *posX, uint8_t *posY, uint8_t *I0)

  Gets PID output cross switch. Read PID output cross switch from FPGA.
- int setCrossBar (uint8_t sel, uint8_t ch0, uint8_t ch1, uint8_t ch2, uint8_t ch3)

  Sets input cross switch.
- int setBPMorient (uint8_t sel, uint8_t is90deg)

  Sets BPM orientation for diffOverSum.
- int getBPMorient (uint8_t sel, uint8_t *is90deg)

  Sets BPM orientation for diffOverSum. Read BPM orientation from FPGA.
- int setOutputMux (uint8_t sel)

  Sets value of output mux.
- int getOutputMux (uint8_t *sel)

  Gets value of output mux.
- int setBPMscale (double scaleX, double scaleY, int bpm)

  Sets BPM scaling parameters for position X and Y in [um]. Write BPM scaling parameters to FPGA.
- int setBPMoffset (double offsetX, double offsetY, int bpm)

  Sets BPM offset in [um] ONLY FOR TETRAMMs. Write BPM offsets to FPGA.
- int setBPMoffsetEnbox (double offsetX, double offsetY, int bpm, int encType)

  Sets BPM offset in [um] ONLY FOR Enboxes. Write BPM offsets to FPGA. (>= 1.8-16).
- int setBPMdevice (uint8_t sel, uint8_t isEnbox)

  Sets BPM device for diffOverSum. Write to FPGA. (>= 1.8-18).
- int setEnboxType (uint8_t sel, uint8_t encType)

  Sets the EnBOX type (relative or absolute) . Write to FPGA. (>= 1.8-18).
- int setEnboxSumDiff (uint8_t sel, uint8_t isSumDiff)

  Set diffOverSum calculations (for EnBOX). 0:Normal the positions are calculated from enc1 and enc2. 1:SumDiff the positions are calculated from enc1+enc2 and enc1-enc2. Write to FPGA. (>= 1.8-18).
- int getBPMoffset (double *offsetX, double *offsetY, int bpm)

  Gets BPM offsets in [um] ONLY FOR TETRAMMs. Read BPM offsets from FPGA.
- int getBPMoffsetEnbox (double *offsetX, double *offsetY, int bpm, int encType)

  Gets BPM offsets in [um] ONLY FOR Enboxes. Read BPM offsets from FPGA.
- int getBPMselector (int *posX, int *posY, int *I0)

  Gets BPM sources. Read BPM sources from FPGA.
- int setBPMselector (int posX, int posY, int I0)

  Sets BPM sources. Write BPM sources to FPGA.
- int getCrossBar (uint8_t sel, uint8_t *ch0, uint8_t *ch1, uint8_t *ch2, uint8_t *ch3)

  Gets input cross switch. Read input cross switch from FPGA.
- int setROClimits (int sel, double min, double max)

  Sets ROC min and max values. ROC should be in [um] for positions and in [A] for intensity. Write min and max ROC levels of specific axis to FPGA.
- int getROClimits (int sel, double *min, double *max)

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

Variables

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

7.4.1 Variable Documentation

7.4.1.1 fd_DMA

```
int fd_DMA  [extern]
```

7.4.1.2 fd_Mbox

```
int fd_Mbox  [extern]
```

7.4.1.3 in

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

7.4.1.4 loginLevel

```
int loginLevel  [extern]
```

7.4.1.5 out

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

7.4.1.6 p

```
fftw_plan p  [extern]
```

7.4.1.7 SAMP_FREQ

```
int SAMP_FREQ  [extern]
```

7.5 best_c_interface_data.cpp File Reference

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

Functions

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

  Gets arrays of "samples" from display DMA.
- int getPosArray_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosScalar_sel (int selector, double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Gets arrays of postions and intensity from display DMA.
- int getPosArray (double ∗posX, double ∗posY, double ∗I0, unsigned int length)

  Same as getPosArray_sel, but only for 1st TetrAMM.
- int getVoltageArray_sel (int sel, double(∗voltages)[4], unsigned int length)

  Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).
- int getVoltageArray (double(∗voltages)[4], unsigned int length)

  Gets arrays of voltages from display DMA.
- int getVoltageScalar_sel (int sel, double ∗ch0, double ∗ch1, double ∗ch2, double ∗ch3, unsigned int length)

  Gets arrays of voltages one of X PreDAC (the PreDAC is only 1 for for the time beign).
- int getCurrentArray_sel (int sel, double(∗currents)[4], unsigned int length)

  Gets arrays of currents one of two TetrAMMs.
- int getCurrentArray (double(∗currents)[4], unsigned int length)

  Gets arrays of currents from display DMA.
- double fftw_abs (fftw_complex f)
- int getFFT (unsigned int selector, double ∗outM, double ∗outF, int 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

7.5.1 Function Documentation

7.5.1.1 fftw_abs()

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

7.5.2 Variable Documentation

7.5.2.1 fd_DMA

```
int fd_DMA  [extern]
```

7.5.2.2 fd_Mbox

```
int fd_Mbox  [extern]
```

7.5.2.3 in

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

7.5.2.4 loginLevel

```
int loginLevel  [extern]
```

7.5.2.5 out

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

7.5.2.6 p

```
fftw_plan p  [extern]
```

7.5.2.7 SAMP_FREQ

```
int SAMP_FREQ  [extern]
```

7.6 best_c_interface_tetramm.cpp File Reference

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

Functions

- int tetrammSetRange (int range, unsigned char selector)

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

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

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

  Gets number of TetrAMMs.
- unsigned int getEnboxesNumber ()

  Gets number of Enbox. (>= 1.8-15).
- int getTetramms (int *pos_A, int *pos_B)

  Gets TetrAMMs. (>= 1.8-14).
- int getEnboxes (int *pos_A, int *pos_B)

  Gets Enbox. (>= 1.8-15).
- int enboxEncoderSelect (int enc_type, unsigned char selector)

Variables

- int loginLevel
- int fd_Mbox

7.6.1 Variable Documentation

7.6.1.1 fd_Mbox

```
int fd_Mbox  [extern]
```

7.6.1.2 loginLevel

```
int loginLevel  [extern]
```

7.7 changelog.md File Reference

Index