int ResetSSIMinAndMax(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function resets the value accumulated in SSI Max and SSI Min registers.

int ResetPhaseNoisePeak(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function resets the value accumulated in the Phase Noise Peak diagnostic register.

int ResetOptPowL2MinAndMax(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function resets the Opt Power L2 Min and Opt Power L2 Max registers. The Cmd Busy bit in Status Register 1 is set immediately and cleared when the registers have been reset.

int ResetSigRMSL2MinAndMax(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function resets the Sig RMS L2 Min and Sig RMS L2 Max registers. The Cmd Busy bit in Status Register 1 is set immediately and cleared when the registers have been reset.

SetKpAndKvCoeff

int SetKpAndKvCoeff(struct SIS1100\_Device\_Struct\* dev, unsigned short axis, unsigned short Kp, unsigned short Kv)

This function set Kp and Kv digital filter coefficients.

There’s a digital filter circuit on the Zygo Measurement Board 4104C designed to remove noise from the measurement data. It’s a second order digital filter, with two programmable gains, Kp and Kv, to permit a wide range of low pass filter cutoff frequencies, from 15 kHz to 2.7 MHz.

The Kv value for the digital filter is set to produce the best dynamic response for a given value of KP.

The Kp value for the digital filter primarily determines the bandwidth and response time of the digital filter.

Best Kp and Kv matched value can be found in the annexes.

EnableGlitchFilter

int EnableGlitchFilter(struct SIS1100\_Device\_Struct\* dev, unsigned short axis, unsigned short glitchFilterTime).

This function set the value of the glitch filter time ranged from 0 to 255.

The glitch filter controls suppression of glitches that would otherwise cause a loss of valid measurements.

When glitchFilterTime is set to 0 there is a filter time constant of 1.5 µs; when set to 1 the time constant is 3.8 µs.

GetVMEExtSampFlag

bool GetVMEExtSampFlag(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function polls the state of the VME External Sample flag bit. This is useful to know when the position sampling has come to end.

Return:

True if the flag is set

False otherwise

IsVMEIntReqPending

bool IsVMEIntReqPending(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function check whether there’s an interrupt request pending on VME Bus. If the VME Interrupts is enable bit, the interrupt is asserted on the VMEbus.

Return:

True if there’s

False otherwise

IsAPDCtrlSoftErrs

bool IsAPDCtrlSoftErrs(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function check whether there’s a software error raise by the APD controller.

Return:

True if there’s.

False otherwise

ReadAPDCtrlSoftErrs

int ReadAPDCtrlSoftErrs(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function reads software errors raised by APD controller.

There are 4 types of software errors as listed bellow:

Exception Error

Stack Overflow

Switch Default Error

Comm. Error

Divide by Zero Exception

All of the above errors are Fatal Error except for the Divide by Zero Exception and Comm Error. Fatal errors cause the APD Controller does the following:

Shut down High Voltage Supply.

Set fatal error on all axes.

The APD Controller moves to a Fatal Error State.

ResetPosition

int ResetPosition(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function resets or presets the position measurement. It resets the measurement function, but does not reset errors. If the Preset is enabled (using EnableReset function), the value in the Preset Position register is preset into the position measurement. The reset or preset value is before the Offset register is subtracted. A detailed figure of position measurement can be found in the annexes.

EnablePreset

int EnablePreset(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function causes the Preset Position value to be taken into consideration in position measurement function. The Preset Position value minus the initial position value is added to the position value so that the first position value (PPO) is equal to the Preset Position value. After an axis reset, this function causes the position value to be initialized to the value in the 37-bit Preset Position register.

Dev: the device

Axis: the axis number

EnableSCLKResetOnAxisReset

int EnableSCLKResetOnAxisReset(struct SIS1100\_Device\_Struct\* dev, unsigned short axis).

This function enables SCLK0 or SCLK1 (as selected by the SCLKSelectOnAxisReset function) to perform a quick reset and a time reset on the first occurrence of SCLK after an axis reset.

Rmk: Make sure to select the clock signal to perform the reset( using SCLKSelectOnAxisReset function) before calling this function.

DisableSCLKResetOnAxisReset

int DisableSCLKResetOnAxisReset(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function disable quick reset and time reset on the first occurrence of SCLK after an axis reset.

SCLKSelectOnAxisReset

int SCLKSelectOnAxisReset(struct SIS1100\_Device\_Struct\* dev, unsigned short axis, unsigned char SCLK)

This function determines which SCLK signal on the P2 bus causes the reset enabled by the SCLK Reset Enable function.

Rmk: MUST call EnableSCLKResetOnAxisReset after this function.

arguments:

SCLK: the select clock to perform the said reset.

0 for SCLK0

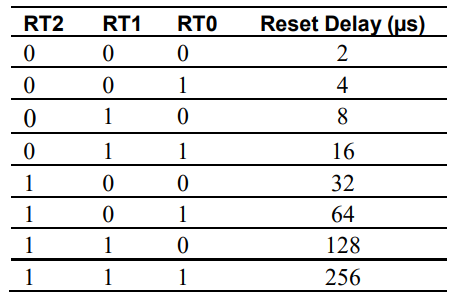
1 for SCLK1

SetTimeDelayBetweenResAndCompleteBit

int SetTimeDelayBetweenResAndCompleteBit(struct SIS1100\_Device\_Struct\* dev, unsigned short axis, unsigned char SCLK)

This function determines the time delay between an axis reset or position reset command and the assertion of the Position Reset Complete. The delay choices are shown in Table 4-4. If the Enable Reset Finds Velocity bit is set, this is the delay after the velocity finding operation is finished.

Rmk: this function is available only for axis 1 and 3. However, exceptionally, this setting on axis 1 is replicated on axis 2. In the same vein, the one on axis 3 is replicated on axis 4.



EnableResetFindsVelocity

int EnableResetFindsVelocity(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function enables searching for the measurement signal frequency after an axis reset. This is required if the velocity when the axis is reset may be greater than 0.1 m/sec.

DisableResetFindsVelocity

int DisableResetFindsVelocity(struct SIS1100\_Device\_Struct\* dev, unsigned short axis)

This function disables searching for the measurement signal frequency after an axis reset.

**BoardControlMode**

int BoardControlMode(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int biasMode)

The function configures the running mode of a specific axis. It starts by selecting the mode using the BiasControlMode function(Refer to that section for more detail about the different mode.) then start the bias calculation to automatically determine the bias.

**StartBiasCalculation**

int StartBiasCalculation(struct SIS1100\_Device\_Struct\* dev, unsigned char axis)

The function triggers the bias calculation of a specific axis. It's used after setting up the bias mode to automatically determine the bias.

**BiasControlMode**

int BiasControlMode(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int mode)

This function select the Bias control mode of the ZMI board. There are 4 modes:

* **Bias Off Mode.** In this mode, APD Bias is set to the minimum which is 55V
* **Bias Constant Voltage Mode.** In Constant Voltage Mode, the APD Bias DAC register sets the APD bias voltage directly. In other modes, this register is read-only and contains the APD bias voltage that is set by the controller. The APD Bias voltage is calculated as follows:

APD\_Bias\_Voltage = APD\_Bias\_DAC \* 61.65 mV

APD\_Bias\_DAC = APD\_Bias\_Voltage / 61.65 mV

* **Bias Constant Gain Mode.** The APD Gain L2 Set register specifies the APD gain in Constant Gain mode. In other modes, this register is not used. The APD gain values used in all calculations are relative to the responsivity of a standard PIN photodiode at 633 nm. The register value is calculated as follows:

APD\_Gain\_L2\_Set = Log2(APD\_Gain\_Set)\*1024

APD\_Gain\_Set = 2(APD\_Gain\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register. The default value is approximately 7 (2875 L2) and the accepted range is 4 (2048 L2) to 32 (5120 L2).

* **Constant Optical Power Mode.** This register specifies the maximum optical power that is expected in Constant Optical Power mode. This is used to determine the APD gain. The units of optical power are microwatts into the fiber optic receiver.

The register value is calculated as follows:

APD\_Opt\_Pwr\_L2\_Set = Log2(APD\_Opt\_Pwr\_Set)\*1024

APD\_Opt\_Pwr\_Set = 2(APD\_Opt\_Pwr\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register. The default value is 1uW (0 L2) and the accepted range is 70nW (-3930 L2) to 10uW (3402 L2).

* **Sig RMS Adjust Mode.** This register specifies the desired Sig RMS L2 value used in Sig RMS Adjust mode and in Constant Optical Power mode. This is used by the internal processor to determine the APD gain. The register value is calculated as follows:

APD\_Sig\_RMS\_L2\_Set = Log2(APD\_Sig\_RMS\_Set)\*1024

APD\_Sig\_RMS\_Set = 2(APD\_Sig\_RMS\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register.

The default value is 13,200 and the accepted range is within the values set in Sig RMS L2 Min Lim and Sig RMS L2 Max Lim.

**configureFlyscan**

int configureFlyscan(struct SIS1100\_Device\_Struct\* dev, unsigned char nbrAxis, USHORT freqKHz, UCHAR trig)

This function configures the Flyscan mode to continuously acquire position data from the selected axes. Based on the number of axis the sample frequency can reach up to 16Khz without memory overlapping.

The number of axis is ranged from 1 to 4. If it’s lesser than or equal to 2, the maximum sample frequency is 16Khz. If it’s greater than 2, therefore the maximum frequency is 8Khz.

The acquisition process can be started as soon as the configuration is done by setting trig to 1.

Rmk: Assume the followings rules:

if you want to perform flyscan with only 1 axis, use the main axis ( axis 3)

If you want to perform Flyscan with 2 axis, use axis 1 and 3

If you want to perform Flyscan with 3 axis, use axis 1, 2 and 3

**acquireFlyscanData**

int acquireFlyscanData(struct SIS1100\_Device\_Struct\* dev, unsigned char nbrAxis, PUINT startAdress\_axis1, PUINT startAdress\_axis3)

Zygo provides up to 64Kbytes RAM to stream values at the full resolution of lambda/4096.

This function continuously acquire flyscan data from selected axes. It should be used after setting up the flyscan using the function configureFlyscan.

The RAM space is organized as 64 pages of 512 samples of 16bit words. you can store up to 32768 consecutive samples of the input data for one selected axis.

The data format for position is a 32-bit signed value (including three fractional bits) represented in two consecutive 16-bit words, MSB first. The SCLK rate may be as fast as 16Khz.

startAdress\_axis1 is the pointer to the memory allocated to store data from axis 1 and 2

startAdress\_axis3 is the pointer to the memory allocated to store data from axis 3 and 4

**SetPositionOffset32**

**int SetPositionOffset32 (struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int offsetPos)**

This function is use in 32 bits mode to set the value of position offset on a specific axis.

**Arguments:**

dev: device

axis : axis to work on

offsetPos: 32 bits value of the register

SetCompARegVal32

int SetCompARegVal32(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int compAval32)

The function provides a mean to set the 32bits value of comparator A register

**Arguments:**

dev: device

axis : axis to work on

compAval32: 32 bits value of the register

SetCompBRegVal32

int SetCompARegVal32(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int compBval32)

The function provides a mean to set the 32bits value of comparator B register

**Arguments:**

dev: device

axis : axis to work on

compBval32: 32 bits value of the register

int Enable37bitsSignExtension(struct SIS1100\_Device\_Struct\* dev, unsigned char axis)

The function is intended to extend the sign of the value measure on 37bits. This is particularly useful when the 37bits mode is activated.

**Arguments:**

dev: device

axis : axis to work on

Disable37bitsSignExtension

int Disable37bitsSignExtension(struct SIS1100\_Device\_Struct\* dev, unsigned char axis)

The function disable sign extension on 37bits. This is useful when switching back to 32bits mode

**Arguments:**

dev: device

axis : axis to work on

SetKpAndKvCoeff

int SetKpAndKvCoeff(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned short Kp, unsigned short Kv)

The function sets coefficients Kp and Kv of the digital filter on a specific axis.

The user should determine the desired measurement bandwidth for their particular application.

The proper Kp and Kv values should be selected according to the performance attributes shown

in the filter performance tables (settling time, overshoot, following error, bandwidth, and gain

peaking) and graphs (magnitude, phase, and group delay vs. frequency).

For a closed loop control system, the filter bandwidth may be set to approximately 10 times the

control loop bandwidth. For a given bandwidth there may be several reasonable choices that

differ primarily in gain peaking and group delay response.

For open loop measurements, the filter bandwidth may be set to slightly higher than the required

measurement bandwidth. For a given bandwidth there may be several reasonable choices that

differ primarily in overshoot and settling time.

Possible values and the performance related are given in the Appendix.

**Arguments:**

dev: device

axis : axis to work on

Kp: Value of the Kp coefficient

Kv: Value of the Kv coefficient

SetCompARegVal37

int SetCompARegVal37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int compAval32, unsigned int compAvalExt)

The function sets the 37bits value of the comparator A register. The value should be break into 2 packets: the first one of size 32bits and the last one of size 5bits(the 5 MSB)

**Arguments:**

dev: device

axis : axis to work on

compAval32: The 32 bits packet

compAvalExt: The 5 bits packet

Returns:

0 if success

-1 else

SetCompBRegVal37

int SetCompBRegVal37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int compBval32, unsigned int compBvalExt)

The function sets the 37bits value of the comparator A register. The value should be break into 2 packets: the first one of size 32bits and the last one of size 5bits(the 5 MSB)

**Arguments:**

dev: device

axis : axis to work on

compBval32: The 32 bits packet

compBvalExt: The 5 bits packet

Returns:

0 if success

-1 else

SetPositionOffset37

int SetPositionOffset37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int offsetPos32, unsigned int offsetPosExt).

The function sets the 37bits value of the offset position register. The value should be break into 2 packets: the first one of size 32bits and the last one of size 5bits(the 5 MSB)

**Arguments:**

dev: device

axis : axis to work on

offsetPos32: The 32 bits packet

offsetPosExt: The 5 bits packet

Returns:

0 if success

-1 else

SetPresetPosition37

int SetPresetPosition37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int presetPos32, unsigned int presetPosExt)

The function sets the 37bits value of the preset position register. The value should be break into 2 packets: the first one of size 32bits and the last one of size 5bits(the 5 MSB)

**Arguments:**

dev: device

axis : axis to work on

offsetPos32: The 32 bits packet

offsetPosExt: The 5 bits packet

Returns:

0 if success

-1 else

SetPresetPosition32

int SetPresetPosition37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int presetPos32)

The function sets the 32bits value of the preset position register.

**Arguments:**

dev: device

axis : axis to work on

offsetPos32: The 32 bits packet.

Returns:

0 if success

-1 else

ReadVMEErrs

int ReadVMEErrs(struct SIS1100\_Device\_Struct\* dev, unsigned char axis)

The function reads and parses values of all Zygo VME Error registers on a specific axis. This include any error to the VME Bus configuration.

**Arguments:**

dev: device

axis : axis to work on

Returns:

0 if success

-1 else

ReadAllErrs

int ReadAllErrs(struct SIS1100\_Device\_Struct\* dev, unsigned char axis)

The function reads and parses values of all Zygo Error registers on a specific axis. This include any that can be raised by the ZMI4104C board.

**Arguments:**

dev: device

axis : axis to work on

Returns:

0 if success

-1 else

ClearEEPROMErrs

int ClearEEPROMErrs(struct SIS1100\_Device\_Struct\* dev)

The function resets the EEPROM error register. All EEPROM errors previously raised are clear. If an error bit is still set, it’s likely the cause of the said error is still present.

**Arguments:**

dev: device

axis : axis to work on

Returns:

0 if success

-1 else

SetAPDGainL2

int SetAPDGainL2(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int APDGain)

This register specifies the APD gain in Constant Gain mode. In other modes, this register is not used. The APD gain values used in all calculations are relative to the responsivity of a standard PIN photodiode at 633 nm. The register value is calculated as follows:

APD\_Gain\_L2\_Set = Log2(APD\_Gain\_Set)\*1024

APD\_Gain\_Set = 2(APD\_Gain\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register. The default value is approximately 7 (2875 L2) and the accepted range is 4 (2048 L2) to 32 (5120 L2).

**Arguments:**

dev: device

axis : axis to work on

APDGain: The APDGain value

Returns:

0 if success

-1 else

SetAPDSigRMSL2

int SetAPDSigRMSL2(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int APDSigRMS)

The function sets the desired Sig RMS L2 value used in Sig RMS Adjust mode and in Constant Optical Power mode. This is used by the internal processor to determine the APD gain.

The register value is calculated as follows:

APD\_Sig\_RMS\_L2\_Set = Log2(APD\_Sig\_RMS\_Set)\*1024

APD\_Sig\_RMS\_Set = 2(APD\_Sig\_RMS\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register. The default value is 13,200 and the accepted range is within the values set in Sig RMS L2 Min Lim and Sig RMS L2 Max Lim.

**Arguments:**

dev: device

axis : axis to work on

APDSigRMS: The APDSigRMS value

Returns:

0 if success

-1 else

SetAPDOptPwrL2

int SetAPDOptPwrL2(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int APDOptPwrL2)

This function specifies the maximum optical power that is expected in Constant Optical Power mode. This is used to determine the APD gain. The units of optical power are microwatts into the fiber optic receiver. The register value is calculated as follows:

APD\_Opt\_Pwr\_L2\_Set = Log2(APD\_Opt\_Pwr\_Set)\*1024

APD\_Opt\_Pwr\_Set = 2(APD\_Opt\_Pwr\_L2\_Set / 1024)

The value in this register only takes effect when processing a Start Bias Calc command from the P2 Command Register or the VME Command Register.

The default value is 1uW (0 L2) and the accepted range is 70nW (-3930 L2) to 10uW (3402 L2)The range is 0 to 0xFFFF.

**Arguments:**

dev: device

axis : axis to work on

APDOptPwr: The APDSigRMS value

Returns:

0 if success

-1 else

SetAPDBiasDAC

int SetAPDBiasDAC(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int APDBiasDac)

this function sets the APD bias voltage in Constant Voltage Mode.

The APD Bias voltage is calculated as follows:

APD\_Bias\_Voltage = APD\_Bias\_DAC \* 61.65 mV

APD\_Bias\_DAC = APD\_Bias\_Voltage / 61.65 mV

ParseVMEErrorStatus2

int ParseVMEErrorStatus2(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int\* VMEErrorStatus2Reg)

The function reads and parses VME Error Status 2 register.

VME Error Status 2 register contains the following error bits:

* Proc Fail error bit
* Bias supply error bit
* Write Protect error bit
* Sig max reached flag
* Sig min reached flag
* Bias error bit
* APD DC Error bit
* APD Command error
* APD Fail error bit
* APD Temp error bit

VMEErrorStatus2Reg: this argument is provided to store the value of the register

ParseVMEPosErrs

int ParseVMEPosErrs(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int\* VMEPosErrReg)

The function reads and parses VME Position register. This register includes:

* The measurement error bit
* The reference error bit

It can also be used to check whether the measurement or the reference signal is present or not.

VMEPosErrReg: this argument is provided to store the value of the register

ParseVMEErrorStatus1

int ParseVMEErrorStatus1(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int\* VMEErrorStatus1Reg)

The function reads and parses VME Error status 1 register. The function detects:

* Overflow on the VME 32bits position register
* Overflow on the VME 37bits position register
* The user velocity error
* The velocity error

It can also be used to check whether the measurement or the reference signal is present or not.

VMEErrorStatus1Reg: this argument is provided to store the value of the register

ParseVMEErrorStatus0

int ParseVMEErrorStatus0(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int\* VMEErrorStatus0Reg)

The function reads and parses VME Error status 0 register. This function detects:

* A CEC Error
* A phase noise error
* An acceleration error
* A glitch on the measurement signal
* A dropout on the measurement signal
* The SSI limits reach
* The saturation of the measurement signal
* The measurement signal missing eeror
* The overtempt error
* The FPGA sync error
* The reset failure
* The reference signal missing error
* The reference PLL Error
* A write error
* A power error

VMEErrorStatus0Reg: this argument is provided to store the value of the register

ParseAPDErrCode

int ParseAPDErrCode(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, unsigned int\* APDErrCode)

The function catches errors which occurred during the APD gain calculation on a specific axis. An exhaustive list of those errors can be found in Appendixes.

APDErrCode: this argument is provided to store the value of the register

ReadSamplePosition37

int ReadSamplePosition37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, double\* position).

The function samples and reads the 37bits position value on a specific axis. Reading this register latches data and reads the full position value

Rmk: This is different than reading the "position Register" which does not latch data

position: this argument is provided to store the measured position

ReadSamplePosition32

int ReadSamplePosition32(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, double\* position).

The function samples and reads the 32bits position value on a specific axis. Reading this register latches data and reads the full position value

Rmk: This is different than reading the "position Register" which does not latch data

position: this argument is provided to store the measured position

ReadPosition37

int ReadPosition37(struct SIS1100\_Device\_Struct\* dev, unsigned char axis, double\* position)

configure CEC hardware

At the startup CEC is disabled. When a motion occurs, CEC coeff are calculated. To configure CEC hardware:

1-First of all we should set CE\_Min\_Vel and CE\_Max\_Vel

- The default value of CE\_Min\_Vel is 96 which represents a 7.3Khz Doppler shift or a velocity of approximately 1.2mm/s. Its value shouldn't be less than 24 which represents a 1.8Khz Doppler shift and may cause improper operation of the CEC function

- The default value of CE\_Max\_Vel is 31457 which represents a 2.4Mhz Doppler shift or a velocity of approximately 0.38m/s(double pass)

Then, we should perform a motion at a velocity higher than the CE\_Min\_Vel and greater than CE\_Max\_Vel so that the the CEC coeff can be calculated. The motion should last at least 4.1ms before CEC coeff can be updated

Rmk:

When the measurement card is returning incoherent values, change the board’s position in the VME chassis.

Qt App tests

* Plot graphs: ok
* Initialize ok