M2i.46xx/-Exp VEE driver

Library LibM2iBase.vee
DAQ Example

English version **November 16, 2017**

SPECTRUM INSTRUMENTATION GMBH · AHRENSFELDER WEG 13-17 · 22927 GROSSHANSDORF · GERMANY
PHONE: +49 (0)4102-6956-0 · FAX: +49 (0)4102-6956-66 · E-MAIL: info@spec.de · INTERNET: www.spectrum-instrumentation.com

Installation

Driver structure

The driver for VEE is based on the standard board drivers for Windows 2000, Windows XP and Windows Vista, Windows 7 and Windows 8. New versions of the operating system and VEE drivers can be download from the internet www.spectrum-instrumentation.com free of charge. This driver supports HP-VEE, Agilent VEE, Keysight-VEE beginning with version 5.0. The driver consist of libraries for accessing the board and some demo programs that show how to use the library. The demo programs may be used as a base for own programs.

Installation of board drivers

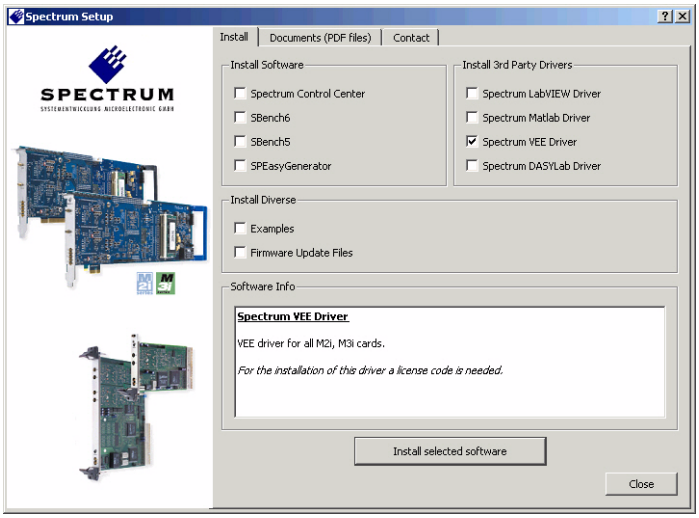
The standard board drivers should be installed prior to the VEE driver installation. An installation guide is found in the hardware manual of the specific board. Normally board drivers are automatically installed when the board is first time plug in the PC.

An update to e newer driver version could be done at any time without changing the VEE driver. Use at least the board driver version that was shipped together with the VEE driver. Using an older driver version is not recommended and could lead to any errors.

Installation of VEE drivers

Please follow these steps when installing the VEE driver:

- Install the card(s) into the system as shown in the hardware manual
- Install the standard Windows driver as shown in the hardware manual
- Install the VEE driver as explained below

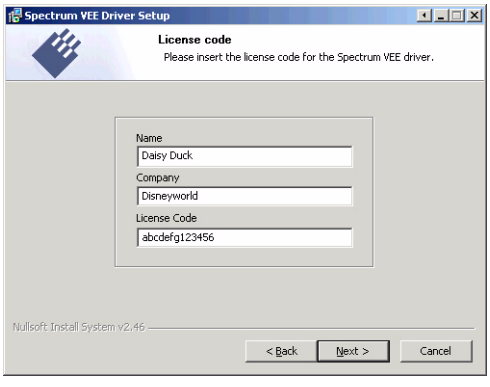


The VEE driver is delivered as a self extracting archive secured by an installation code. If you purchased the VEE driver together with your card you'll find the current driver on the CD delivered with the card. Please follow the CD menu to „Software Installation" -> „VEE driver" as shown on the right side.

It is also possible to install the VEE driver manually selecting the install file with the windows explorer. Please select the path CD:\Install\win\spcm_drv_vee_install.exe and execute the file. The installer will guide you through the installation routing step by step.

If you purchased the VEE driver license later you'll can download the latest version from the it the Spectrum homepage. Please store the exe file somewhere on your system and start it from this location.

The VEE driver is delivered together with an installation license. You need this license for installation on a new machine. The installation license is bound to one person name/company. Please enter the license information in the exact way as it is found in the license paper. The name and company as well as the license itself is case sensitive. Please be sure to use the correct spelling including upper and lower characters. The license code itself is completely of lower characters. Please do not add any further blanks neither in the license code not in the name/company information.



The VEE driver files are installed under the program folder in an extra directory \program files\Spectrum GmbH\VEE drivers. When moving the files please make sure to move the complete directory with all sub-directories as the driver consists of several examples and libraries that are used together with the examples.

Please note that the installer and the licensing has been updated June 2010. If you purchased a license prior to that date, you'll need to obtain an updated license code from Spectrum. Please request a free update providing your card's serial number.

VEE Driver Update

As the VEE driver also uses the standard Windows drivers as a base, any updates on these drivers will improve the system and and any changes are available under VEE immediately. Updating the VEE driver can simply be done by installation of the new VEE driver archive. It is normally not necessary to insert the license code again as long as the driver is updated on a machine where a previous version of the VEE driver already has been installed.

Please note that the installer and the licensing has been updated June 2010. If you purchased a license prior to that date, you'll need to obtain an updated license code from Spectrum. Please request a free update providing your card's serial number.

Demo mode

The VEE driver runs fine with demo cards installed in your system. Please follow the steps in the hardware manual to see how you insert a simulated demo card into your system. Please keep in mind that the generated data is only simulated. The simulation and calculation of demo data takes more time than just transferring data from hardware to the PC. Therefore the performance of the system is worse when using demo cards.

Library LibM2iBase.vee/LibM3iBase.vee

This library contains all driver functions that are related to the M2i and M2i-Express series (LibM2iBase.vee) or M3i and M3i-Express series (LibM3iBase.vee). These functions found in this library cover the complete range of boards of one series. Therefore there may be some functions that are not used on your board type. It accesses the board driver DLL. For using this library the appropriate header file Spcm_vee16.h must be present in the same directory. If you encounter any problems when using the library please check for this file.

The library is shipped as a source code file. This allows the user to implement changes or additional functionality in the drivers. If you find any bugs in the driver library please inform us so that we can fix them.

Example files in directory spcm_examples

Together with the VEE driver library there are several examples installed in the same directory. These examples show the use of the library for different setups and board versions. However, these files are only examples that show the use of the library, they are not fully-tested applications. And they do not offer the complete functionality of the board. The examples are shipped "as-is". They could be modified in any way by the user and could also be used as a base for own applications.

Library LibM2iBase.vee

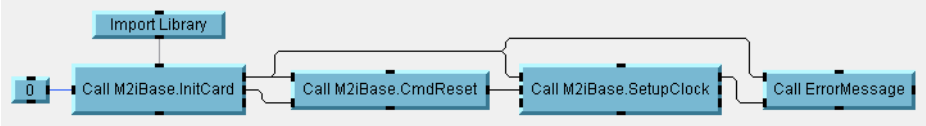
General Information

This library is designed to cover the complete range of M2i- and M2i-Express boards. Therefore it has a lot of functions for data read/write and all functions for setting up the base board related stuff. Not all of these functions have to be used for your special board. Please refer to the examples to see which ones are necessary and have a look at the table on the next page.

Initialisation

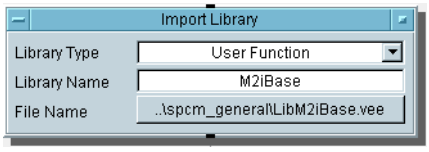
Before accessing the board an initialisation of the driver and the board must be performed. This is done with the user function "InitCard" described afterwards. All user functions except the Init function require the card handle to know which board should be accessed. The card handle is derived by the „InitCard“ function and must be routed to all hardware accessing functions. The „InitCard“ function retrieves an index of the card to be initialised. The index starts with zero with the first card found in the system.

Error processing



Each user function that could generate an error in the driver has an error input and an error output. Before processing the functionality the user functions checks the error code whether it is still not set. The error code could be simply routed through all functions calls until a final error checking with the function "ErrorMessage". The „ErrorMessage“ function is described later in more detail.

Importing library

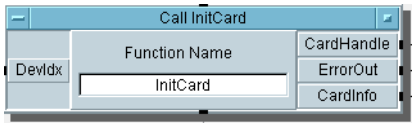


Before using the driver library it must be imported in your VEE program. Use the "Import library" function to do so. You need to select the path where you installed the library to. This must also be done for all the examples. The library functions are available in VEE after you one time called the "Import library" function.

General User functions

The library contains more than 20 user functions that could be used to control the M2i- and M2i-Express board series. Not all of these functions are related to the board type that you use. All settings are directly written through to the driver. Therefore all limitations, all format and constant definitions of the standard driver and are also used for the VEE driver. To see what values are allowed for an input, take a look at the hardware manual and check the setup that is described there. The use of the library is shown with the different examples explained later.

InitCard



The Init user function initialises loads the driver dlls that are needed, initialises the card with the given index and reads out some basic information of the card. If more than one card is in the system this function must be called multiple times with a different index to open all different cards.

The „InitCard" function has to be called at the very beginning of the program to load the needed basic driver functions.

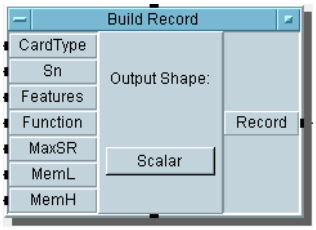
Inputs

DevIdx	The card index that should be opened.

Outputs

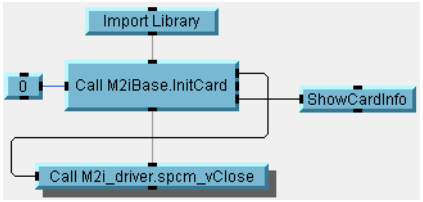
CardHandle	If the card was opened successful this output contains the valid card handle. All other hardware accessing functions need this card handle.
ErrorOut	Contains the error code of this function. If everything went well this output is zero (ERR_OK)
CardInfo	A record containing some basic card information. Feel free to add other information to this record if your program needs additional information of the card.

Record CardInfo



CardType	The exact type of card that was found under that index. The card types are described in the hardware manual. As an example the M2i.3011 returns #H33011
Sn	The serial number of the card as also found on the delivery note
Features	Installed features as found in register SPC_PCIFEATURES and described in the hardware manual
Function	The basic card function as found in register SPC_FNCTYPE
MaxSR	The maximum sampling rate of the card
MemL	The installed memory in bytes. Split into a 32 bit low part
MemH	and a 32 bit high part

Closing the card



After using the „InitCard" function the driver and handle of this card is open inside VEE. It is only possible to open a hardware driver once. At the end of the program it is absolutely necessary to close the handle again to re-use this card later on. This is shown at the VEE program except at the right.

The card handle is closed automatically if the driver library is unloaded. Unfortunately VEE does not unload the drivers it has once opened when finishing a program. Therefore you may get the error message „card with that device name wasn't found" when starting your VEE program a second time. This indicates that the card handle wasn't closed properly and the card can't be accessed. In that case please just close the VEE environment and re-open it immediately. Don't forget to update your program to have the close call at every program exit!.

CmdReset

The reset function performs a software reset on the board. After reset all internal settings are set back to the default values and the board is in idle condition.

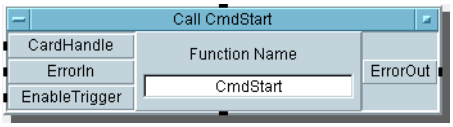
Inputs

CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.

CmdStart



Starts the board with current setup. All settings must be done before. If used on an output board, valid data must be written to the hardware with one of the "WriteData" user functions before. The „EnableTrigger“ selects whether the card is started with trigger engine already enabled or whether this should be done later on by a specific call.

Inputs

CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function
EnableTrigger	Write a „1“ to this input to enable the trigger engine right from the start. Write a „0“ to start the card without trigger enabled. It is send necessary to enable the trigger with a call to the „CmdEnDisTrigger“ function later.

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.

CmdStop

Immediately stops the board output either in standard or in FIFO mode.

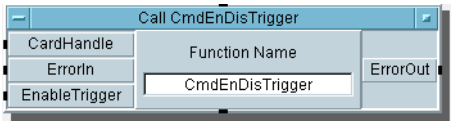
Inputs

CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.

CmdEnDisTrigger



Enables or disables the trigger engine while the card is running. While the trigger engine is disabled none of the trigger inputs will generate a card trigger. This function need to be used if the card has been started with the trigger engine disabled.

Inputs

CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function
EnableTrigger	Write a zero to this input to disable the trigger engine. Write anything not zero to enable the trigger engine.

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.

CmdForceTrigger

Sends a force trigger command to the card. In case that the card has not triggered on an external signal so far the card immediately triggers one time. The card then behaves as if used together with the software trigger.

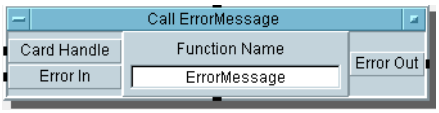
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

ErrorMessage



Checks the error code. If the error code is not zero the error information is read from the driver and displayed in a message box. Afterwards the driver is closed and the execution of the program terminates.

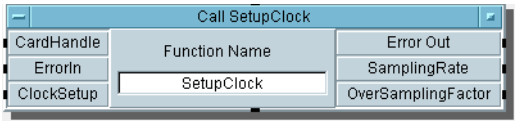
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was zero at call time and no error was to display

SetupClock



Sets all clock information for the board to the driver. The driver checks them for correct settings and gives back an error code if the setup is not allowed. Some of the record fields are only used if the matching clock mode is selected. Please refer to the hardware manual for further information.

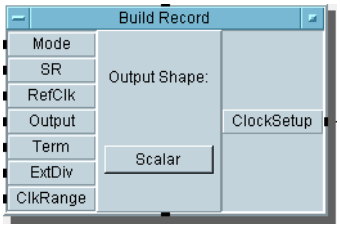
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
ClockSetup	A record containing all the clock setup of the card.

Outputs

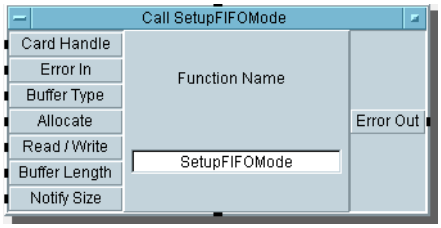
ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
SamplingRate	The sampling rate that was finally set by the driver. As the driver sets the nearest matching sampling rate depending on the capabilities of the card this may differ from the desired sampling rate.
OverSamplingFactor	The oversampling factor is read out from the driver. This factor shows the rate by which the internal ADC is running faster than the sampling rate itself. It is used for scaling of timestamps.

Record ClockSetup



Mode	Selected clock mode as found in the hardware manual (SPC_CM_xxx)
SR	Desired sampling rate if internal clock mode is used (in Hz)
RefClk	Fed in reference clock if reference clock mode is used (in Hz)
Output	A zero disables the clock output, a non-zero enables it
Term	A zero disables the clock termination, a non-zero enables it
ExtDiv	The external clock divider if one of the divided modes are used
ClkRange	The clock range if external clock is used

SetupFIFOMode



This is a general setup function for the FIFO mode. Most of the FIFO mode work including the buffer handling is done inside the library. Prior to starting the FIFO mode all buffer settings have to be done with this function.

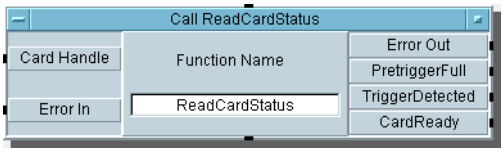
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
Buffer Type	The kind of FIFO buffer to be allocated 0 = data, 1 = timestamp, 2 = aba
Allocate	Writing a 1 allocates the buffer data. Writing a 0 frees the buffer data. The old Buffer is also automatically freed if a new buffer is allocated.
Read/Write	The desired transfer direction: 0 = Write, 1 = Read
Buffer Length	The length of the buffer to be allocated in Bytes
Notify Size	The notify size of the transfer in bytes. Please see the hardware manual for details on the notify size

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

ReadCardStatus



Reads out the current card run status and splits the status word into the 3 run flags PretriggerFull, TriggerDetected and CardReady.

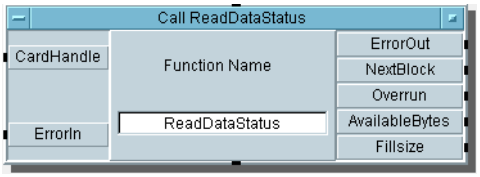
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
PretriggerFull	Returns a „1" if the pretrigger area has been filled and the card is able to detect trigger
TriggerDetected	Returns a „1" if the trigger has been detected and the acquisition/output has started
CardReady	Returns a „1" if the card has stopped and the acquisition/ replay has finished

ReadDataStatus



Reads out the current data transfer status and splits the status word. The function also reads the available bytes and the current fill size of the buffer. This function is solely used for FIFO mode.

Inputs

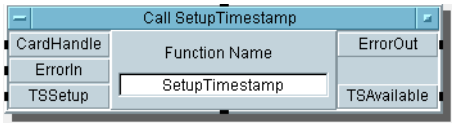
CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
NextBlock	Returns a „1" if the next data block is available
Overrun	Returns a „1" if an overrun/underrun has occurred while FIFO transfer was running
AvailableBytes	The number of bytes that is available for transfer
Fillsize	The current fill size of the on-board buffer in promille (1/1000)

General User Objects

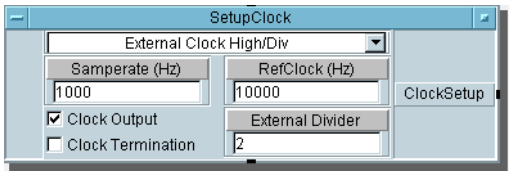
Inside the VEE library there are some different user objects that group settings that belong together and also contain the initial setup of the inputs objects. These user objects can be used to speed up vee programming and to have an easy-to-use interface that automatically disables input objects that are not used on the current setup. These user objects are also used throughout the examples.



To use these objects please just copy them from the main part of the library inside your vee program. After this you only need to connect the output of the object to the matching setup function and add the user object to the panel.

SetupClock

This object handles the complete clock setup of one card. All clock related settings are grouped inside the panel. Depending on the selected clock mode some of the inputs are then disabled.



Inputs

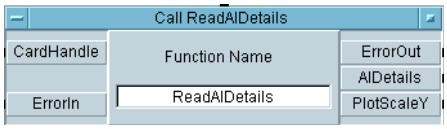
no inputs available

Outputs

ClockSetup	A record containing the clock setup as needed by the function „SetupClock“. Please see this chapter to have a documentation of the record

Analog Input User functions

ReadAIDetails



The function reads out all details of the analog input. The resulting record is used to feed the AI setup function and helps to make the programming universal.

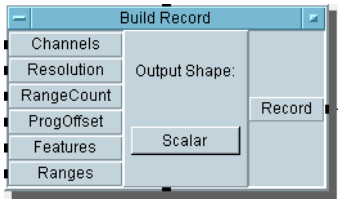
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

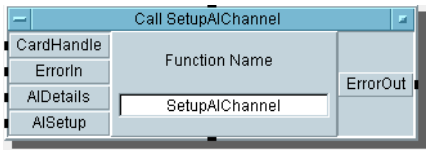
ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
AIDetails	A record containing all the details of the analog inputs.
PlotScaleY	Record containing the correct plot scaling for the data acquisition card. Can be directly fed into the „Scales" input of the „XY Trace" object

Record AI Details



Channels	Number of analog input channels on this card
Resolution	Analog input resolution in bits
RangeCount	Number of different programmable input ranges per channel
ProgOffset	maximum value of the programmable offset if the card has a programmable offset. This value is either in percent or mV depending on the AI features
Features	A bitfield with the analog input features. This is read-out from the register SPC_READAIFEATURES. The available features are found in the manual
Ranges	An array containing all the available input ranges as integer values scaled to mV.

SetupAIChannel



Makes the setup of one input channel. Please be sure to call this function only for input channels that are installed on the specific card type otherwise you will get an error.

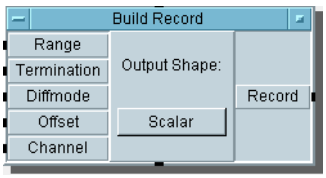
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
AI Details	The details of the analog inputs as returned by the „ReadAIDetail" function
AISetup	A record containing all the analog setup of one specific input channel.

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

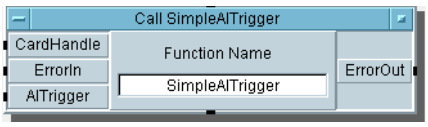
Record AISetup



Please note that this function is a general setup function covering the complete range of Spectrum analog input cards. Not all features are supported by all card types. The „SetupAIFunction" will automatically determine the features available and program only those features. Please just ignore the features inside the record that are not supported by your hardware.

Range	The selected input range for this channel. The input range has to be in mV as expected from the driver.
Termination	A zero disables the input termination (high impedance). A value different from zero enables the 50 Ohm termination
Diffmode	Switches between single-ended by writing a zero and differential inputs by writing a non-zero
Offset	Programs the analog offset of the channel. The value has to be either in percent or mV depending on the capabilities of the card which can be found inside the „AIDetails"
Channel	The index of the channel which has to be programmed.

SimpleAITrigger



The simple AI trigger functions offers an easy way to access some of the basic trigger functions of the card. It only supports a part of the very powerful trigger engine. The more complex trigger settings need to be programmed separately.

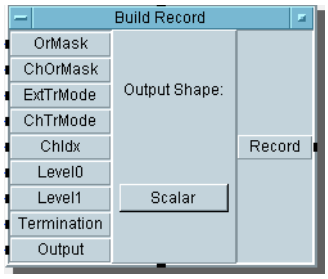
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
AITrigger	A record containing all the trigger settings

Outputs

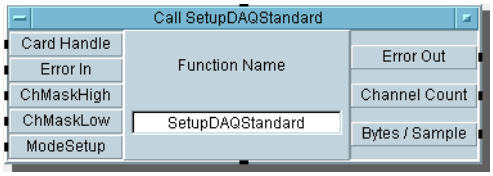
ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

Record AITrigger



OrMask	The global trigger OR mask. Inhere one of the global trigger sources, software or external are selected
ChOrMAsk	The channel OR mask. One of the channel trigger sources can be selected here. Please note that this function only supports simple one source triggers and no OR conjunction is possible
ExtTrMode	If external trigger source is selected one of the external trigger modes need to be selected. Trigger modes are found in the hardware manual (SPC_TM_POS,..)
ChTrMode	If channel trigger is selected this field holds the channel trigger mode (SPC_TM_POS,...)
ChIdx	The index of the channel which was selected to be trigger source. Channel counting starts with zero. Please be sure to not exceed the number of installed channels with this value
Level0	The trigger level 0 to be programmed if channel trigger is selected. Please check with the hardware manual to see which
Level1	The trigger level 1 if a channel trigger mode is selected that needs two trigger levels to be programmed.
Termination	A zero disables the external trigger termination, a non-zero value enables it
Output	A zero disables the external trigger output, a non-zero value enables it

SetupDAQStandard



The function programs one of the standard acquisition modes and all mode related settings like memory size and segment size. Please use the related „SetupDAQFIFO" in case that the FIFO mode should be used.

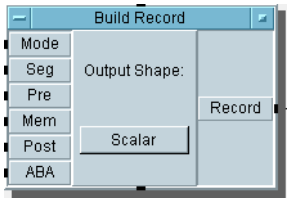
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
ChMaskHigh ChMaskLow	The channel mask of activated channels for the next run. Split into a high 32 bit and a low 32 bit part
ModeSetup	A record containing all the mode settings

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
Channel Count	The total number of activated channels.
Bytes / Sample	The number of Bytes user per sample for storage.

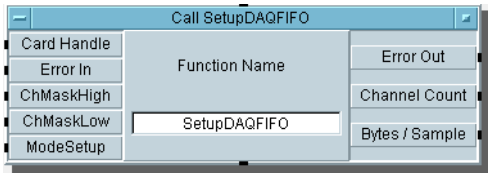
Record ModeSetup



Mode	The selected recording mode as found inside the hardware manual (SPC_REC_STD_SINGLE). Please be sure that the selected mode is installed on your hardware. Some of the modes require an extra option to be installed
Seg	Segment size in samples/channel for all modes that use a segment size
Pre	Pretrigger in samples/channel for Gated Sampling mode
Mem	Total memory in samples/channel to be acquired. Be sure not to exceed the amount of installed memory when programming this
Post	Posttrigger value in sample/channel. This gives the number of samples to be acquired after receiving the trigger event or in case of Gated Sampling after detection of the ending gate edge
ABA	The ABA divider in case that the ABA mode is selected

SetupDAQFIFO

The function programs one of the FIFO acquisition modes and all mode related settings like Post trigger and segment size. Please use the related „SetupDAQStandard" in case that the standard mode should be used.



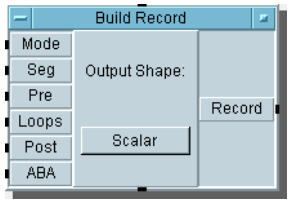
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
ChMaskHigh ChMaskLow	The channel mask of activated channels for the next run. Split into a high 32 bit and a low 32 bit part
ModeSetup	A record containing all the mode settings

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
Channel Count	The total number of activated channels.
Bytes / Sample	The number of Bytes used per sample for storage.

Record ModeSetup

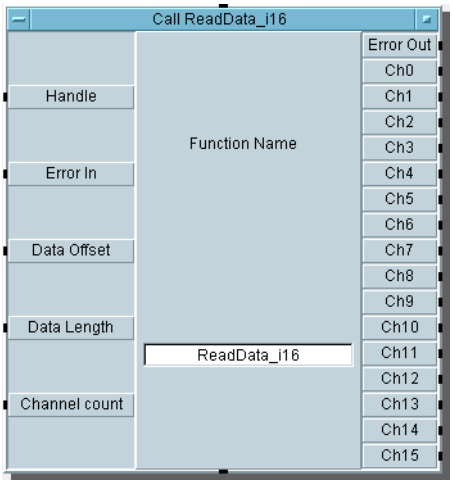


Mode	The selected recording mode as found inside the hardware manual (SPC_REC_FIFO_SINGLE). Please be sure that the selected mode is installed on your hardware. Some of the modes require an extra option to be installed
Seg	Segment size in samples/channel for all modes that use a segment size
Pre	Pretrigger in samples/channel for Gated Sampling mode
Loops	Number of segments to be acquired. If left to zero the acquisition will run endless. Otherwise the segment size and the loops define the recording length
Post	Posttrigger value in sample/channel. This gives the number of samples to be acquired after receiving the trigger event or in case of Gated Sampling after detection of the ending gate edge
ABA	The ABA divider in case that the ABA mode is selected

ReadData_i16

This function reads out sorted data from the on-board memory of the FIFO buffer. It is feed with the matching setup to be read out containing of data offset and data length. Data is sorted to up to 16 output channels and normalized to 16 bit integer values.

Please be sure to set the input parameters matching the current acquisition. Otherwise data may be corrupted.



Inputs

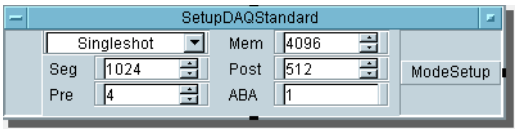
CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function
Data Offset	The offset in samples/channel from where the read function should start
Data Length	The data length in samples/channel to be read. In standard mode this is normally similar to the programme memory size (SPC_MEMSIZE) and in FIFO mode this is similar to the notify size. When using standard mode one can read the acquired data in smaller chunks by using the offset and the length parameters
Channel count	The number of channels to be read. This must be similar to the programmed number of channels to acquire

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.
Ch0. Ch15	Normalized and sorted channel data in 16 bit integer format

Analog Input User Objects

SetupDAQStandard



The object handles all standard data acquisition setup of one card. All mode and memory related settings are grouped inside this panel. Depending on the selected mode some of the inputs are disabled. The mode drop-down box also contains card modes that are only available when a specific option is installed like Multiple Recording or Gated Sampling. Please be sure to have this option installed before selecting this mode. Otherwise the driver will return an error code indicating that this option is not available.

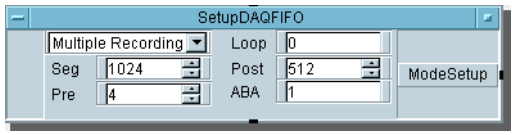
Inputs

no inputs available

Outputs

ModeSetup	A record containing the standard mode setup as needed by the function „SetupDAQStandard“. Please see this chapter to have a documentation of the record.

SetupDAQFIFO



The object handles all FIFO data acquisition setup of one card. All mode and memory related settings are grouped inside this panel. Depending on the selected mode some of the inputs are disabled. The mode drop-down box also contains card modes that are only available when a specific option is installed like Multiple Recording or Gated Sampling. Please be sure to have this option installed before selecting this mode. Otherwise the driver will return an error code indicating that this option is not available.

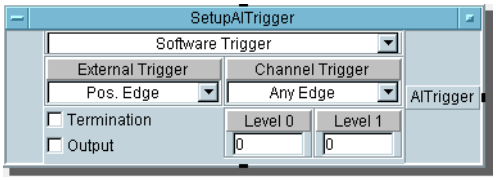
Inputs

no inputs available

Outputs

ModeSetup	A record containing the standard mode setup as needed by the function „SetupDAQFIFO“. Please see this chapter to have a documentation of the record.

SetupAITrigger



This object handles the simple AI trigger setup. It allows to select one single trigger source and sets up all related trigger settings for this source. depending on the selected trigger source and also depending on the selected mode some of the inputs are disabled by the object then. The trigger source list contains all possible analog inputs of all available analog cards. Therefore it is possible that some channels are listed here that are not available on your specific card

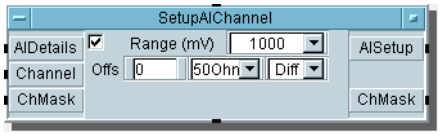
Inputs

no inputs available

Outputs

AITrigger	A record containing the standard trigger setup as needed by the function „SimpleAITrigger“. Please see this chapter to have a documentation of the record.

SetupAIChannel



This AIChannel object covers the complete range of Spectrum analog data acquisition card. Therefore it contains all possible settings that may be modified for the different card types. By feeding in the „AIDetails“ record as received from „ReadAIDetails“ The object itself disables the settings that are not available for your specific card type. The ChMask input and output is an easy option to let the channel mask be setup automatically by the AIChannel object. Therefore the object examines the activation checkbox and sets the corresponding bit inside the ChMask bitfield. Please simply route the ChMask through all SetupAIChannel objects, initialised with a zero.

Inputs

AIDetails	The AIDetails record as received from the „ReadAIDetails“ function. Please see this chapter to have a documentation of this record
Channel	The index of the channel for which the setup object is called
ChMask	The current channel mask to be routed through

Outputs

AISetup	A record containing the AI channel setup as needed by the function „SetupAIChannel“. Please see this chapter to have a documentation of the record
ChMask	The updated channel mask

Timestamp Option

The following user functions and user objects are only available if the option timestamp is installed.

CmdWaitExtraDMA

Waits on the extra DMA to be ready. The extra DMA contains the timestamp and the ABA data.

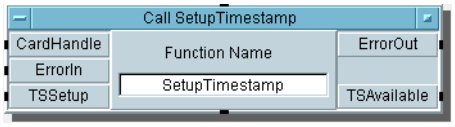
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

SetupTimestamp



The function checks for the availability of the timestamp option and programs the current timestamp setup.

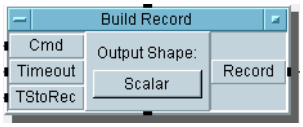
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
TSSetup	A timestamp setup record as described below

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
TSAvailable	A „1" indicates that the timestamp option is installed

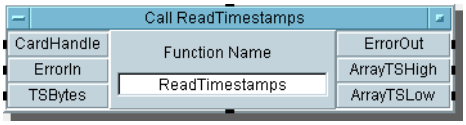
Record TSSetup



The record is used for the „Setup Timestamp" user function

Cmd	Timestamp Command as listed in the hardware manual
Timeout	Timeout if waiting for an external ref clock signal
TStoRec	Number of timestamps to be recorded

ReadTimestamps



The ReadTimestamps function reads out a number of timestamps and sorts them into two arrays containing the high and the low part of the timestamp.

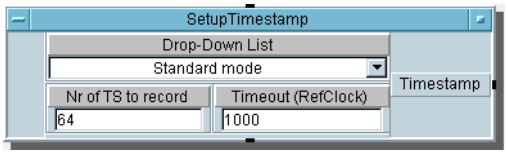
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
TSBytes	Number of timestamp bytes to be read by the function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
ArrayTSHigh	An array of 32 bit integer values containing the upper 32 bit of the timestamps
ArrayTSLow	An array of 32 bit integer values containing the lower 32 bit of the timestamps

User Objects Timestamp



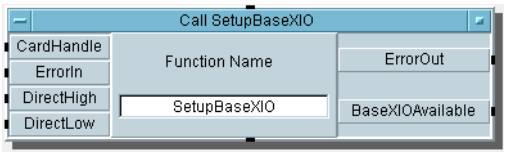
The user object allows the complete setup of the timestamp parameters. It generates a TSSetup record that can be directly used with the SetupTimestamp user function to program all details. Please see the hardware manual for details on the timestamp setup.Outputs

Timestamp	A TSSetup record to be used with the SetupTimestamp user function

BaseXIO Option

The following user functions and user objects are only available if the option BaseXIO is installed on your card.

SetupBaseXIO



Checks whether the BaseXIO option is installed and sets the direction of each group of 4 BaseXIO lines.

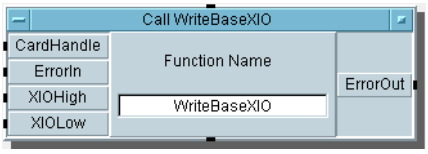
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
DirectHigh	Direction of the upper 4 BaseXIO lines (line 4 to 7). A zero turns the lines to input, a „1" to output.
DirectLow	Direction of the lower 4 BaseXIO lines (line 0 to 3). A zero turns the lines to input, a „1" to output.

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
BaseXIOAvailable	A „1" indicates that the BaseXIO option is installed

WriteBaseXIO



The function writes a BaseXIO value to the outputs

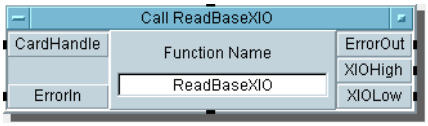
Inputs

CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function
XIOHigh	The upper 4 BaseXIO outputs (line 4 to 7)
XIOLow	The lower 4 BaseXIO outputs (line 0 to 3)

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.

ReadBaseXIO



The functions reads the BaseXIO lines.

Inputs

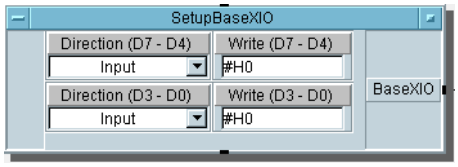
CardHandle	a valid card handle as returned from the „InitCard" function
ErrorIn	The routed error code from the last driver function

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn" was not zero at call time.
XIOHigh	The upper 4 BaseXIO inputs (line 4 to 7)
XIOLow	The lower 4 BaseXIO inputs (line 4 to 7)

User Objects BaseXIO

SetupBaseXIO

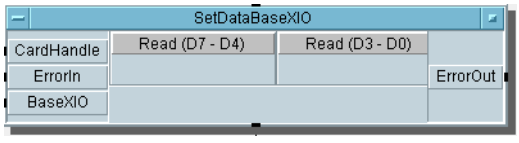


This user object only has a function if the option BaseXIO is installed on the card. The direction and some write data of the BaseXIO option are selected and collected to a record.

Outputs

BaseXIO	Record containing the direction and write data for both bundles of BaseXIO lines

SetDataBaseXIO



The function writes the data from the SetupBaseXIO function and reads out and displays the BaseXIO lines.

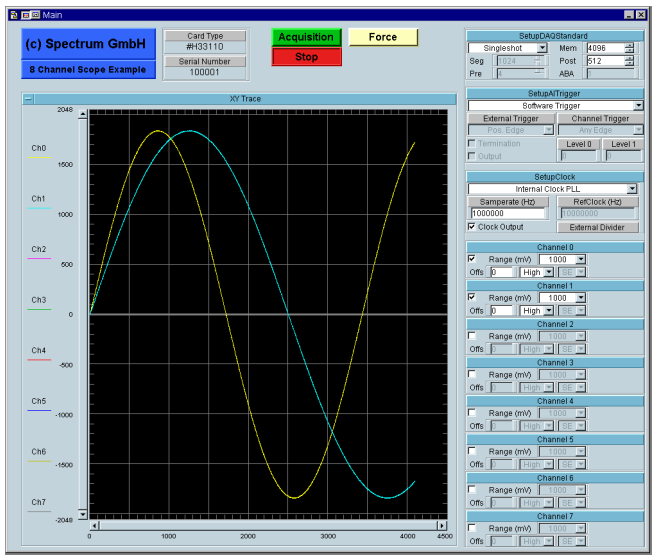
Inputs

CardHandle	a valid card handle as returned from the „InitCard“ function
ErrorIn	The routed error code from the last driver function
BaseXIO	record collected by „SetupBaseXIO“

Outputs

ErrorOut	The error code of this function or the error code of the last user function if „ErrorIn“ was not zero at call time.

Examples



These examples show the use of the driver for different types of boards. There are examples for standard mode and for FIFO mode. The standard mode examples can be used as a simple scope utility for the board. The examples can be used to test the board or as a base for an own program. The examples could be changed and used for own applications without any limitations.

Prior to the first start the path to the library must be changed in the examples to work properly on the installed system.

Scope_8Channel.vee

This example shows the use of the VEE library for analog data acquisition cards with up to 8 channels. The example works with analog data acquisition cards of the M2i and M2i-Express series.

After running the vee program the interface will be updated permanently with the changes and certain input fields will be disabled based on the current selection

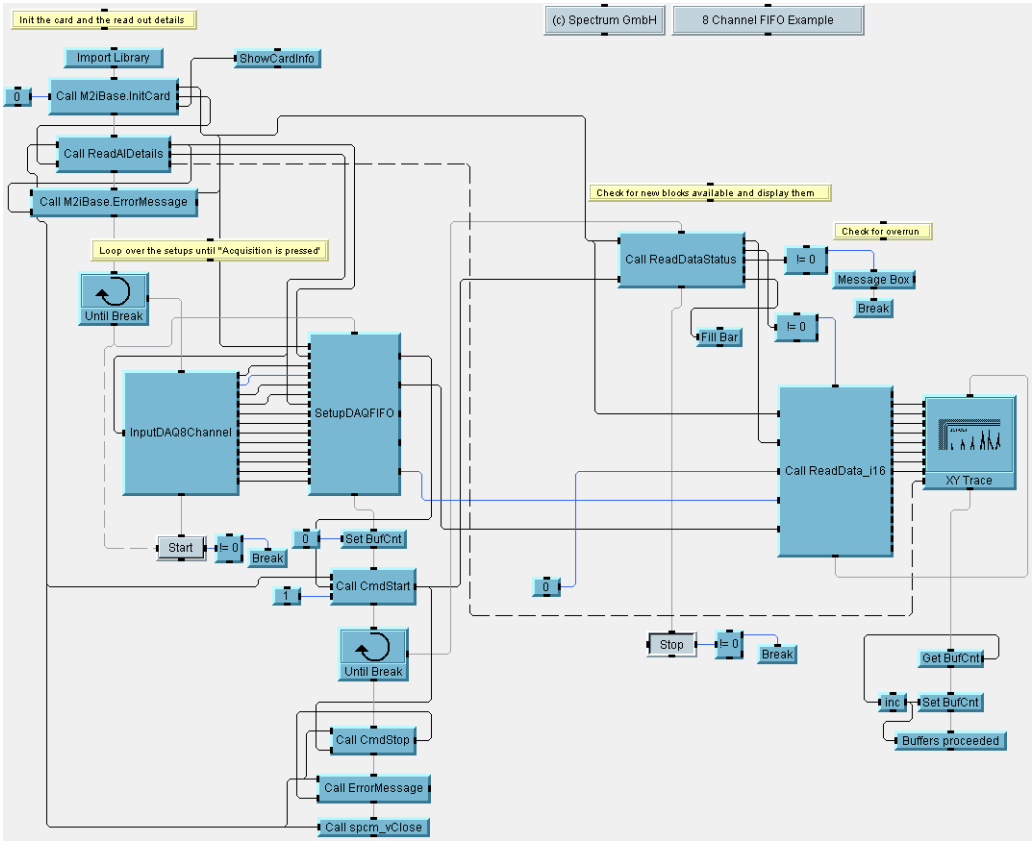
Pressing the „Acquisition" button starts the scope mode. Each recording is directly displayed on the XY Trace afterwards. The scope mode runs until the stop button has been pressed.

If no trigger condition is found it is possible to use the force trigger button to force a trigger event.

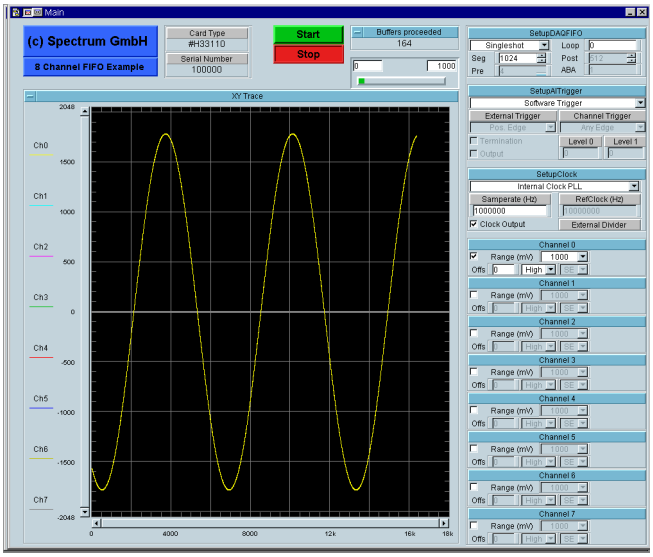
Local objects

The calls to the user functions of the library are encapsulated in local user objects to get a better visibility. These user objects are directly taken from the library and can be also used for own programs.

Setup



DAQFIFO_8Channel



This example shows the use of the FIFO mode with the VEE driver. It records and displays continuously data until the user stops the acquisition. The example supports up to 8 analog channels.

Setup

After running the example one can adjust the setup to match the requirements. In that stage all the setup objects are running inside a loop and input fields that are not available due to the current setup are disabled.

FIFO speed

The speed of the FIFO mode is affected by system speed, by other tasks that run in the background and by the software itself. If buffer overruns occur, set the buffer size and number of buffers to higher values. To get the maximum performance that is possible with VEE it is necessary to stop the data display. Any display routine always needs a lot of processing power.

BuffersProceeded

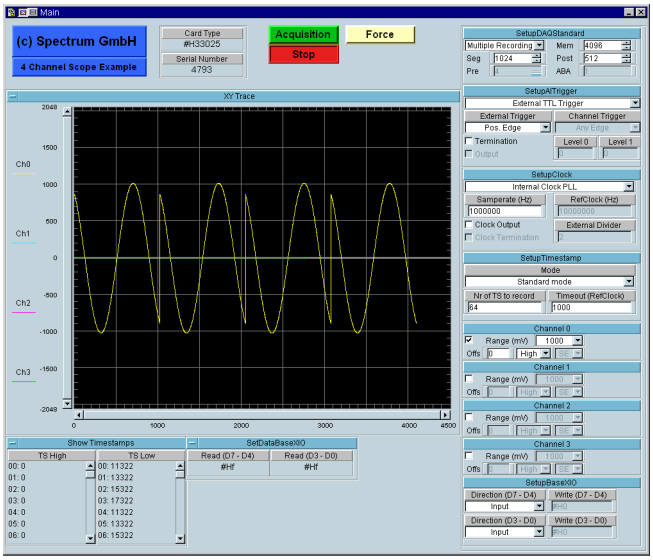
The display box shows the number of buffers that has so far been transferred by the FIFO mode example. Each buffer has a size that has been defined by the function „SetupFIFO". This section also shows the current fill level of the hardware FIFO buffer as returned from the „ReadDataStatus" function.

FIFO Loop

The FIFO loop performs the following steps:

- wait until the next buffer is ready
- read next buffer from driver.
- display the buffer in a strip chart.
- Increment the number of buffers proceeded and display them.
- check for an buffer underrun.
  - show error message if it occurs.
- check whether stop has been pressed

Scope4Ch_TS_BaseXIO



This exampel shows the use of the Timestamp and BaseXIO function with the VEE driver. It is based on a standard scope example and reads out timestamp and BaseXIO data in addition. The example only works if the timestamp and/or the BaseXIO option are installed on the card. It will check for both options and program only the options that is installed on the card.

The setup of the example is similar to the Scope_8Channels example and is therfore not described into more detail.

Error Codes

The following error codes could occur when a driver function has been called. Please check carefully the allowed setup for the register and change the settings to run the program.

error name	value (hex)	value (dec.)	error description
ERR_OK	0h	0	Execution OK, no error.
ERR_INIT	1h	1	An error occurred when initializing the given card. Either the card has already been opened by another process or an hardware error occurred.
ERR_TYP	3h	3	Initialization only: The type of board is unknown. This is a critical error. Please check whether the board is correctly plugged in the slot and whether you have the latest driver version.
ERR_FNCNOTSUPPORTED	4h	4	This function is not supported by the hardware version.
ERR_BRDREMAP	5h	5	The board index re map table in the registry is wrong. Either delete this table or check it carefully for double values.
ERR_KERNELVERSION	6h	6	The version of the kernel driver is not matching the version of the DLL. Please do a complete re-installation of the hardware driver. This error normally only occurs if someone copies the driver library and the kernel driver manually.
ERR_HWDRVVERSION	7h	7	The hardware needs a newer driver version to run properly. Please install the driver that was delivered together with the card.
ERR_ADRRANGE	8h	8	One of the address ranges is disabled (fatal error), can only occur under Linux.
ERR_INVALIDHANDLE	9h	9	The used handle is not valid.
ERR_BOARDNOTFOUND	Ah	10	A card with the given name has not been found.
ERR_BOARDINUSE	Bh	11	A card with given name is already in use by another application.
ERR_EXPHW64BITADR	Ch	12	Express hardware version not able to handle 64 bit addressing -> update needed.
ERR_FWVERSION	Dh	13	Firmware versions of synchronized cards or for this driver do not match -> update needed.
ERR_LASTERR	10h	16	Old error waiting to be read. Please read the full error information before proceeding. The driver is locked until the error information has been read.
ERR_BOARDINUSE	11h	17	Board is already used by another application. It is not possible to use one hardware from two different programs at the same time.
ERR_ABORT	20h	32	Abort of wait function. This return value just tells that the function has been aborted from another thread. The driver library is not locked if this error occurs.
ERR_BOARDLOCKED	30h	48	The card is already in access and therefore locked by another process. It is not possible to access one card through multiple processes. Only one process can access a specific card at the time.
ERR_DEVICE_MAPPING	32h	50	The device is mapped to an invalid device. The device mapping can be accessed via the Control Center.
ERR_NETWORKSETUP	40h	64	The network setup of a digitizerNETBOX has failed.
ERR_NETWORKTRANSFER	41h	65	The network data transfer from/to a digitizerNETBOX has failed.
ERR_FWPOWERCYCLE	42h	66	Power cycle (PC off/on) is needed to update the card's firmware (a simple OS reboot is not sufficient !)
ERR_NETWORKTIMEOUT	43h	67	A network timeout has occurred.
ERR_BUFFERSIZE	44h	68	The buffer size is not sufficient (too small).
ERR_RESTRICTEDACCESS	45h	69	The access to the card has been intentionally restricted.
ERR_INVALIDPARAM	46h	70	An invalid parameter has been used for a certain function.
ERR_REG	100h	256	The register is not valid for this type of board.
ERR_VALUE	101h	257	The value for this register is not in a valid range. The allowed values and ranges are listed in the board specific documentation.
ERR_FEATURE	102h	258	Feature (option) is not installed on this board. It's not possible to access this feature if it's not installed.
ERR_SEQUENCE	103h	259	Command sequence is not allowed. Please check the manual carefully to see which command sequences are possible.
ERR_READABORT	104h	260	Data read is not allowed after aborting the data acquisition.
ERR_NOACCESS	105h	261	Access to this register is denied. This register is not accessible for users.
ERR_TIMEOUT	107h	263	A timeout occurred while waiting for an interrupt. This error does not lock the driver.
ERR_CALLTYPE	108h	264	The access to the register is only allowed with one 64 bit access but not with the multiplexed 32 bit (high and low double word) version.
ERR_EXCEEDSINT32	109h	265	The return value is int32 but the software register exceeds the 32 bit integer range. Use double int32 or int64 accesses instead, to get correct return values.
ERR_NOWRITEALLOWED	10Ah	266	The register that should be written is a read-only register. No write accesses are allowed.
ERR_SETUP	10Bh	267	The programmed setup for the card is not valid. The error register will show you which setting generates the error message. This error is returned if the card is started or the setup is written.
ERR_CLOCKNOTLOCKED	10Ch	268	Synchronization to external clock failed: no signal connected or signal not stable. Please check external clock or try to use a different sampling clock to make the PLL locking easier.
ERR_CHANNEL	110h	272	The channel number may not be accessed on the board: Either it is not a valid channel number or the channel is not accessible due to the current setup (e.g. Only channel 0 is accessible in interlace mode)
ERR_NOTIFYSIZE	111h	273	The notify size of the last spcm_dwDefTransfer call is not valid. The notify size must be a multiple of the page size of 4096. For data transfer it may also be a fraction of 4k in the range of 16, 32, 64, 128, 256, 512, 1k or 2k. For ABA and timestamp the notify size can be 2k as a minimum.
ERR_RUNNING	120h	288	The board is still running, this function is not available now or this register is not accessible now.
ERR_ADJUST	130h	304	Automatic card calibration has reported an error. Please check the card inputs.
ERR_PRETRIGGERLEN	140h	320	The calculated pretrigger size (resulting from the user defined posttrigger values) exceeds the allowed limit.
ERR_DIRMISMATCH	141h	321	The direction of card and memory transfer mismatch. In normal operation mode it is not possible to transfer data from PC memory to card if the card is an acquisition card nor it is possible to transfer data from card to PC memory if the card is a generation card.
ERR_POSTEXCDSEGMENT	142h	322	The posttrigger value exceeds the programmed segment size in multiple recording/ABA mode. A delay of the multiple recording segments is only possible by using the delay trigger!
ERR_SEGMENTINMEM	143h	323	Memsize is not a multiple of segment size when using Multiple Recording/Replay or ABA mode. The programmed segment size must match the programmed memory size.
ERR_MULTIPLEPW	144h	324	Multiple pulsewidth counters used but card only supports one at the time.
ERR_NOCHANNELPWOR	145h	325	The channel pulsewidth on this card can't be used together with the OR conjunction. Please use the AND conjunction of the channel trigger sources.
ERR_ANDORMASKOVRLAP	146h	326	Trigger AND mask and OR mask overlap in at least one channel. Each trigger source can only be used either in the AND mask or in the OR mask, no source can be used for both.
ERR_ANDMASKEDGE	147h	327	One channel is activated for trigger detection in the AND mask but has been programmed to a trigger mode using an edge trigger. The AND mask can only work with level trigger modes.
ERR_ORMASKLEVEL	148h	328	One channel is activated for trigger detection in the OR mask but has been programmed to a trigger mode using a level trigger. The OR mask can only work together with edge trigger modes.
ERR_EDGEPERMOD	149h	329	This card is only capable to have one programmed trigger edge for each module that is installed. It is not possible to mix different trigger edges on one module.
ERR_DOLEVELMINDIFF	14Ah	330	The minimum difference between low output level and high output level is not reached.
ERR_STARHUBENABLE	14Bh	331	The card holding the star-hub must be enabled when doing synchronization.

error name	value (hex)	value (dec.)	error description
ERR_PATPWSMALLEDGE	14Ch	332	Combination of pattern with pulsewidth smaller and edge is not allowed.
ERR_PCICHECKSUM	203h	515	The check sum of the card information has failed. This could be a critical hardware failure. Restart the system and check the connection of the card in the slot.
ERR_MEMALLOC	205h	517	Internal memory allocation failed. Please restart the system and be sure that there is enough free memory.
ERR_EEPROMLOAD	206h	518	Timeout occurred while loading information from the on-board EEProm. This could be a critical hardware failure. Please restart the system and check the PCI connector.
ERR_CARDNOSUPPORT	207h	519	The card that has been found in the system seems to be a valid Spectrum card of a type that is supported by the driver but the driver did not find this special type internally. Please get the latest driver from www.spectrum-instrumentation.com and install this one.
ERR_FIFOHWOVERRUN	301h	769	Hardware buffer overrun in FIFO mode. The complete on-board memory has been filled with data and data wasn't transferred fast enough to PC memory. If acquisition speed is smaller than the theoretical bus transfer speed please check the application buffer and try to improve the handling of this one.
ERR_FIFOFINISHED	302h	770	FIFO transfer has been finished, programmed data length has been transferred completely.
ERR_TIMESTAMP_SYNC	310h	784	Synchronization to timestamp reference clock failed. Please check the connection and the signal levels of the reference clock input.
ERR_STARHUB	320h	800	The auto routing function of the Star-Hub initialization has failed. Please check whether all cables are mounted correctly.
ERR_INTERNAL_ERROR	FFFFh	65535	Internal hardware error detected. Please check for driver and firmware update of the card.