M2i.xxxx LabVIEW Driver

**Driver for all M2i cards
and related digitizerNETBOX products**

**Installation, Libraries,
Data sorting, Examples,
Standard mode, FIFO mode**

English version **November 22, 2019**

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

General Information

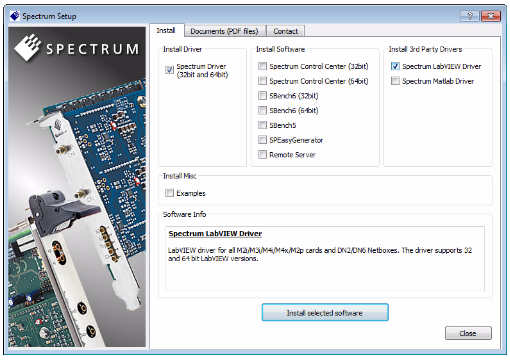
This driver is suitable for all cards of the M2i and M3i series as well as the related digitizerNETBOX and generatorNETBOX products from Spectrum. The driver supports all LabVIEW versions starting with LabVIEW 2015. The Spectrum LabVIEW driver supports Windows (32bit and 64bit) operating systems only, LabVIEW for Linux and LabVIEW RT are not supported. Please follow the install instructions to have the drivers properly installed in your system.

These examples are not tailored to the newer generation M4i, M4x or M2p cards. For these newer families please use the unified „spcm_xxxx" LabVIEW examples.

Installation

LabVIEW Driver Installation

Please follow these steps when installing the LabVIEW driver:

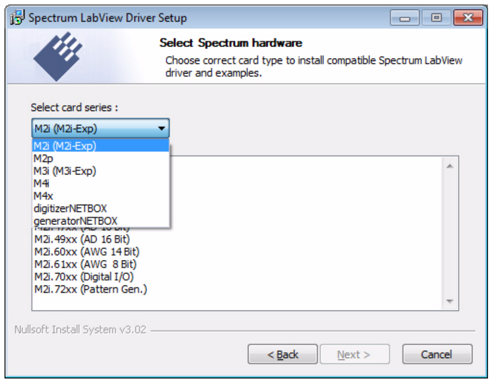


- Install the card(s) into the system as shown in the hardware manual
- Install the standard Windows driver as shown in the hardware manual
- Install the LabVIEW driver as explained below

The LabVIEW driver is delivered as a self extracting archive. You'll find the current driver on the USB-Stick delivered with the card. Please follow the USB-Stick menu to „Software Installation" -> „Spectrum LabVIEW driver" as shown on the right side.

It is also possible to install the LabVIEW driver manually selecting the install file with the Windows explorer. Please select the path:

<USB-Stick>:\Install\win\spcm_drv_labview_install.exe



and execute the installer file. The installer will guide you through the installation routine step by step.

At any time you can download the latest version from the Spectrum homepage https://spectrum-instrumentation.com/en/downloads/drivers

Please store the downloaded installer *.exe file somewhere on your system and start it from this location.

During the installation routine you will need to select which type of LabVIEW is installed on your computer (either a 32 bit or 64 bit version) and for what Spectrum products you want the examples to be installed for.

The LabVIEW driver files are installed per default in the user directory within the „my documents" folder as an extra directory:

- 32 bit LabVIEW: \Users\<WINDOWS_USERNAME>\Spectrum GmbH\SpcmLabVIEWDriver32
- 64 bit LabVIEW: \Users\<WINDOWS_USERNAME>\Spectrum GmbH\SpcmLabVIEWDriver64

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 has been updated January 2013. Drivers released before this date needed a separate installation license. Nowadays a separate license for the LabVIEW driver is no longer needed. You can download and install the latest LabVIEW driver at any time from the Spectrum homepage.

LabVIEW Driver Update

As the LabVIEW driver also uses the standard Windows drivers as a base, any updates on these drivers will improve the system and any changes are available under LabVIEW immediately. Updating the LabVIEW driver can simply be done by installation of the new LabVIEW driver archive.

General Information

Demo mode

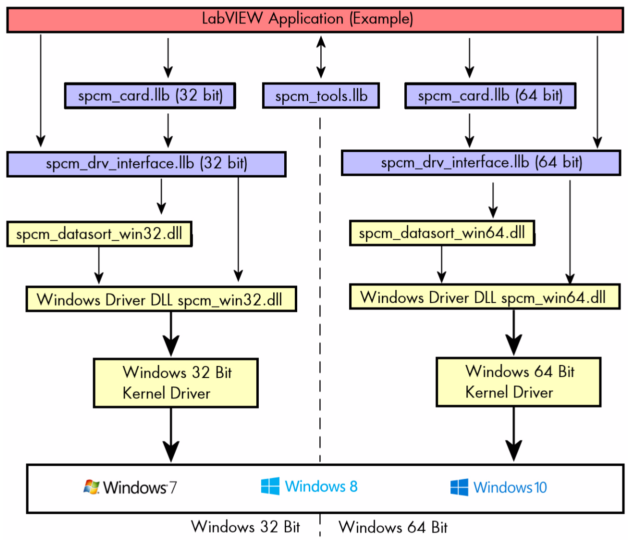
The LabVIEW driver runs fine with demo cards installed in your system. Please follow the steps in the hardware manual to see how you insert a simulated demo card as a virtual card into your system. Please keep in mind that the generated data is only simulated. The simulation and calculation of demo data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

Driver Structure

The driver itself consists of three LabVIEW libraries in either 32 bit or 64 bit version (shown in blue) and one additional DLL spcm_datasort_win32.dll or spcm_datasort_win64.dll (shown in yellow). All hardware access is routed through the standard Windows drivers and using the standard Windows kernel driver.

Access of the cards can be solely done by using the direct driver interface spcm_drv_interface.llb but using the more comfortable spcm_card.llb as shown in the examples is the much easier way.

The components of the Spectrum LabVIEW driver are:



spcm_win32.dll / spcm_win64.dll

This is the standard Windows driver as it is installed along with the kernel driver when the new hardware is detected in the system for the first time. The Windows driver can be updated from the Spectrum website at any time under www.spectrum-instrumentation.com. This driver is used by all software that will access the cards.
The driver library is available as 32 bit version (spcm_win32.dll) and 64 bit version (spcm_win64.dll).

spcm_datasort_win32.dll / spcm_datasort_win64.dll

This is a special helper DLL that is used by several Spectrum drivers for third-party products like LabVIEW or MATLAB. It handles the data access and offers some additional functions to sort data and allows also to re-calculate RAW data samples to true voltage values. This library also handles the FIFO mode and holds the application data buffer when FIFO mode is used. This DLL is also updated with the regular Windows driver updates.

spcm_drv_interface.llb

This LabVIEW library implements the complete driver interface between LabVIEW and the DLL. It mainly handles the driver handle and the error code and calls the different driver function inside the DLLs. The installer will automatically select the matching version for either 32 bit or 64 bit systems.

spcm_card.llb

This is an additional LabVIEW library that uses the functions of the driver interface spcm_drv_interface.llb and groups functions that contain together. The included VIs are more complex and offer an easy way to get started. All the spcm_card.llb VIs are explained in greater detail later on. The VIs included in this library cover about 99% of the driver functionality. The installer will automatically select the matching version for either 32 bit or 64 bit systems.

spcm_tools.llb

This library offers some simple helper functions to convert hardware details to readable strings like version or data conversion. Feel free to use these tools or to implement your own ones.

Not supported functions

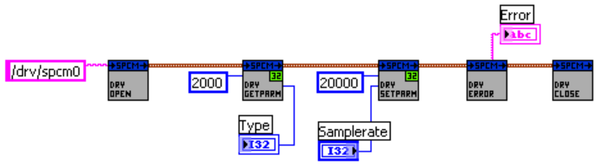
The spcm_card.llb library doesn't cover some special modes of single cards. It can also be that some functionality is added to the standard driver later on. As changing the VI interface would mean that none of the examples or customer applications would work any more after an update these VI interfaces are not updated in the future. Any further or later added content can be directly accessed using the driver functions that are located in the spcm_drv_interface.llb library.

Libraries

Library spcm_drv_interface.llb

Overview

All library functions get a cluster containing the driver handle and the current error code. The function is only executed if the error code is zero. This allows easy error routing without the need to check for driver errors after each call. An example is shown below:



On the left one sees the open function generating the cluster that is routed through all other driver calls until it stops in the close function.

In this example we open the driver, read out the card type (shown in the digital indicator „Type") and try to set the sampling rate from the digital control „Samplerate". The sampling rate register number is found in the hardware manual, it is „20000".

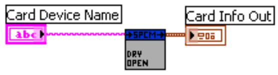
After these two function calls we check for the driver error and display the error message in the string indicator „Error".

Library Functions

The following library functions are available inside the library

spcm_hOpen.vi

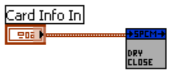
Calls the spcm_hOpen function of the driver. The open function tries to open the driver handle. It will return a valid card information cluster containing the card handle and the error code. This card information cluster is routed through all VIs of this library. The function can open real cards as well as demo cards with no difference calls.



Card Device Name	the device name to open. Under windows it can be any name finishing by a number giving the index of the card to open.
Card Info Out	the generated card information cluster. It contains the card handle and the error information. If the open function succeeded the error information will be zero.

spcm_vClose.vi

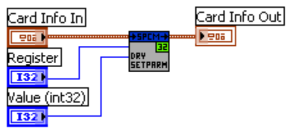
Calls the spcm_vClose function of the driver. The close function closes the card handle allowing further use of this card by other software. If the close function isn't called the card will be locked preventing any other software from accessing this card. The close function is automatically called when the DLL is unloaded. LabVIEW will unload the DLL when closing.



Card Info In	a valid card information cluster containing a valid card handle

spcm_dwSetParam_i32.vi

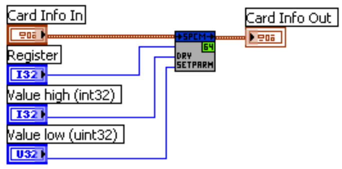
Calls the spcm_dwSetParam_i32 function of the driver. The function will set a software register with a 32 bit integer value. Please use the spcm_dwSetParam_i64m function if the value of the software register exceeds the 32 bit integer range.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to write. Please have a look at the hardware manual to see the valid software registers
Value (int32)	the value to write to the software register limited to 32 bit integer
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error

spcm_dwSetParam_i64m.vi

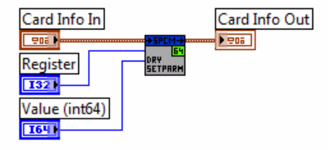
Calls the spcm_dwSetParam_i64m function of the driver. The function will set a software register with a 64 bit integer value. The value to write needs to be given in two 32 bit integer words.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to write. Please have a look at the hardware manual to see the valid software registers
Value high (int32)	the high 32 bit part of the 64 bit value to write to the software register. This part contains the sign bit
Value low (uint32)	the low 32 bit part of the 64 bit value to write. This part is unsigned.
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error

spcm_dwSetParam_i64.vi

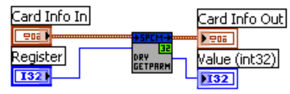
Calls the spcm_dwSetParam_i64 function of the driver. The function will set a software register with a 64 bit integer value.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to write. Please have a look at the hardware manual to see the valid software registers
Value (int64)	the value to write to the software register as a 64 bit integer
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error

spcm_dwGetParam_i32.vi

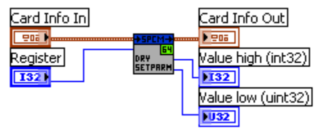
Calls the spcm_dwGetParam_i32 function of the driver. The VI reads a software register with up to 32 bit integer values. If the value exceeds the 32 bit integer range one is requested to use the spcm_dwGetParam_i64m.vi. Using the 32 bit function with a value exceeding the range will result in an error generated.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to read. Please have a look at the hardware manual to see the valid software registers
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error
Value (inbt32)	the current value of the software register limited to 32 bit integer

spcm_dwGetParam_i64m.vi

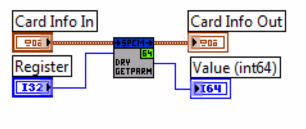
Calls the spcm_dwGetParam_i64m function of the driver. The VI reads a software register with 64 bit integer values. The value is split up in two parts and returned as two 32 bit integer values.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to read. Please have a look at the hardware manual to see the valid software registers
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error
Value high (int32)	the high 32 bit part of the 64 bit value that is read from the software register. This part contains the sign bit
Value low (uint32)	the low 32 bit part of the 64 bit value that is read. This part is unsigned

spcm_dwGetParam_i64.vi

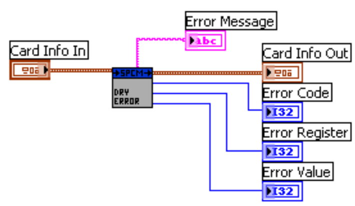
Calls the spcm_dwGetParam_i64 function of the driver. The VI reads a software register with 64 bit integer values.



Card Info In	a valid card information cluster containing a valid card handle
Register	the value of the software register to read. Please have a look at the hardware manual to see the valid software registers
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error
Value (int64)	the current value of the software register as a 64 bit integer

spcm_dwGetErrorInfo.vi

Calls the spcm_dwGetErrorInfo function of the driver. The function checks for an error code and reads out all error information and the error message if an error has occurred.



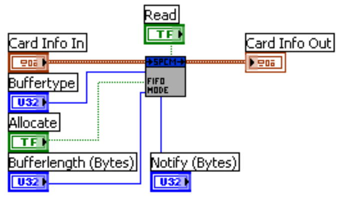
Card Info In	a valid card information cluster containing a valid card handle
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error
Error Message	the error message from the driver. This error message will help to examine which part of the setup was wrong
Error Code	the error code from the driver. If no error occurred this value is zero
Error Register	the register that generates the error. Please see the hardware manual for a cross reference list of the software registers
Error Value	the value that was written when the error occurred.

Data transfer library functions

The following functions are used for data transfer and FIFO mode control. These functions are located inside the helper DLL spcm_datasort_win32.dll.

dwSetupFIFOMode.vi

This VI handles the FIFO mode of the card and all transfers for timestamps and ABA data. Before starting FIFO transfer one has to allocate a FIFO buffer calling this setup function with the allocate flag set. After finishing the FIFO transfer a second call with the allocate flag cleared will delete the FIFO buffer again. Data can be accessed with the functions explained further below.

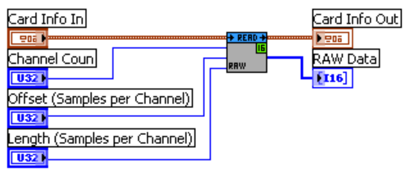


Card Info In	a valid card information cluster containing a valid card handle
Buffertype	the type of FIFO buffer to allocate, a 0 stands for data, a 1 for timestamps and a 2 for slow ABA data
Allocate	allocates the FIFO buffer if true and deletes the FIFO buffer if false
Bufferlength (Bytes)	the length of the FIFO buffer in bytes. Be sure to check the samples format to do the correct calculations on this value
Notify (Bytes)	the notify length in bytes. Every time after this number of bytes have been transferred an interrupt is generated and the user program is informed that new data is available. This value must be a multiple of 4k (4096). Please see the hardware manual for further information on the notify size
Read	the flag defines the direction of the following FIFO transfer
Card Info Out	a copy of the card information cluster input containing an error code if the DLL function has returned with an error

dwDataRead_raw16.vi

This VI reads the data from the card in raw format for all cards that have 16 bit wide samples (analog resolution > 8 bit) or digital cards with at least 16 digital channels. Using this function is the fasted way to get data into LabVIEW. Data is unsorted and in no way converted. Please check the hardware manual to see the data ordering in the RAW buffer.

This VI can be used with FIFO mode as well as with standard mode. In FIFO mode it will read out the next free block of data, in standard mode it will read some data directly from the on-board memory.



Card Info In	a valid card information cluster containing a valid card handle
Channel Count	the number of channels to be read. This value must match the number of channels that have been acquired!
Offset (Samples/Ch)	the offset from where the reading should start (standard mode). Offset is given in samples per channel, not in bytes
Length (Samples/Ch)	the length of the data to be read starting from offset (standard mode) or from the current buffer position (FIFO mode). The length value is given in samples per channel and must not exceed the previously acquired data
Card Info Out	a copy of the card information cluster input containing an error code if the dll function has returned with an error
RAW Data	An array containing the raw and unsorted data as 16 bit integer values.

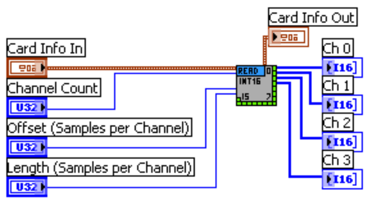
dwDataRead_raw8.vi

This VI does exactly the same as the above described but returning 8 bit wide raw data instead of 16 bit. Use this function for all analog cards with 8 bit resolution and digital cards with 8 channels only activated.

dwDataRead_i16.vi

The DataRead function reads data, sorts them and returns up to 16 arrays of data (only 4 shown in the picture on the right). Each array contains data of one analog channel or a bundle of 16 digital channels and can be directly used for display and further calculations.

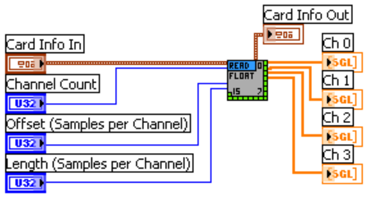
Data is stored as 16 bit integer values independent of the original data format. For 8 bit cards this means that memory storage space is doubled! Each 8 bit sample will be converted to 16 bit integer value.



Card Info In	a valid card information cluster containing a valid card handle
Channel Count	the number of channels to be read. This number must be equal to the number of installed channels on the card. Channels that are not acquired due to a different channel enable mask will be left empty
Offset (Samples/Ch)	the offset from where the reading should start (standard mode). Offset is given in samples per channel, not in bytes
Length (Samples/Ch)	the length of the data to be read starting from offset (standard mode) or from the current buffer position (FIFO mode). The length value is given in samples per channel and must not exceed the previously acquired data
Card Info Out	a copy of the card information cluster input containing an error code if the dll function has returned with an error
Ch0, Ch1,... Ch15	arrays containing the sorted data for one channel

dwDataRead_float.vi

The DataRead function reads data, sorts them, recalculates them to voltage and returns up to 16 arrays of data (only 4 shown in the picture on the right). Each array contains data of one analog channel and can be directly used for display and further calculations.



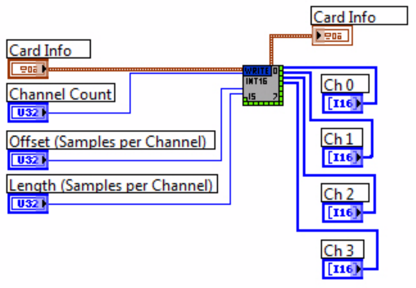
Data is stored as float values with single precision. The sorting functions recalculates the raw integer data to a true voltage level taking the programmed input range and also the programmed offset into account.

Please keep in mind that single values have 4 bytes for each sample. Acquiring 4 channels of 8 bit data with 10 MSamples of memory per each channel would result in a PC memory usage of 4 channels * 10 MSamples * 4 bytes = 160 MBytes when using this sorting function.

Card Info In	a valid card information cluster containing a valid card handle
Channel Count	the number of channels to be read. This number must be equal to the number of installed channels on the card. Channels that are not acquired due to a different channel enable mask will be left empty
Offset (Samples/Ch)	the offset from where the reading should start (standard mode). Offset is given in samples per channel, not in bytes
Length (Samples/Ch)	the length of the data to be read starting from offset (standard mode) or from the current buffer position (FIFO mode). The length value is given in samples per channel and must not exceed the previously acquired data
Card Info Out	a copy of the card information cluster input containing an error code if the dll function has returned with an error
Ch0, Ch1,... Ch15	arrays containing the sorted data for one channel, data format is single precision float containing the real voltage levels of the inputs

dwDataWrite_i16.vi

The DataWrite function writes data given as sorted arrays of int16 channels. Each channel is either an analog channel of up to 16 bit width or a bundle of up to 16 digital channels (2 bytes). Data is multiplexed inside the driver and written to hardware afterwards.



Card Info In	a valid card information cluster containing a valid card handle
Channel Count	the number of channels to be read. This number must be equal to the number of installed channels on the card. Channels that are not written due to a different channel enable mask will be left empty
Offset (Samples/Ch)	the offset where the writing should start (standard mode). Offset is given in samples per channel, not in bytes
Length (Samples/Ch)	the length of the data to be written starting from offset (standard mode) or from the current buffer position (FIFO mode). The length value is given in samples per channel and must not exceed the available amount of empty data space
Card Info Out	a copy of the card information cluster input containing an error code if the dll function has returned with an error
Ch0, Ch1,... Ch15	arrays containing the sorted data for one channel each

Library spcm_card.llb

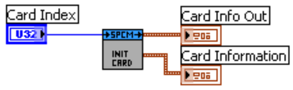
Overview

The spcm_card.llb library is the main library for accessing the Spectrum cards. All VIs route the standard card information shown on the right containing the card handle and the current error code. All VIs can simply be placed one after the other as none of these VIs execute their function if an error code is set.

Standard library functions

init card.vi

This VI is the main entrance point for the card. It must be called first to get a valid card handle. The VI tries to open the card that is given with the index and if successful it reads out some standard information from the card shown below as the card information cluster.



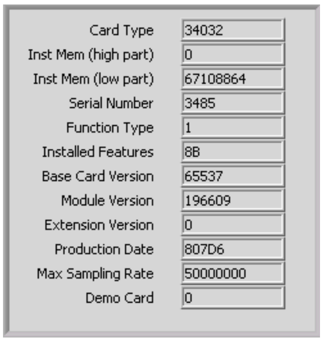
Each card can only be opened by one software at the time. Multiple calls of this initialization function with different index values will open multiple cards. Multiple calls with the same index value will result in an error as the card is opened and locked with the first call.

This function can open real cards as well as demo cards.

Card Index	the index of the card to open. All cards in the system are numbered beginning with a 0. Demo cards are handled a little different. If virtual demo cards are installed by software the card index will be ignored and the first call of the init function will return the first virtual demo card of the system
Card Info Out	A filled card info cluster that is routed through all the other functions. If initialization failed the error code will show an initialization error. The card info cluster is shown above in the overview
Card Information	A filled card information cluster containing all details that are common for all cards

Card information cluster

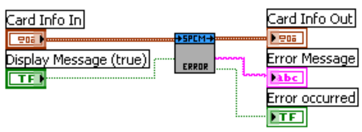
The cluster contains all common information for Spectrums M2i/M3i/M4i/M4x/M2p cards, as well as digitizerNETBOX and generatorNETBOX products. The information can be used to show card details in the software or to check the correct type or version.



Card Type	the type of card found at that position. Card types are listed in the hardware manual. You may use the translation function in the spcm_tools.llb library to show a real name for the card type
Inst Mem (high + low)	installed on-board memory in bytes, in the example the card has a memory of 64 MBytes installed
Serial Number	serial number of the card. The serial number is an unique identifier
Function Type	the card function type (like analog input, digital i/o), details can be found in the hardware manual, in our example the card is an analog input card (1)
Installed Features	shows all installed features on the card. The features are returned as a bitmask, each activated bit stands for one feature installed. In our example bit 4, 3, 1 and 0 are set meaning that feature ABA mode, Timestamp, Gated Sampling and Multiple Recording is installed on the card. All feature codes are explained in the hardware manual
Base card version	the version of the base card split in major and minor version. Please use the translation function from spcm_tools.llb to have a correct version display
Module version	version of the used front-end module, same format as above
Extension version	version of the extension module if one is installed, same format as above
Production date	the production week of the card, the lower 16 bit contain the year, the upper 16 bit the week. Please use the translation function from spcm_tools.llb to have the date printed in correct format
Max Sampling Rate	the maximum sampling rate of the card in hertz. In our example it is 50 MS/s (50000000 Hz). This is the absolute maximum sampling rate that may not be available with all channel combinations!
Demo Card	a simple flag indicating whether this is a virtual demo card or a real card (zero)

error check and message.vi

This VI is used to check the card info for an error and to display an error message if requested. To keep programming simple the VI also gives an error flag that can be directly used for case structures

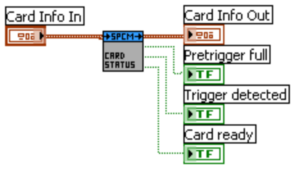


Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with cleared error information
Display Message	the flag selects whether the function should display an error message in case that an error occurred. As default this flag is true

Error Message	the error message string that can be used for own error display routines. Can be ignored if the error message is displayed by the VI itself
Error occurred	A flag indicating the an error has been found, error code is not zero. Can be directly used to drive case structures

read card status.vi

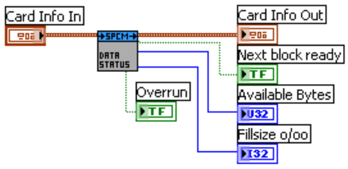
The VI reads the current card status and returns some flags indicating the status. The flags can be directly used to drive case structures or to end while loops.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Pretrigger full	acquistion cards only: the pretrigger area has been filled once, card is armed now and can detect trigger events
Trigger detected	a trigger event has been detected
Card ready	the acquisition/generation of data has been finished

read data status.vi

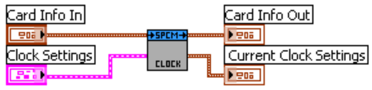
The VI reads the current status of the data transfer. This function is used together with the FIFO mode and controls the transfer and the current transfer status.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Next block ready	is true if a new block of data is ready. That means at least the programmed number of bytes are ready that have been programmed with the dwSetupFIFOMode call as notify size.
Available Bytes	returns the number of bytes that are available for the user and for the copy function
Fillsize o/oo	Gives the current fill size of the hardware FIFO in 1/1000

setup clock

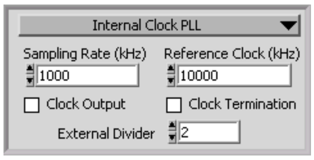
The VI programs the sampling clock and all clock related setup to the card. The clock settings are available as a cluster that is explained next.



Card Info In	a valid card info cluster containing driver handle and error information
Clock Settings	contains all clock related settings as explained below. All these settings are programmed to the card
Card Info Out	a copy of the card info cluster with the error output of this function
Current Clock Settings	contains the current clock settings that are read back from the driver. The cluster is documented below

Clock settings cluster

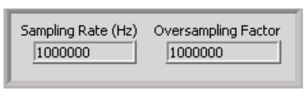
The cluster contains the complete clock setup and is also used throughout our examples. Not all of the settings are used for every clock mode. Please have a look at the hardware documentation to see details about the clock mode and the different setups.



Mode (on top)	selects one of the clock generating modes. The clock mode defines which of the other settings are used and which are ignored. In our examples we use the property nodes of this cluster and disable these settings depending on the currently selected clock mode.
Sampling rate (kHz)	defines the sampling rate for all internal clock modes in kHz (kS/s) and also for the reference clock mode. The driver set's the nearest matching sampling rate which can be read back using the current clock settings cluster described below
Clock Output	if enabled the clock connector outputs the currently used internal sampling clock. This output is only available if using internal sampling clock generation
Reference Clock (kHz)	defines the exact frequency of the reference clock that is fed into the external clock connector. This value is only used if the reference clock mode is selected
Clock Termination	if enabled the clock input termination is switched on. This feature is only available if an external clock or the reference clock mode is used
External Divider	divides externally fed in clock. This input is only available if one of the external clock modes with divider are selected

Current clock settings cluster

The cluster is returned by the setup clock.vi and holds information on the currently selected clock. As the clock generation has some limits due to PLL limitations and divider ranges the internal sampling rate may differ from what you have programmed. When programming the sampling rate the driver finds the nearest matching sampling rate.

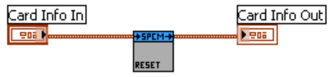


Sampling Rate (Hz)	returns the exact sampling rate in Hz which is now selected on the card. This sampling rate may differ from the one that has been programmed before
Oversampling Factor	fast analog acquistion cards have a minimum sampling rate that is limited by the used ADC. To allow slower sampling rates the driver programs oversampling which can be read out with this indicator

Commands

cmd reset

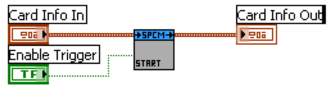
Performs a hardware and software reset of the card



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function

cmd start

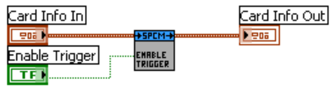
The card is started with the current setup that has to be programmed before using a valid combination of the setup VIs.



Card Info In	a valid card info cluster containing driver handle and error information
Enable Trigger	defines whether the trigger engine should be enabled directly with the start (default) or whether the trigger engine should be enabled with a separate enable trigger command
Card Info Out	a copy of the card info cluster with the error output of this function

cmd en/dis trigger

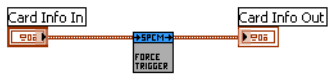
Enables or disables the trigger engine. No trigger detection is done as long as the trigger engine is disabled.



Card Info In	a valid card info cluster containing driver handle and error information
Enable Trigger	a true enables the trigger engine, a false disables it
Card Info Out	a copy of the card info cluster with the error output of this function

cmd force

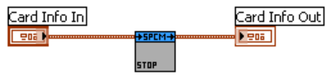
This VI sends a force trigger command that immediately triggers the card if it is waiting for a trigger event



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function

cmd stop

This VI stops the current run, the card data acquisition or generation is aborted



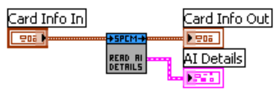
Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function

AI specific library functions

These VIs are used for analog input cards only. In general it is necessary to read out the AI features after initialisation to allow the setup of the analog input ranges according to these features.

read AI details.vi

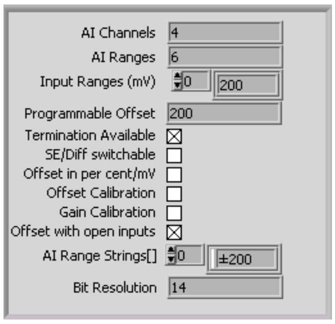
This VI reads out all analog input details from the card. These details are used throughout our examples to setup the analog input clusters according to the specific card that is installed in the system.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
AI Details	a cluster with complete details of the analog inputs as described below

Cluster AI details

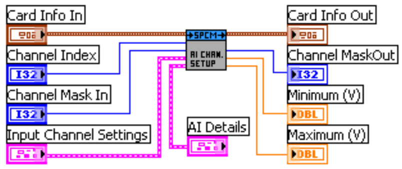
This cluster is returned by the „read AI deatils.vi“ and contains all information on the analog inputs. All these details are read from the driver. The cluster is mainly used to keep the examples and the programs universal as the analog inputs may differ from card to card in the number of input ranges, the availability of certain features or the offset programming mode.



AI Channels	the number of analog input channels (in this example 4 channels)
AI Ranges	the number of ranges for each channel. This is normally fixed for one card series but can differ if special options are ordered. The input ranges are therefore stored in the on-board eeprom and read out with this value and the array just following next in the description
Input Ranges (mV)	an array containing all input ranges as mV values that are available on your card. A 200 as shown in the example means an input range of +/-200 mV
Termination Available	if true each analog input has a software programmable 50 ohm termination available
SE/Diff switchable	if true each analog input can be changed from single-ended to differential by software command
Offset in per cent/mV	if true the offset is programmed in mV absolute, if false the offset is programmed in per cent of the input range
Offset calibration	if true the card has a complete on-board automatic offset calibration
Gain calibration	if true the card has a complete on-board automatic gain calibration
Offset with open inputs	if true the card has a on-board automatic offset calibration that needs all signals to be disconnected from the inputs for doing the offset calibration
AI Range Strings	contains a number of input range strings that can be directly used to fill the ring control of the analog input setup cluster as shown in the example further below
Bit Resolution	contains the analog resolution of the ADC, in our example the ADC has 14 bit resolution

setup AI channel.vi

This VI performs the complete analog input setup for one channel. It therefore get's a AI setup cluster and the number of the channel to perform. To keep the setup of the channel mask easy it will also add the correct channel mask bit to the routed channel mask. After calling all analog input setups the channel mask output of the last VI contains the correct channel mask to be set.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Index	the index of the channel which settings should be programmed. The channel indexing starts with zero!
Channel Mask In	the current channel mask that will be modified by the VI if the channel is activated. The first „setup AI channel“ call must be feeded with a zero and all following calls need to be feeded with the output of the last call
Channel Mask Out	the modified channel mask to be routed to the next call of „setup AI channel“
Input Channel Settings	the cluster with the channel setup as explained below
AI Details	the AI details cluster that was returned by the „read AI details“ VI. This cluster is absolutely necessary as this VI can handle all different card types and has to know which functions the card supports
Minimum, Maximum (V)	These outputs can be optionally used to scale a waveform graph. They contain the minimum and maximum value the input channel will generate as a voltage level. The calculation checks the selected input range as well as the selected user offset

Cluster Input Channel Settings

This cluster contains all analog input channel settings and is used together with the „setup AI channel“ VI. It support all possible settings that an analog input channel can have. It is recommended to adjust the controls of this cluster according to the analog input details as shown in our example.

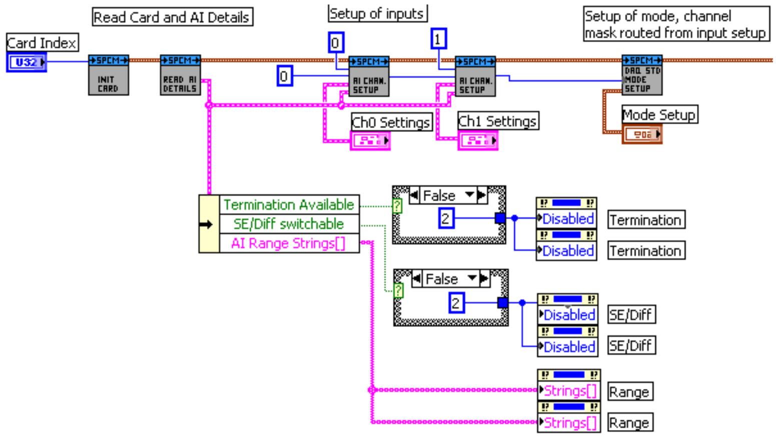
Enable	selects whether this channel should be acquired or not. This input is used by the „setup AI channel“ function to set up the channel mask
Range (top right)	selects the input range for the channel. In our example the input ranges are just numbered starting by zero. It is recommended to use the „AI range strings[]“ from the „AI details cluster“ to fill this ring element with valid setup
Termination (bottom left)	selects the input termination if the card supports software programmable input termination
SE/Diff (bottom middle)	switches the input from single-ended to differential if the card supports this feature by software
Offs	programs the input offset of the channel. Depending on the used card the offset can either be a per cent value of the input range or an absolute mV value

Example for setting up the AI input section

This example is an except from our LabVIEW examples. It shows how one reads out the AI details and sets up the universal analog input cluster to match the current card.

After doing the initialization and reading the AI input details we setup the parts of the input clusters. In our example we have 2 channels where we disable or enable the termination and the SE/Diff switch and where we set the range ring selector with the range strings returned by the „read AI details" VI.

The AI details are also routed to the „AI chan. setup" VI as we need them for the settings. The channel mask is initialized with zero and routed through these two VIs to the „DAQ std mode setup". The „AI chan. setup" VI is called for every channel that should be set and get's the channel number as an input (0 and 1 on top of the icons)

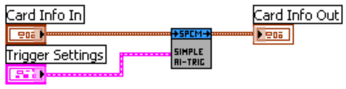


Finally the „DAQ std mode setup" set's the current mode and also programs the channel mask that has been modified by the two channel setup VIs.

The further VI calls are not shown in this example

setup simple AI trigger.vi

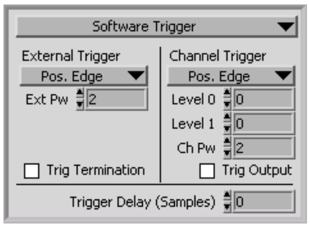
The VI is used to have a simple method for settings triggers. This VI is limited to one trigger source at the time. Doing more complex trigger setups can be done with the function „setup complex AI trigger" that is described next. Throughout the most examples we use this function as it's very easy to program and covers most of the trigger modes one would use.



Card Info In	a valid card info cluster containing driver handle and error information
Trigger Settings	the cluster containing the simple trigger settings. The cluster itself is described next
Card Info Out	a copy of the card info cluster with the error output of this function

Cluster Simple AI Trigger

This cluster contains the simple AI trigger setup. It covers all Spectrum analog input cards and therefore list's all channels that may be available with any Spectrum card. Please use only the channels that are available on your card as a trigger source as using another channel will result in an error message from the driver.

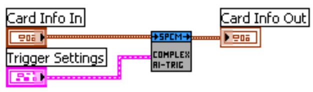


Please note that besides the trigger source on top of the window and the trigger delay all other settings are only used for certain trigger modes. All settings in the left column are only used if external trigger has been selected, all settings in the right column are only used if a channel trigger has been selected.

Trigger Source (top)	selects the single trigger source to be used. In our example the software trigger is selected
External Trigger Mode	selects the external trigger mode if the trigger source is set to external, otherwise this setting is ignored
Ext. Pw	selects the external pulsewidth if an external mode using pulsewidth counter has been selected
Trig Termination	switches the 50 ohm trigger termination if external trigger source has been selected
Channel Trigger Mode	selects the channel trigger mode if one of the channel trigger sources has been selected
Level 0	define the trigger level 0 (upper level) as integer value. Please check the re-calculation and the valid range of the trigger level in the hardware manual
Level 1	define the trigger level 1 (lower level) as integer value. Only available for certain channel trigger modes that need two trigger levels. Please check the hardware manual for details
Ch Pw	selects the channel pulsewidth in a channel trigger source is selected and a channel trigger mode using pulsewidth counter is active
Trig Output	enables the output of the trigger event on the connector if one of the channel trigger sources are selected
Trigger Delay	programs the trigger delay in samples. Is used for all trigger sources and trigger modes

setup complex AI trigger.vi

This VI can be used to set complete complex trigger for analog input cards. It allows to program all details of the trigger engine and is not limited by any pre-selected ring lists. It especially allows to do trigger combinations with OR and AND masks and also individually pulsewidth if the hardware supports this. For a more simple AI trigger setup pleas e have a look at the above „setup simple AI trigger"



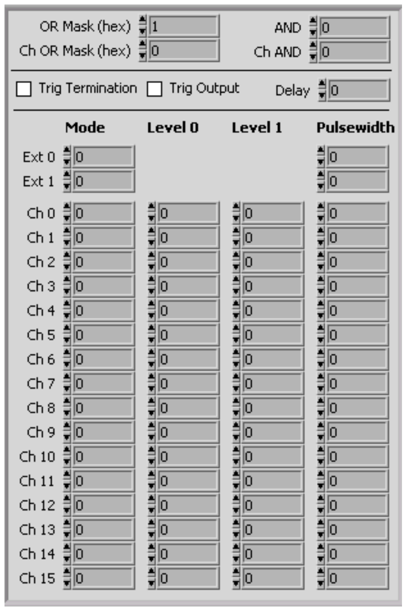
The VI can be used for all Spectrum analog input card as it is valid up to 16 analog channels. Please be sure not to activated channels that are not present on your card. Doing so will result in an driver error message.

Please have a look at the example complex_trigger_scope.vi to see an example how to use this VI and also having some buttons for different example setups of this VI. The example is described in greater detail in the next chapter.

Card Info In	a valid card info cluster containing driver handle and error information
Trigger Settings	the cluster containing the complex trigger settings. The cluster itself is described next
Card Info Out	a copy of the card info cluster with the error output of this function

Cluster Complex AI Trigger

All details are directly inserted as raw mode values. Please check the hardware manual to see the valid settings. If the setup contains errors the driver will report these.



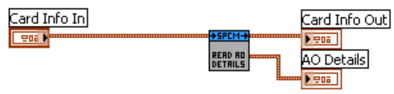
OR Mask (hex)	the global OR mask as found in the register SPC_TRIG_ORMASK containing the enable bits for software and external trigger sources
AND	the global AND mask as found in the register SPC_TRIG_AND containing the enable bits for software and external trigger sources
CH OR Mask (hex)	the channel OR mask. Each analog channel corresponds to one bit-The register behind this control is SPC_TRIG_CH_ORMASK. Please be sure to have a valid trigger mode selected for every channel that has been enabled in this mask
CH AND	the channel AND mask as written to software register SPC_TRIG_CH_ANDMASK0
Trig Termination	switches the 50 ohm trigger termination if an external trigger source has been selected
Trig Output	enables the output of the trigger event on the connector if only channel trigger sources are selected
Delay	programs the trigger delay in samples. Is used for all trigger sources and trigger modes
Ext 0/1 Mode	selects the external trigger mode if this trigger source has been activated in OR Mask or AND Mask
Ext 0/1 Pulsewidth	selects the external pulsewidth if an external mode using pulsewidth counter has been selected
Ch 0..15 Mode	sets the channel trigger mode if this trigger source has been activated in the Channel OR Mask or Channel AND Mask
Ch 0..15 Level 0	Sets the upper trigger level for this channel trigger
Ch 0..15 Level 1	Sets the lower trigger level for this channel trigger if the selected trigger mode uses two trigger levels
Ch 0..15 Pulsewidth	Sets the pulsewidth for this channel trigger if the selected trigger mode uses a pulsewidth counter. Please check whether your hardware supports multiple pulsewidth counters or only one global before programming multiple pulsewidth

AO specific library functions

These VIs are used for analog output cards (arbitrary waveform generators) only. In general it is necessary to read out the AO features after initialisation to allow the setup of the analog outputs according to these features.

read AO details.vi

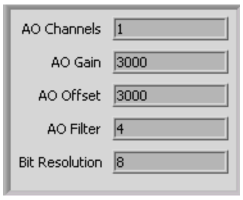
This VI reads out all analog output details from the card. These details are used throughout our examples to setup the analog output clusters according to the specific card that is installed in the system.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
AO Details	a cluster with complete details of the analog outputs as described below

Cluster AO details

This cluster is returned by the „read AO deatils.vi“ and contains all information on the analog outputs. All these details are read from the driver. The cluster is mainly used to keep the examples and the programs universal as the analog outputs may differ from card to card .



AO Channels — the number of analog output channels (in this example 1 channel)

AI Gain — the maximum bipolar output gain in mV. In our example the card can have up to +/- 3000 mV of output swing.

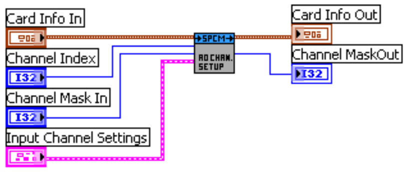
AO Offset — the maximum bipolar programmable analog offset of each channel in mV

AO Filter — the number of programmable output filters. Please have a look in the hardware manual to see which edge frequencies each filter has on your specific card.

Bit Resolution — contains the analog resolution of the DAC, in our example the DAC has 8 bit resolution

setup AO channel.vi

This VI performs the complete analog output setup for one channel. It therefore get's a AO setup cluster and the number of the channel to perform. To keep the setup of the channel mask easy it will also add the correct channel mask bit to the routed channel mask. After calling all analog ouput setups the channel mask output of the last VI contains the correct channel mask to be set.



Card Info In — a valid card info cluster containing driver handle and error information

Card Info Out — a copy of the card info cluster with the error output of this function

Channel Index — the index of the channel which settings should be programmed. The channel indexing starts with zero!

Channel Mask In — the current channel mask that will be modified by the VI if the channel is activated. The first „setup AO channel“ call must be feeded with a zero and all following calls need to be feeded with the output of the last call

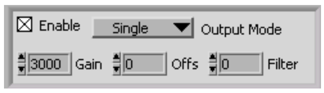
Channel Mask Out — the modified channel mask to be routed to the next call of „setup AO channel“

Output Channel Settings — the cluster with the channel setup as explained below

AO Details — the AO details cluster that was returned by the „read AO details“ VI. This cluster is absolutely necessary as this VI can handle all different card types and has to know which functions the card supports

Cluster Output Channel Settings

This cluster contains all analog output channel settings and is used together with the „setup AO channel“ VI. It support all possible settings that an analog output channel can have. It is recommended to adjust the controls of this cluster according to the analog output details as shown in our example.



Enable — selects whether this channel should be acquired or not. This input is used by the „setup AO channel“ function to set up the channel mask

Output Mode — defines the output mode of the selected channel. Depending on the card version and the channel selection some special output modes are available. The output modes are described in detail in the hardware manual.

Gain — selects the programmed bipolar output gain as mV.

Offs — selects the programmed output offset as mV.

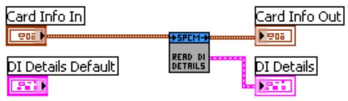
Filter — selects the index of the used output filter for this channel.

DI specific library functions

These VIs are used for digital input cards only.

read DI details.vi

This VI reads out all digital input details from the card. These details are used throughout our examples to setup the input clusters according to the specific card that is installed in the system.



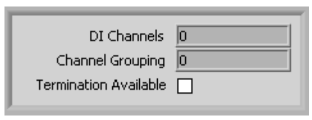
Card Info In — a valid card info cluster containing driver handle and error information

Card Info Out — a copy of the card info cluster with the error output of this function

DI Details — a cluster with complete details of the digital inputs as described below

Cluster DI details

This cluster is returned by the „read DI deatils.vi“ and contains all information on the digital inputs. All these details are read from the driver. The cluster is mainly used to keep the examples and the programs universal as the digital inputs may differ from card to card in the number of channels or the channel grouping.

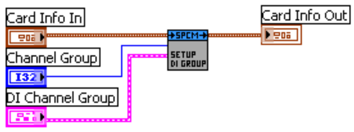


AI Channels — the number of digital input channels, each digital line is counted as one channel

Channel Grouping	the grouping of channels for the termination. A „16“ means that 16 digital channels are set together for the termination. When using the DI input the setup is used for these groups
Termination Available	the channel groups have a software programmable input termination

setup DI channel.vi

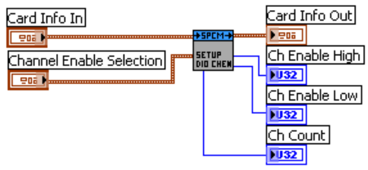
This VI performs the digital input setup for one group of channels. The grouping can be read out using the above mentioned „read DI details“ VI. It therefore get’s a DI setup cluster and the index of the channel group to setup.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Group	the index of the channel group to setup. If channel grouping is set to „16“ index 0 will work for channel 0 to 15, index 1 will work for channel 16 to 31 and so on.
DI Channel Group	the channel group setup cluster that contains all settings for one channel group. The channel group cluster only consists of the termination flag which activates the input channel termination for the complete channel group.

Setup DIO channel enable.vi

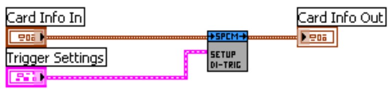
The VI is used to setup the channel enable mask for all digital input and output cards. Therefore it gets a cluster with settings, recalculates them, sets the channel enable and returns some recalculated information for further use within the LabVIEW program.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Enable Select.	the cluster contains a ring element listing all possible channel enable setups for digital cards. However not all cards support all settings. Therefore please check against the hardware manual to see which of these channel enable settings your card supports.
Ch Enable High	the upper 32 bit of the resulting channel enable mask for use with further sub-VIs
Ch Enable Low	the lower 32 bit of the resulting channel enable mask for use with further sub-VIs
Ch Count	the total number of activated channels

Setup DI trigger.vi

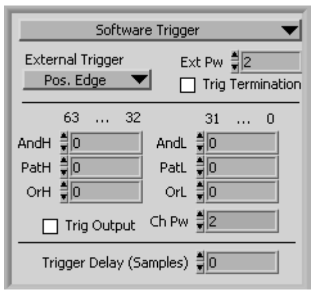
This VI can be used to set the trigger for digital input cards. It allows to program all different trigger modes and all pattern masks. Please be sure not to activated channels that are not present on your card. Doing so will result in an driver error message.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Trigger Settings	the cluster containing the trigger settings. The cluster itself is described next

Cluster DI Trigger

All details inserted here are directly written to the driver with no further check. Invalid settings will be reported by the driver. Please check the hardware manual for further information on the correct setup of the trigger masks. All mask and pattern values are inserted as hexadecimal values to allow an easy overview on the activated channels. The values are splitted in a upper and a lower part each with 32 bit width.



Trigger Source (top)	selects the single trigger source to be used. In our example the software trigger is selected
External Trigger Mode	selects the external trigger mode if the trigger source is set to external, otherwise this setting is ignored
Ext. Pw	selects the external pulsewidth if an external mode using pulsewidth counter has been selected
Trig Termination	switches the trigger termination for the external trigger source
AndH, AndL	selects the AND mask for any pattern trigger mode. Each bit corresponds to one digital channel. If a bit is set to 1 this channel is used for the pattern detection and is compared to the pattern information given in the PatH/PatL mask
PatH, PatL	selects the pattern for which the digital inputs are scanned. Each bit corresponds to one digital channel. If a bit is set to 1 this channel is checked for high level, on setting 0 the channel is checked for low level. The pattern is only checked if the channel is also activated using the AND mask.
OrH, OrL	selects the OR mask for any edged pattern trigger mode. Each bit corresponds to one digital channel. Each activated channel is checked for the occurrence of and edge. The edge itself is defined in the trigger source
Ch Pw	selects the channel pulsewidth in a channel trigger source is selected and a channel trigger mode using pulsewidth counter is active
Trig Output	enables the output of the trigger event on the connector if one of the channel trigger sources are selected

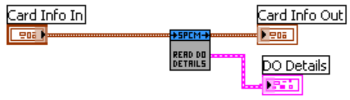
Trigger Delay	programs the trigger delay in samples. Is used for all trigger sources and trigger modes

DO specific library functions

These VIs are used for digital output cards only.

read DO details.vi

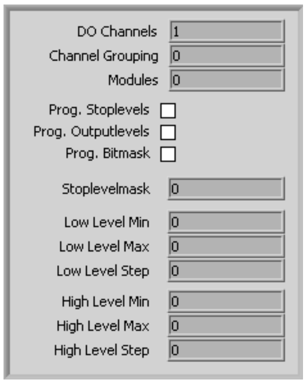
This VI reads out all digital output details from the card. These details are used throughout our examples to setup the digital output clusters according to the specific card that is installed in the system.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
DO Details	a cluster with complete details of the digital outputs as described below

Cluster DO details

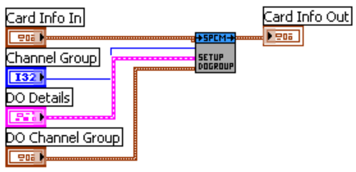
This cluster is returned by the „read DO deatils.vi" and contains all information on the digital outputs. All these details are read from the driver. The cluster is mainly used to keep the examples and the programs universal as the digital outputs may differ from card to card.



DO Channels	the number of digital output channels, each digital line is counted as one channel
Channel Grouping	the grouping of channels for the setup. A „16" means that 16 digital channels are set together for the setup. When using the DO input the setup is used for these groups
Modules	the number of installed modules on the card (1 or 2). One needs to know this number as the stoplevels are programmed per each module and not per channel group.
Prog. Stoplevels	software programmable stoplevels are available. The available stoplevels are found in the Stoplevelmask.
Prog. Outputlevels	software programmable output levels are available. The restrictions of the output levels are found in the Low/High Level Min/Max/Step information.
Prog. Bitmask	output can be enabled bitwise via software.
Stoplevelmask	shows as a mask all stop levels that can be programmed
Low Level Min	shows the minimum low level in mV that can be programmed
Low Level Max	shows the maximum low level in mV that can be programmed
Low Level Step	shows the stepsize of the low level in mV that can be programmed
High Level Min	shows the minimum high level in mV that can be programmed
High Level Max	shows the maximum high level in mV that can be programmed
High Level Step	hows the stepsize of the high level in mV that can be programmed

Setup DO channel group.vi

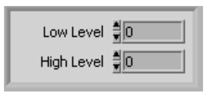
This VI performs the complete digital output setup for one channel group. It therefore gets the DO Details as returned by the above mentioned function and also a DO Channel Group Setup for the setup itself.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Group	the index of the channel group to setup. If channel grouping is set to „16" index 0 will work for channel 0 to 15, index 1 will work for channel 16 to 31 and so on.
DO Details	The Do details of the card as returned by the „read DO details" function
DO Channel Group	the channel group setup cluster that contains all settings for one channel group. The channel group cluster is explained next

Cluster DO Channel Group Settings

This cluster contions the setup for one group of digital output channels. The cluster is used by the „setup DO channel group" function



Low Level	the output low level in mV for this group
High Level	the output high level in mV for this group

The VI can be used for all Spectrum analog input card as it is valid up to 16 analog channels. Please be sure not to activated channels that are not present on your card. Doing so will result in an driver error message.

Please have a look at the example complex_trigger_scope.vi to see an example how to use this VI and also having some buttons for different example setups of this VI. The example is described in greater detail in the next chapter.

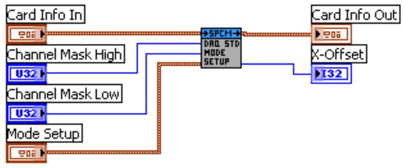
Card Info In	a valid card info cluster containing driver handle and error information
Trigger Settings	the cluster containing the complex trigger settings. The cluster itself is described next
Card Info Out	a copy of the card info cluster with the error output of this function

Acquisition specific library functions

These VIs are used for setting up all acquisition modes. As Standard mode and FIFO mode differ from the possible settings there are separate VIs for these two modes. Please keep in mind that the VIs allow the setup of all acquisition modes even if the mode (like Multiple Recording) is not installed in the hardware. Setting up such a mode in this case will result in a driver error message.

setup DAQ standard.vi

This VI programs all standard acquisition modes and programs all related settings to this mode. Either the „setup DAQ standard" or the „setup DAQ FIFO" VI needs to be used in one LabVIEW program.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Mask High	upper 32 bit of channel mask for all cards that have more than 32 channels on-board (like some digital I/O cards), can be left unconnected for all cards that have less than 32 channels
Channel Mask Low	lower 32 bit of the channel mask. Each channel corresponds to one bit of the mask. This channel mask defines which channels are used for the next acquisition. Please see the hardware manual to see which restrictions are given for the channel mask selection
M ode Setup	a cluster containing the mode setup as show below
X-Offset	the x-offset in samples that can be used to scale a waveform graph correctly. The offset is given in relation to the trigger event

Cluster Standard Mode Setup

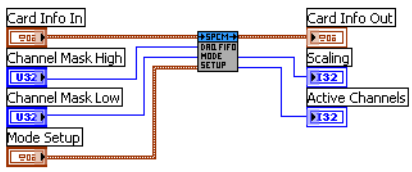
This cluster is used to feed the „setup DAQ standard.vi". It contains all standard mode setup. Depending on the selected mode some of the settings are not used. Please have a look at the scope example explained in the next chapter to see a way how to disable these settings depending on the selected mode.

Mode (top left)	selects the standard acquisition mode. Please be sure that the selected mode is installed on your hardware before selecting it
Mem	selects the on-board memory in samples per channel that is used for the next acquisition
Seg	selects the segment size, only valid if Multiple Recording or ABA mode is selected
Post	selects the posttrigger in samples per channel. Depending on the selected mode this value has a little different meaning: Singleshot: number of samples to acquire after detection of trigger event Multiple Recording, ABA mode: number of samples to acquire after trigger event for each segment Gated Sampling: number of samples to acquire after detection of gate-end signal
Pre	number of samples to acquire before the gate-start signal, therefore only valid if Gated Sampling is selected
ABA	ABA mode only: divides the current sampling rate to form the slow ABA clock to acquire the A-samples

setup DAQ FIFO.vi

This VI programs all FIFO acquisition modes and programs all related settings to this mode. Either the „setup DAQ standard" or the „setup DAQ FIFO" VI needs to be used in one LabVIEW program.

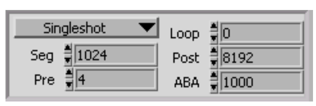
Most settings are similar to the „setup DAQ standard". Please look above for further information on these settings.



Scaling	returns a scaling factor to scale bytes to samples per channel. This scaling factor can be used for the dwSetupFIFOMode VI. If for example 2 channels are active, each with 12 bit resolution, the scaling factor will be 4 as one needs 4 bytes in total to store 1 sample per channel
Active Channels	returns the number of active channels to allow easy multiplexing and de-multiplexing

Cluster FIFO Mode Setup

This cluster contains all DAQ FIFO mode related settings:



Mode (top left)	selects the FIFO acquisition mode. Please be sure that the selected mode is installed on your hardware before selecting it
Loop	selects the number of segments/gates to acquire, leave zero if FIFO should run endless

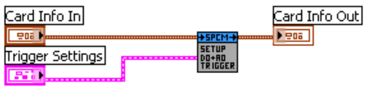
Seg	selects the segment size for Multiple Recording and ABA mode, for singleshot it forms together with Loop the total data length to acquire
Post	selects the posttrigger in samples per channel: Multiple Recording, ABA mode: number of samples to acquire after trigger event for each segment Gated Sampling: number of samples to acquire after detection of gate-end signal
Pre	number of samples to acquire before the gate-start signal, therefore only valid if Gated Sampling is selected
ABA	ABA mode only: divides the current sampling rate to form the slow ABA clock to acquire the A-samples

Output (generation) specific library functions

These VIs are used for setting up all generation modes and generation specific setup. As Standard mode and FIFO mode differ from the possible settings there are separate VIs for these two modes. Please keep in mind that the VIs allow the setup of all generation modes even if the mode (like Multiple Replay) is not installed in the hardware. Setting up such a mode in this case will result in a driver error message.

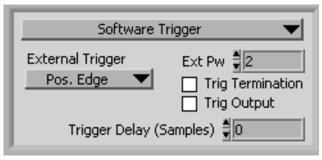
setup output trigger.vi

Defines the trigger mode for all output cards (analog and digital output cards and arbitrary waveform generators).



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Trigger Settings	a cluster that contains the complete trigger setup for the output mode. The cluster is described in detail below.

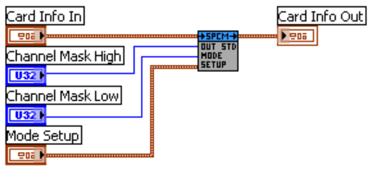
Cluster Output Trigger Settings



Trigger Source (top)	selects the single trigger source to be used. In our example the software trigger is selected
External Trigger Mode	selects the external trigger mode if the trigger source is set to external, otherwise this setting is ignored
Ext. Pw	selects the external pulsewidth if an external mode using pulsewidth counter has been selected
Trig Termination	switches the trigger termination for the external trigger source
Trig Output	enables the output of the trigger event on the connector if one of the channel trigger sources are selected
Trigger Delay	programs the trigger delay in samples. Is used for all trigger sources and trigger modes

setup output standard mode.vi

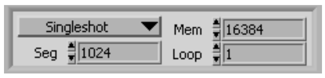
This VI programs all standard output (generation) modes and programs all related settings to this mode. Either the „setup output standard“ or the „setup output FIFO“ VI needs to be used in any LabVIEW program that performs output.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Channel Mask High	upper 32 bit of channel mask for all cards that have more than 32 channels on-board, can be left unconnected for all cards that have less than 32 channels
Channel Mask Low	lower 32 bit of the channel mask. Each channel corresponds to one bit of the mask. This channel mask defines which channels are used for the next acquisition. Please see the hardware manual to see which restrictions are given for the channel mask selection
Mode Setup	a cluster containing the mode setup as shown below

Cluster Standard Mode Setup

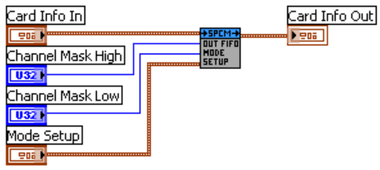
This cluster is used to feed the „setup output standard.vi“. It contains all standard mode setup. Depending on the selected mode some of the settings are not used. Please have a look at the examples explained in the next chapter to see a way how to disable these settings depending on the selected mode.



Mode (top left)	selects the standard output mode. Please be sure that the selected mode is installed on your hardware before selecting it
Mem	selects the on-board memory in samples per channel that is used for the next generation
Seg	selects the segment size, only valid if Multiple Recording is selected
Loop	Defines the number of loops to output. A zero stands for endless looping, a 1 for one loop until the programmed memory size is one time completely replayed. The meaning of this value differs a little depending on the selected mode: Singleshot: Defines the number of singleshots that are performed. Each detected trigger event will generate one singleshot until the loop counter expires. Continuous: Defines the number of loops the programmed memory is replayed after one trigger event. Multiple Replay: Values > 1 defines the number of segments that are replayed. Each segemnt will be replayed after detection of a new trigger event. Gated Replay: Values > 1 defines how many gate segments are replayed

setup output FIFO.vi

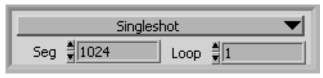
This VI programs all FIFO output modes and programs all related settings to this mode. Either the „setup output standard" or the „setup output FIFO" VI needs to be used in any LabVIEW program that performs output.



All settings of this vi are similar to the „setup output standard" Please look above for further information on these settings.

Cluster FIFO Mode Setup

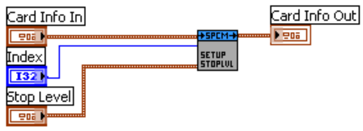
This cluster contains all output FIFO mode related settings:



Mode (top)	selects the FIFO output mode. Please be sure that the selected mode is installed on your hardware before selecting it
Loop	selects the number of segments/gates to output, leave zero if FIFO should run endless
Seg	selects the segment size for Multiple Recording, for singleshot it forms together with Loop the total data length to output

setup output stop level.vi

This VI defines the levels that are outputted after the normal output has finished or between the segments of the Multiple Replay mode or between the gate segments of the Gated Replay mode. On digital output cards the stoplevel can be set individually for each installed module. On analog output cards the stop level can be set for each analog channel. The programmed stoplevel has to be one of the available stoplevels defined in the AO/DO Details vi.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Index	the part for which the stop level should be programmed. On digital cards this is the index of the module one whishes to change the stop levels. On analog cards this is the index of the channel.
Stop Level	a cluster containing the stop level as a ring element with all possible stop levels inside.

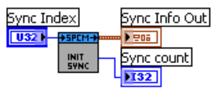
Synchronization specific library functions

These VIs are used for setting up all synchronization setup. These VIs need the option Star-Hub installed on one of the cards in the system. Without this option none of the functions will work.

These two VIs are the only ones that are needed to set up the synchronization. All the remaining setup is done via the standard VIs. The trigger modes that are programmed are automatically combined inside the Star-Hub. That means programming an OR trigger on card0 and an OR trigger on card 1 automatically sets these two cards as OR'd inside the Star-Hub.

init sync.vi

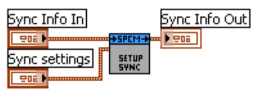
This VI must be called when a Star-Hub should be used to synchronize the installed cards. The VI tries to open the Star-Hub that is given with Sync Index and if successful it gives back the Sync Info Out which is similar to the Card Info Out and how many cards are connected to it.



Sync Index	the index of the Star-Hub. Use this if there is more than one Star-Hub in the system (standard is "0": the first Star-Hub).
Sync Info Out	a filled Star-Hub Info Cluster. Similar to the Card Info Cluster it contains the Star-Hub Handle and Error Code.
Sync Count	the number of Cards connected to the Star-Hub.

setup sync.vi

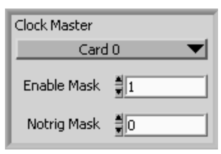
This VI programs the Star-Hub settings.



Sync Info In	a valid Star-Hub info cluster containing driver handle and error information.
Sync Settings	the synchronization setup cluster as described below.
Sync Info Out	a copy of the Star-Hub info cluster with the error output of this function.

Cluster Sync settings

The cluster that is used for the above documented setup sync.vi. The cluster contains the complete synchronization setup.



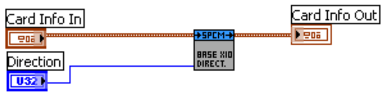
Clock Master	selects the card that acts as clock master for the complete synchronization
Enable Mask	a bit mask that enables or disables all the cards that should participate on the synchronization.
Notrig Mask	a bit mask to exclude the cards that shouldn't be triggered by the Star-Hub trigger but use their own local trigger engine.

Option BaseXIO specific library functions

These VIs are used for programming the BaseXIO option. The VIs will only work if that option is installed on your card. The BaseXIO option is running completely independent from the main card function and can be called asynchronously at any time.

BaseXIO direction.vi

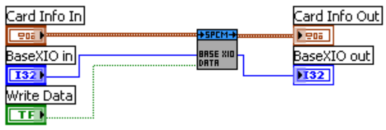
The VI sets the data direction for the BaseXIO option. The direction of each 4 channels can be set separetely. Please see the hardware manual for the details



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
Direction In	the direction word

BaseXIO data.vi

The VI programs the BaseXIO outputs and reads the inputs.



Card Info In	a valid card info cluster containing driver handle and error information
Card Info Out	a copy of the card info cluster with the error output of this function
BaseXIO in	data to be put out. To output data the corresponding direction bits have to be set to output
WriteData	flag that enables the output. If disabled data is only read and no data is written to the driver
BaseXIO out	the data of the BaseXIO inputs. If channels are set to output this value contains the prior written data

Library spcm_tools.llb

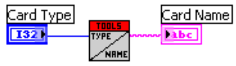
Overview

This library offers some simple helper functions to convert values used by our driver to readable strings.

Library Functions

spcm_translate_card_type.vi

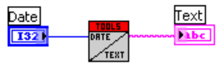
This VI translates the card type from an 32 bit integer value as returned by the driver to a card name string of the form M2i.2031.



Card Type	the card type as returned by the driver
Card Name	the card name as listed in our documentation and the order information

spcm_translate_date.vi

The VI translates the date code from a 32 bit integer value to a readable string of the form „week 12 of 2006“. The date code contains week and year put together in one 32 bit value.



Date	the date code as returned by the driver for production date or calibration date
Text	the formatted date text

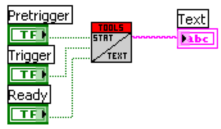
spcm_translate_version.vi

The VI translates the version code from a 32 bit integer value to a readable string in the form „V2.1“. The version code contains major and minor version number put together in one 32 bit integer value.

Version	the version code as returned by the driver for base card version, module version or extension version
Text	the formatted version text

spcm_translate_status.vi

The VI translates the status bit information into a readable status string that is used throughout our examples to show the current status of the card. The translate function can directly be connected to the status read function „read card status“

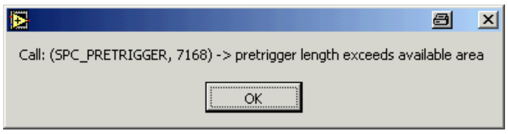


Pretrigger	pretrigger flag from read card status
Trigger	trigger flag from read card status
Ready	ready flag from read card status
Text	the formatted status text

Analog Input (AI) examples

This chapter gives you a brief overview of the examples that come together with the driver. Please keep in mind that these are only examples to show how the driver can be programmed. Although most of these examples can also be used as complete and comfortable stand-alone programs that wasn't our intention. Therefore there might be some limits in the examples and some settings are not checked on LabVIEW example level but only on the level of the standard driver.

Encountering an error message as shown on the right is not a bug of the LabVIEW driver or the example but it is simply a setup that isn't valid. Please check all details of the hardware manual to see what was going wrong here. In our example one tries to use Multiple Recording and exceeds the available pretrigger length. The register name (SPC_PRETRIGGER) gives you a clue where to search inside the hardware manual.



The examples are delivered „as is" and they're not intended to become more powerful applications as this makes the understanding of the examples very difficult. If you encounter any general problems with the examples please contact our support.

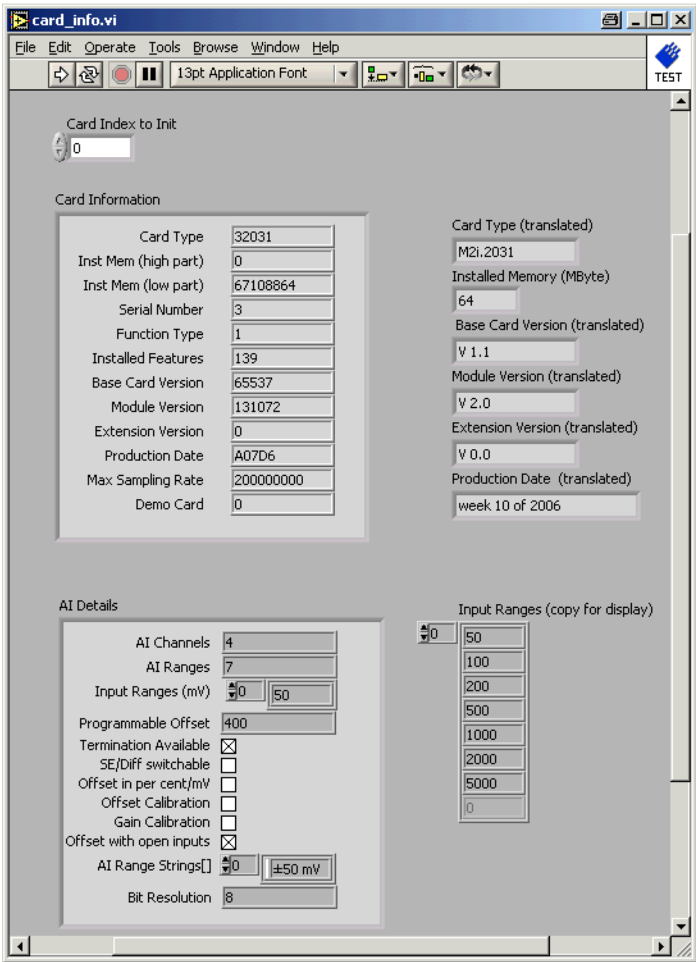
Please feel free to use the examples and to modify them for your own applications. Please keep in mind that we do not support any modified examples!

Card Information (card_info.vi)

This VI shows how to initialize the card and how to read out further information on the card. The VI does the following steps:

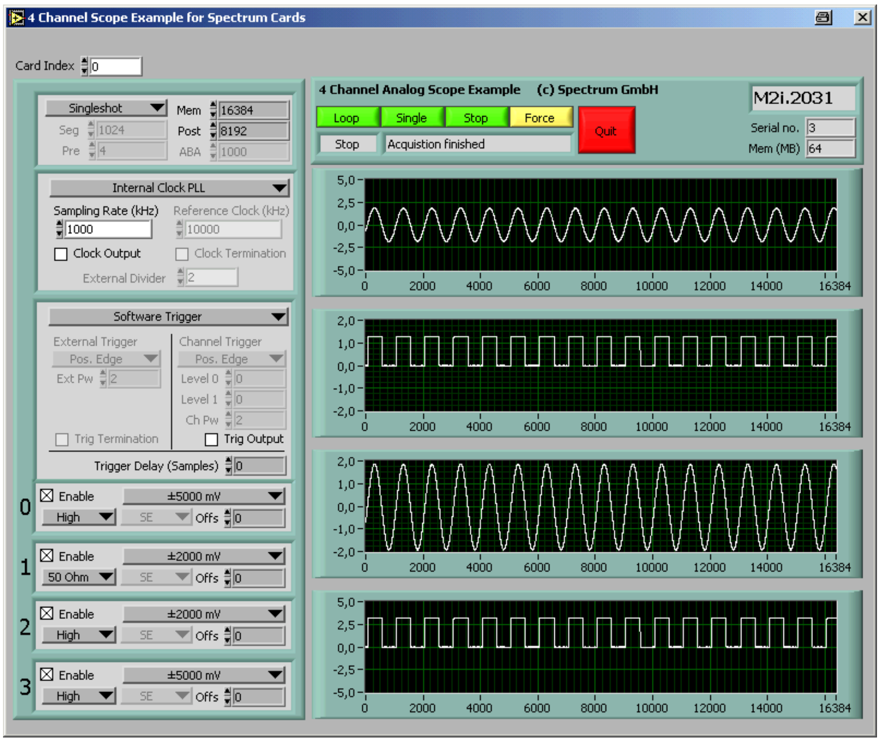
- Open the driver
- Read out the common card information
- Translates some of the information using one of the tools that are included in the delivery
- Display the translated content like card type, versions and production date
- Read out the card type specific information (in our example the card is an analog acquisition card) and shows the information on screen
- Checks for an error
- Closes the driver again

This example can be used as a base for own programs that do not fit under one of the other examples that are explained on the following pages.



4 Channel Analog Scope (scope_4channel.vi)

This example gives you the chance to use the analog acquisition card as a simple scope. It is possible to make single acquisitions as well as to run the card in a acquisition loop. All clock settings, single trigger sources and all input channel settings can be tested and changed in the interface. The example covers all acquisitions cards with up to 4 channels independent of the analog resolution or the maximum speed the card has. When using cards with more channels together with this example please keep in mind that only the first 4 channels can be used.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Setup

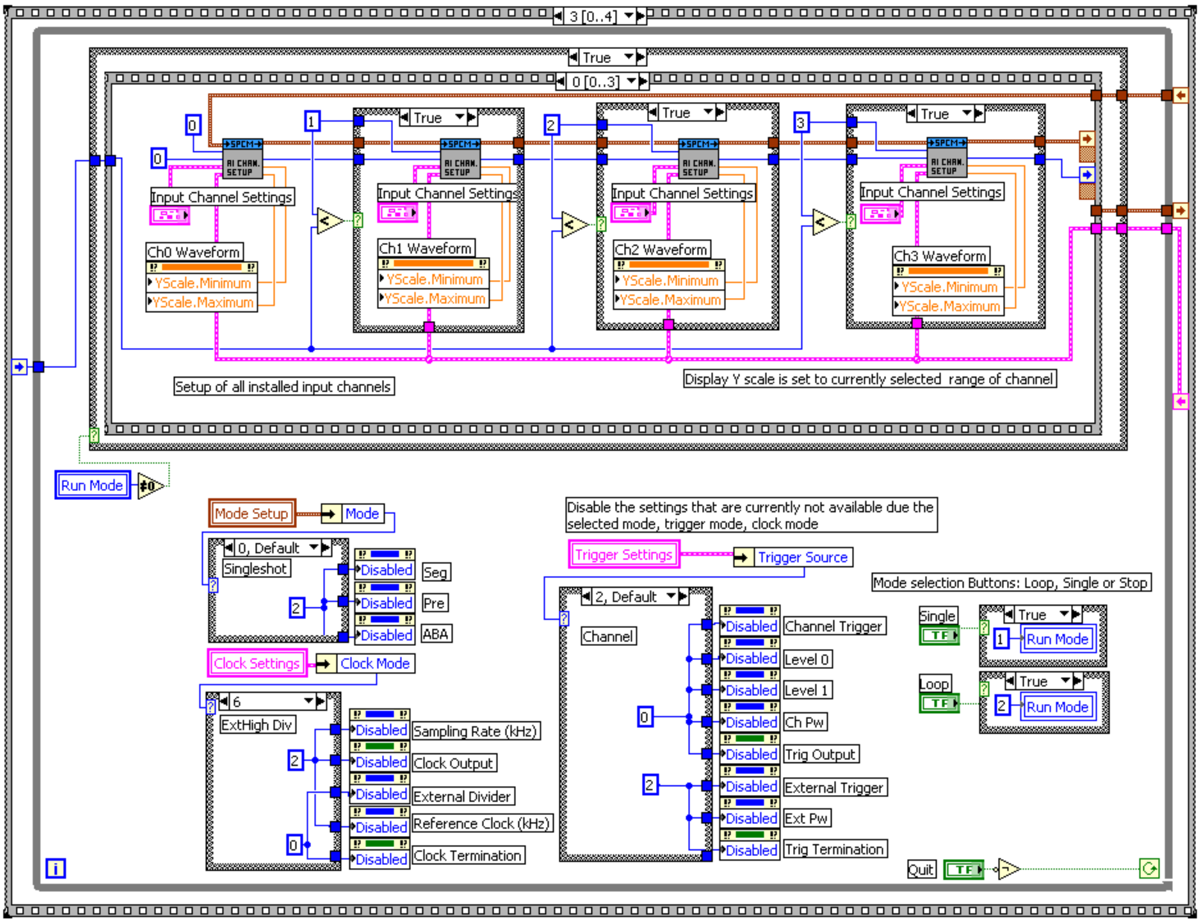
On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Remarks on the example

- The example is not optimized for display speed. Especially the data sorting and transforming of raw data to voltage levels takes some time.
- The example was done to show the use of the card as a scope. However the programmable memory is not limited in the example. Please keep in mind that one sample of data is converted to a 4 byte float value. It is not advised to use more memory than it is installed in the PC system as this may crash LabVIEW or slow down the system extremely.
- The example uses status polling to have complete control on the system and to keep the example simple to understand. However it is of course also possible to use one of the interrupt driver wait functions and change the example into a thread based program.

The example diagram

The following diagram shows the main loop of the example. All other sequence steps before this mainloop are only needed for initialization and for the initial setup of the user interface.



The bottom half handles the enabling/disabling of the different user interface components depending on the selected modes and also changed the run mode depending on the buttons that have been pressed.

The upper part contains the main acquisition sequence showing in the picture the first step of the acquisition. This sequence is called whenever the run mode is not in the stop state (not zero). The sequence contains the following steps:

Sequence 0 (shown above)

Setup of all installed input channels. The example handles up to 4 channels simultaneous. If less channels are installed the setup for the non-installed channels is not written to the driver. Second, this sequence step scales the waveform display graphs to match the currently selected input range and input offset. Therefore it uses the corresponding outputs of the „AI Chan. Setup" VI

Sequence 1

Does the rest of the setup. Mode, clock and trigger is written to the driver.

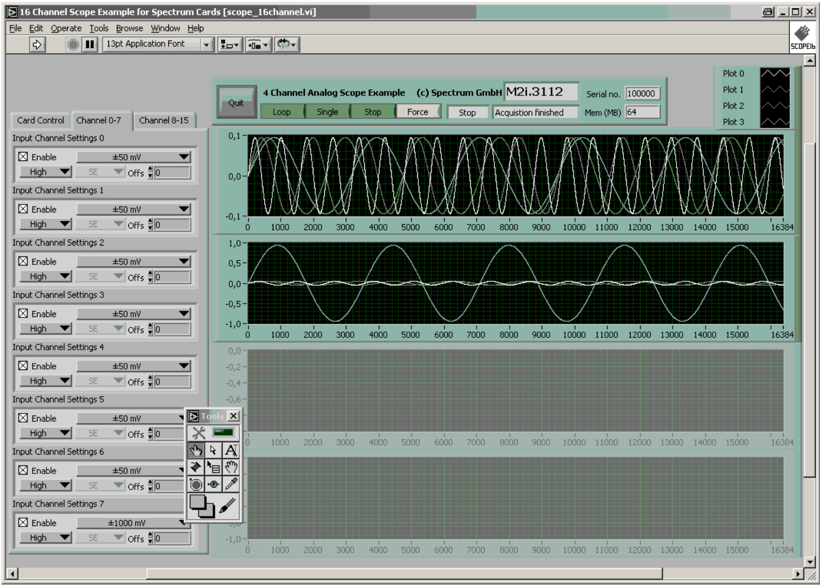
Sequence 2

Starts the card and waits inside a loop for the acquisition ready flag of the driver. The loop permanently checks for the current state and display the acquisition state in the status window. Further the loop checks whether force trigger or stop buttons are pressed and then executes the corresponding command. The loop is ended if the acquisition has been finished, an error has occurred or stop/quit button has been pressed.

Sequence 3

The last sequence step reads out the data if no error has occurred. It also scales the waveform graph to the currently selected memory size. If an error is found the error message is displayed and the loop is stopped.

16 Channel Analog Scope (scope_16channel.vi)



This example gives you the chance to use the analog acquisition card as a simple scope. It is possible to make single acquisitions as well as to run the card in a acquisition loop. All clock settings, single trigger sources and all input channel settings can be tested and changed in the interface. The example covers all acquisitions cards with up to 16 channels independent of the analog resolution or the maximum speed the card has.

The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

The card setup is splitted into three pages to group them for easier use. On the „Card Control" page all settings are placed that are related to the card like clock, memory or mode setup. The other two pages contain the complete analog setup for 8 channels each.

The channel data display is combined in 4 waveform diagrams each containing 4 analog channels.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Setup

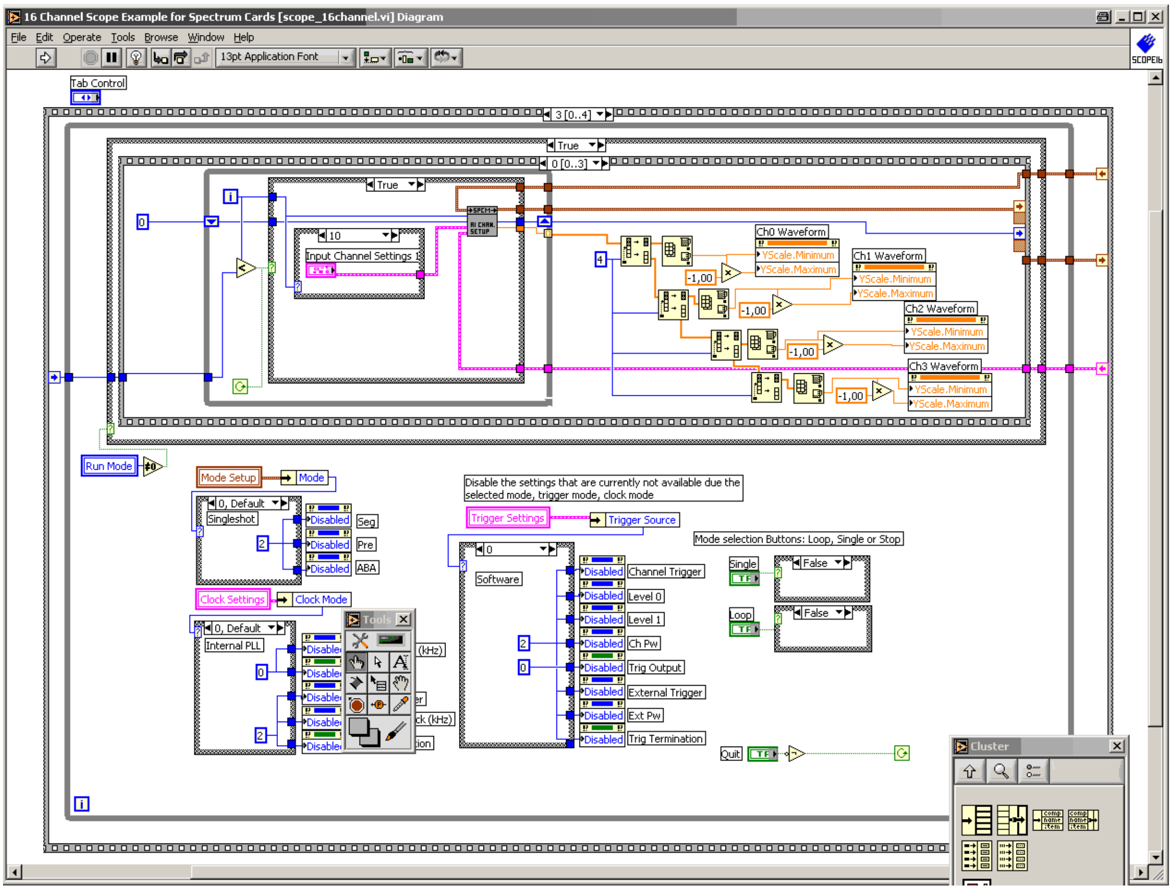
On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Remarks on the example

- The example is not optimized for display speed. Especially the data sorting and transforming of raw data to voltage levels takes some time.
- The example was done to show the use of the card as a scope. However the programmable memory is not limited in the example. Please keep in mind that one sample of data is converted to a 4 byte float value. It is not advised to use more memory than it is installed in the PC system as this may crash LabVIEW or slow down the system extremely.
- The example uses status polling to have complete control on the system and to keep the example simple to understand. However it is of course also possible to use one of the interrupt driver wait functions and change the example into a thread based program.

The example diagram

The following diagram shows the main loop of the example. All other sequence steps before this mainloop are only needed for initialization and for the initial setup of the user interface.



The bottom half handles the enabling/disabling of the different user interface components depending on the selected modes and also changed the run mode depending on the buttons that have been pressed.

The upper part contains the main acquisition sequence showing in the picture the first step of the acquisition. This sequence is called whenever the run mode is not in the stop state (not zero). The sequence contains the following steps:

Sequence 0 (shown above)

Setup of all installed input channels. The example handles up to 16 channels simultaneous. If less channels are installed the setup for the non-installed channels is not written to the driver. Second, this sequence step scales the waveform display graphs to match the currently selected input range and input offset. Therefore it uses the corresponding outputs of the „AI Chan. Setup" VI

Sequence 1

Does the rest of the setup. Mode, clock and trigger is written to the driver.

Sequence 2

Starts the card and waits inside a loop for the acquisition ready flag of the driver. The loop permanently checks for the current state and display the acquisition state in the status window. Further the loop checks whether force trigger or stop buttons are pressed and then executes the corresponding command. The loop is ended if the acquisition has been finished, an error has occurred or stop/quit button has been pressed.

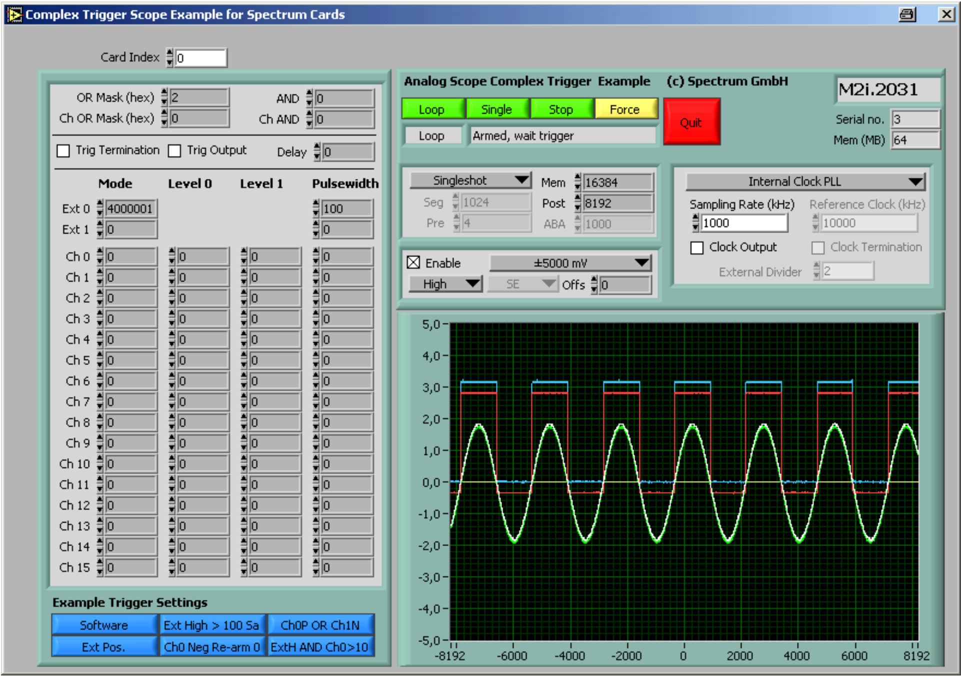
Sequence 3

The last sequence step reads out the data if no error has occurred. It also scales the waveform graph to the currently selected memory size. If an error is found the error message is displayed and the loop is stopped.

Complex Trigger Example (scope_complex_trigger.vi)

This VI shows you how to program the complex trigger engine under LabVIEW. It is a simple copy of the standard scope example having only one input channel setup that is used for all channels on the card.

The example support all Spectrum analog acquisition cards with up to 16 analog channels. All channels are displayed in one waveform chart with different colours.



User Interface

Trigger Setup

The example expects raw trigger settings in the complex trigger setup on the left. Please have a look at the description of the „setup complex AI trigger.vi" to have an overview on the possible settings.

Please be sure to use the correct settings for the modes and enable masks. Any incorrect setup will be reported as an error by the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	tarts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Example Trigger Settings

These buttons set the complex trigger window to some tested example settings. The intention of these buttons is to show some valid setups of different type to give you a fast way to try the example.

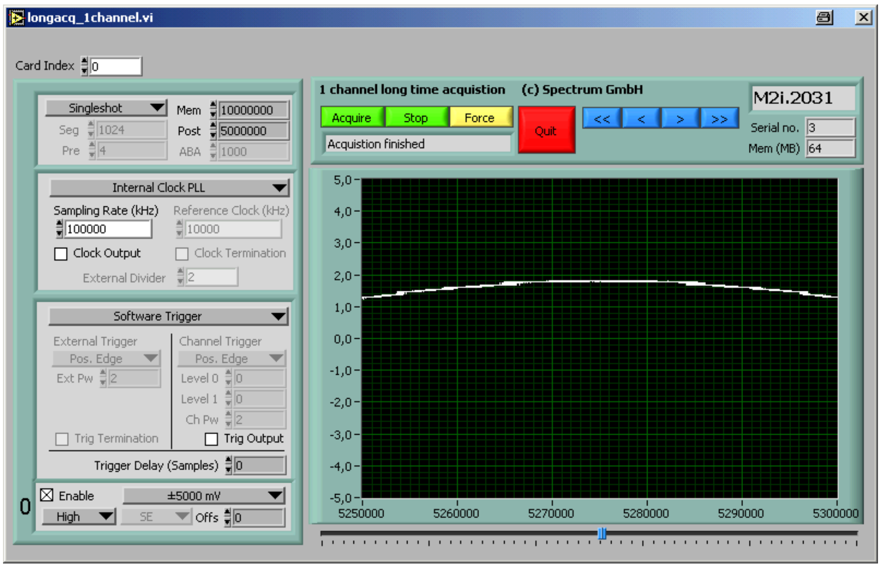
Software	the trigger engine is set to simple software trigger
Ext. Pos.	the trigger engine is set to one source external trigger input with positive edge.
Ext High > 100 Sa	the trigger engine is set to external high pulse > 100 Samples. This setup is shown in the screenshot above.
Ch0 Neg Re-arm 0	the trigger engine is set to one source channel 0 with re-arm mode and negative level 10 at zero voltage. The re-arm level is set to 0
Ch0P OR Ch1N	the trigger engine expects a positive edge on channel 0 at the level 10 or a negative edge at channel 1 at the level -10. This mode is only available if the card has at least two analog channels equipped
ExtH and Ch0 > 10	this shows an AND conjunction of external high level and level of channel 0 is above level 10

Pressing one of these buttons will stop the current loop as the setup is programmed new. Please restart the acquisition again by pressing the „Loop" or the „Single" button.

Long on-board memory acquisition (longacq_1channel.vi)

This example shows how to make very long shots up to the full on-board memory of 4 GByte and how to read them piece by piece afterwards and display a chunk of the memory on screen.

To keep the example simple to understand we focus on only one channel to be displayed.



User Interface

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Acquire	starts a single acquisition to the on-board memory with the currently selected setup
Stop	aborts the current acquisition
Force	forces a trigger if no trigger event is found by the hardware.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current acquisition state as it is read from the hardware
<, >	jumps one single data block to the left or to the right
<<, >>	jumps data blocks to the left or to the right. If holding these buttons pressed one can scroll through the signal continuously

Waveform Graph

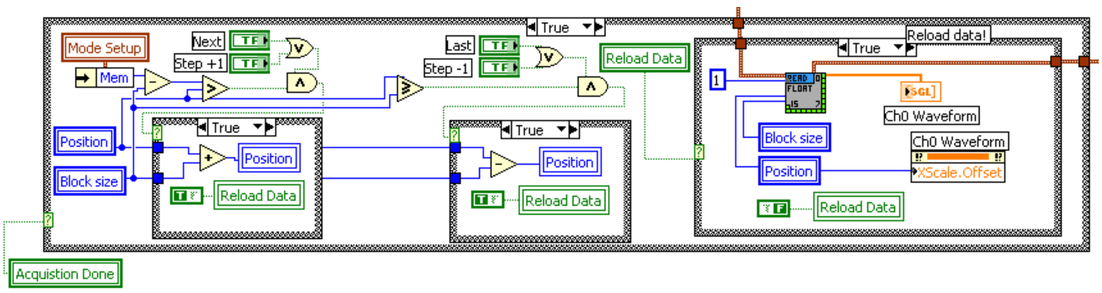
The waveform graph shows the current chunk inside the recorded data. Navigation can be done with the above mentioned navigation buttons. The slider just beneath the waveform graphs shows the current position of the waveform in relation to the complete acquired data. This position slider is used throughput the diagram below and is manipulated by the navigation buttons.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings

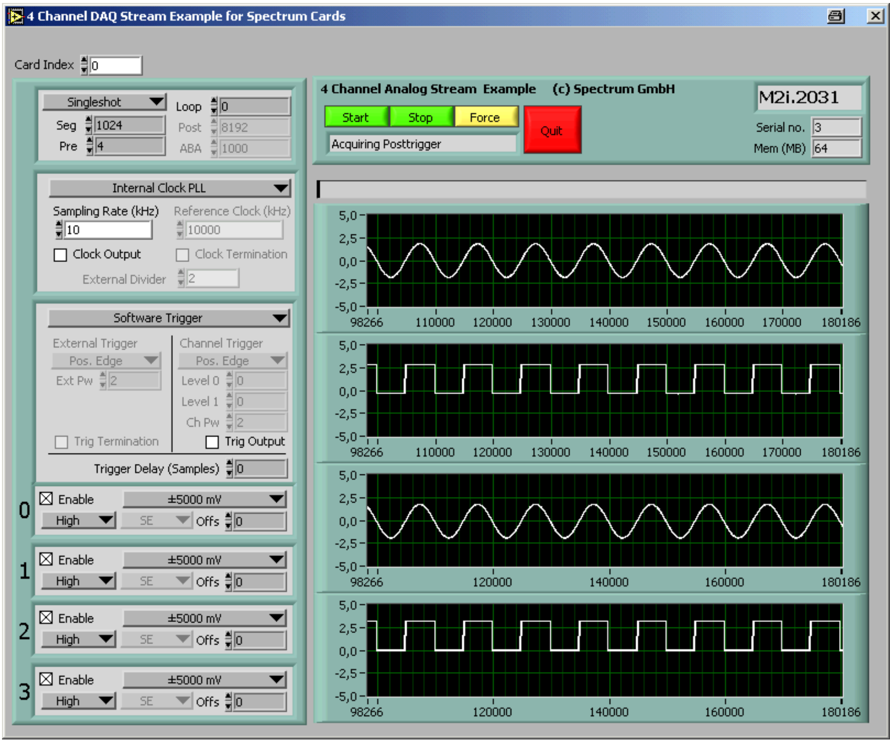
Example diagram

The diagram shows an except of the long acquisition example. As soon as the acquisition done flag has been set the part shown here allows the navigation inside the long acquisition. Pressing one of the navigation buttons will move the current position by the block size. Pressing one of the navigation buttons will also initiate a reload of data as shown on the right using the new position.



FIFO Acquisition Example (daq_stream_4channel.vi)

This example shows a continuous acquisition using the FIFO mode. The example does a complete setup for the card and starts with continuous data streaming afterwards until it is stopped or the programmed loop counting value is reached.



User Interface

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Buttons + Status

Start	starts the card in FIFO acquisition using the current setup on the left.
Stop	stops the current acquisition
Force	forces a trigger if no trigger event is found by the hardware
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Waveform Graph

The four waveform graphs show the data that is acquired continuously. Therefore the graphs are initialized with a size 5 times the currently programmed block size. This allows a more or less continuously display update.

Above these graphs there is a fill size display that shows the current fill size of the hardware FIFO buffer. If the system is fast enough this example should be able to run with a few MHz including display without getting an overrun.

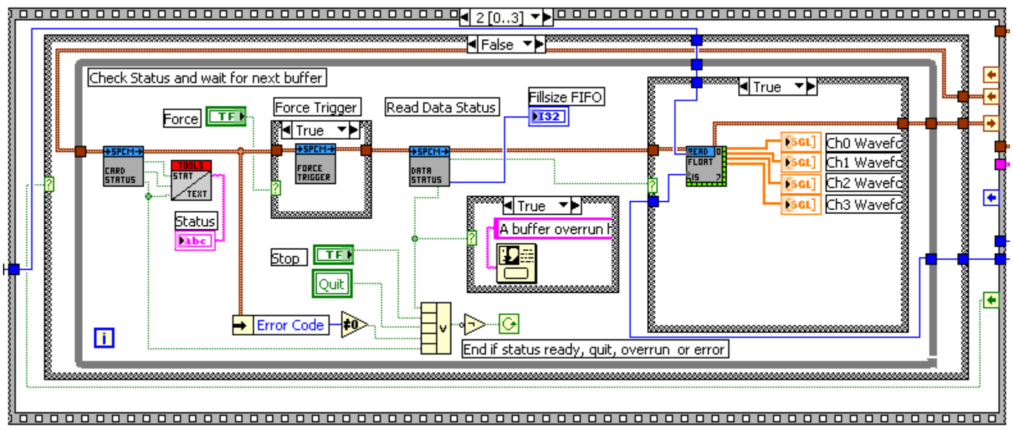
Remarks

The example was done to show how FIFO mode is running and to test the system using FIFO mode. It was never designed for maximum speed. To get maximum speed one is requested to use the dwDataRead_raw functions instead of the sorting functions found in this example. Also maximum speed can only be reached when not doing such complex displays as done here in this example.

Example diagram

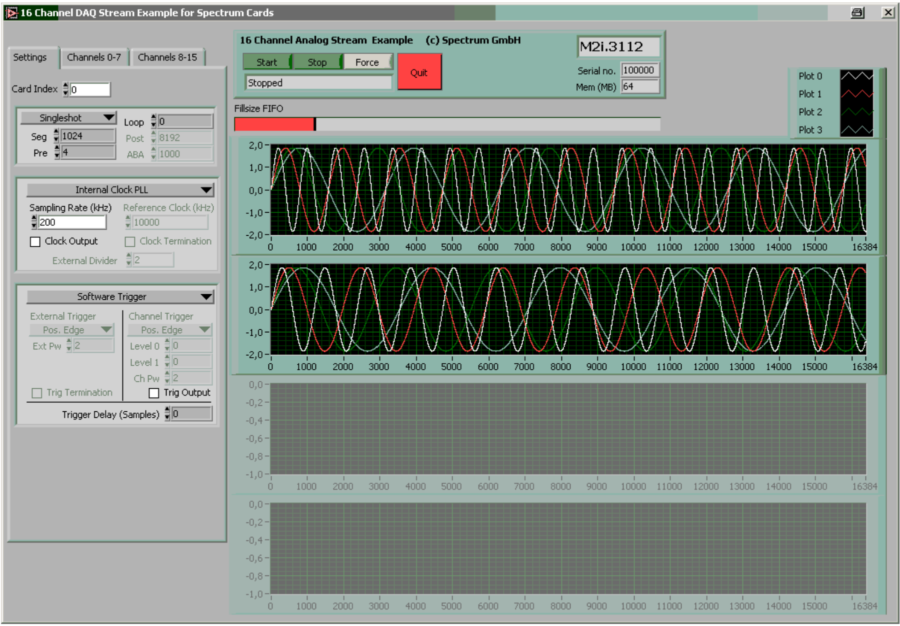
This diagram except shows the main FIFO loop. It checks for the current card status and displays a status message. 2nd it checks the current data status, shows the current fill size and checks for overrun.

If the data status shows that a new block of data is available the Read Float function is called and data is displayed in the 4 waveform charts



FIFO Acquisition Example (daq_stream_16channel.vi)

This example shows a continuous acquisition using the FIFO mode. The example does a complete setup for the card and starts with continuous data streaming afterwards until it is stopped or the programmed loop counting value is reached.



User Interface

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Buttons + Status

Start	starts the card in FIFO acquisition using the current setup on the left.
Stop	stops the current acquisition
Force	forces a trigger if no trigger event is found by the hardware
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Waveform Graph

The four waveform graphs show the data that is acquired continuously. Therefore the graphs are initialized with a size 5 times the currently programmed block size. This allows a more or less continuously display update.

Above these graphs there is a fill size display that shows the current fill size of the hardware FIFO buffer. If the system is fast enough this example should be able to run with a few MHz including display without getting an overrun.

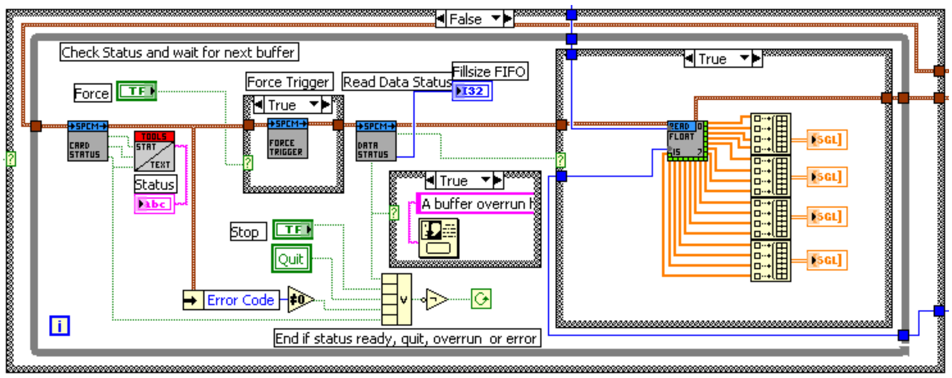
Remarks

The example was done to show how FIFO mode is running and to test the system using FIFO mode. It was never designed for maximum speed. To get maximum speed one is requested to use the dwDataRead_raw functions instead of the sorting functions found in this example. Also maximum speed can only be reached when not doing such complex displays as done here in this example.

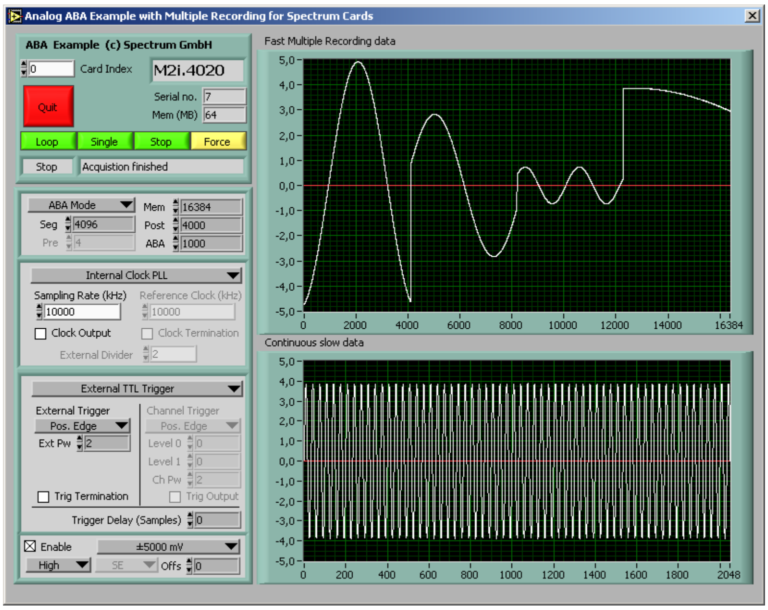
Example diagram

This diagram except shows the main FIFO loop. It checks for the current card status and displays a status message. 2nd it checks the current data status, shows the current fill size and checks for overrun.

If the data status shows that a new block of data is available the Read Float function is called and data is displayed in the 4 waveform charts (4 channels each)



DAQ ABA Example (daq_aba_multiple_rec.vi)



This example shows how to use the ABA feature together. To run this example the ABA feature has to be installed on the card.

All clock settings, single trigger sources and all input channel settings can be tested and changed in the interface. The example covers all analog acquisitions cards independent of the analog resolution or the maximum speed the card has. To keep the example simple there is only one analog channel displayed.

The example needs an external trigger signal to run. To see the slow („A") acquisition working it is advised to start the example without having the trigger source connected.

The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Display

The upper graphical display shows the fast („B") waveform that was acquired. As the example is using ABA mode the waveform will be separated in multiple waveform chunks each representing one segment of data.

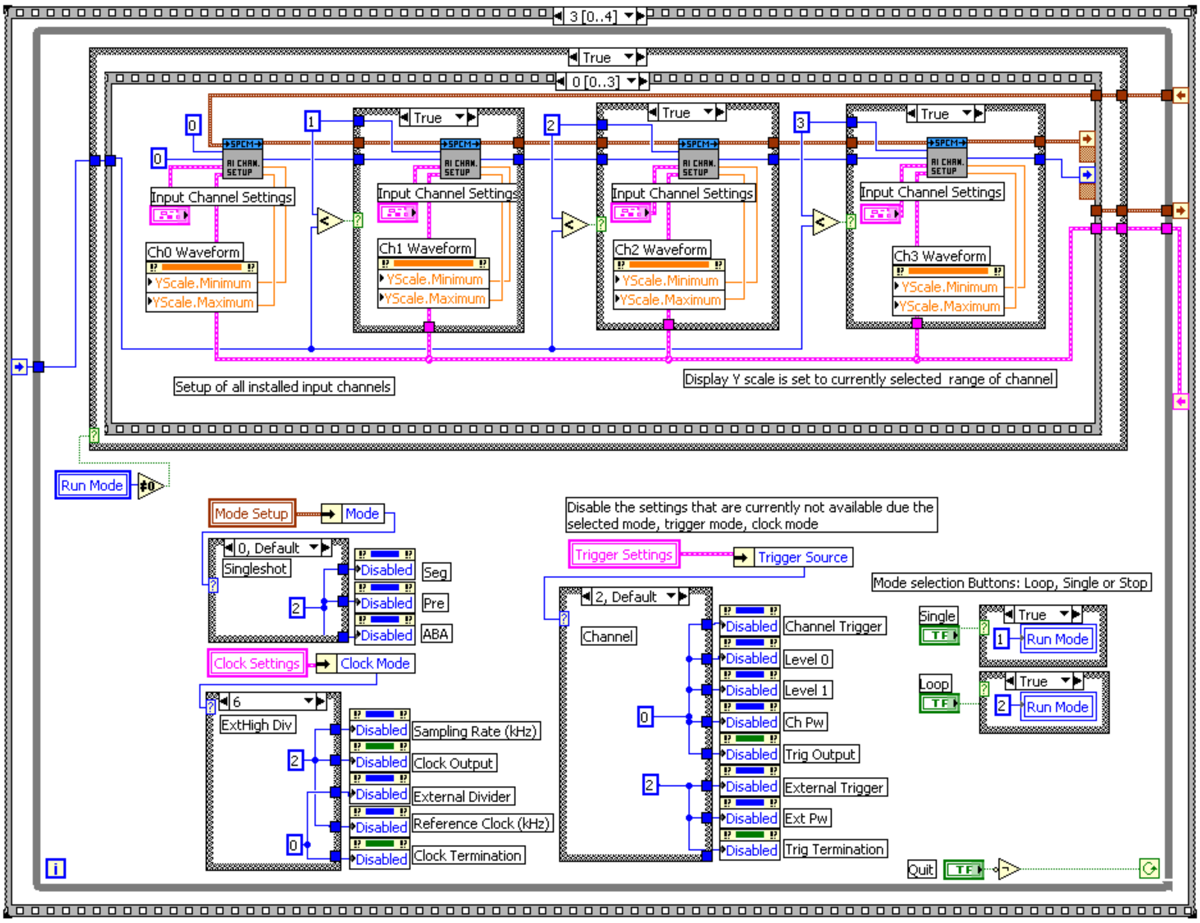
The lower graphical display continuously shows the slow („A") data. This display is continuously updated until the card has been stopped or the acquisition of the fast data has been finished.

Remarks on the example

- The example is not optimized for display speed. Especially the data sorting and transforming of raw data to voltage levels takes some time.
- The example was done to show the use of the card as a scope. However the programmable memory is not limited in the example. Please keep in mind that one sample of data is converted to a 4 byte float value. It is not advised to use more memory than it is installed in the PC system as this may crash LabVIEW or slow down the system extremely.
- The example uses status polling to have complete control on the system and to keep the example simple to understand. However it is of course also possible to use one of the interrupt driver wait functions and change the example into a thread based program.

The example diagram

The following diagram shows the main loop of the example. All other sequence steps before this mainloop are only needed for initialization and for the initial setup of the user interface.



The bottom half handles the enabling/disabling of the different user interface components depending on the selected modes and also changed the run mode depending on the buttons that have been pressed.

The upper part contains the main acquisition sequence showing in the picture the first step of the acquisition. This sequence is called whenever the run mode is not in the stop state (not zero). The sequence contains the following steps:

Sequence 0 (shown above)

Setup of all installed input channels. The example handles up to 4 channels simultaneous. If less channels are installed the setup for the non-installed channels is not written to the driver. Second, this sequence step scales the waveform display graphs to match the currently selected input range and input offset. Therefore it uses the corresponding outputs of the „AI Chan. Setup“ VI

Sequence 1

Does the rest of the setup. Mode, clock and trigger is written to the driver.

Sequence 2

Starts the card and waits inside a loop for the acquisition ready flag of the driver. The loop permanently checks for the current state and display the acquisition state in the status window. Further the loop checks whether force trigger or stop buttons are pressed and then executes the corresponding command. The loop is ended if the acquisition has been finished, an error has occurred or stop/quit button has been pressed.

Sequence 3

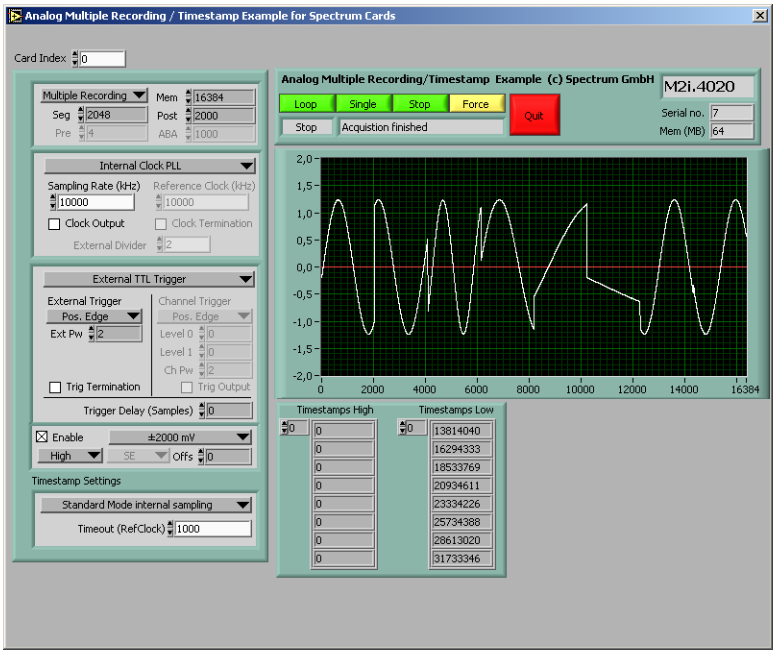
The last sequence step reads out the data if no error has occurred. It also scales the waveform graph to the currently selected memory size. If an error is found the error message is displayed and the loop is stopped.

DAQ Multiple Recording + Timestamp Example (daq_multi_timestamp.vi)

This example shows how to use the Multiple Recording feature together with the Timestamp feature. To run this example these both features have to be installed on the cards.

All clock settings, single trigger sources and all input channel settings can be tested and changed in the interface. The example covers all analog acquisitions cards independent of the analog resolution or the maximum speed the card has. To keep the example simple there is only one analog channel displayed.

The example needs an external trigger signal to run. Please change the trigger mode if you need an internal (channel) trigger.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Display

The graphical display shows the waveform that was acquired. As the example is using Multiple Recording mode the waveform will be separated in multiple waveform chunks each representing one segment of data.

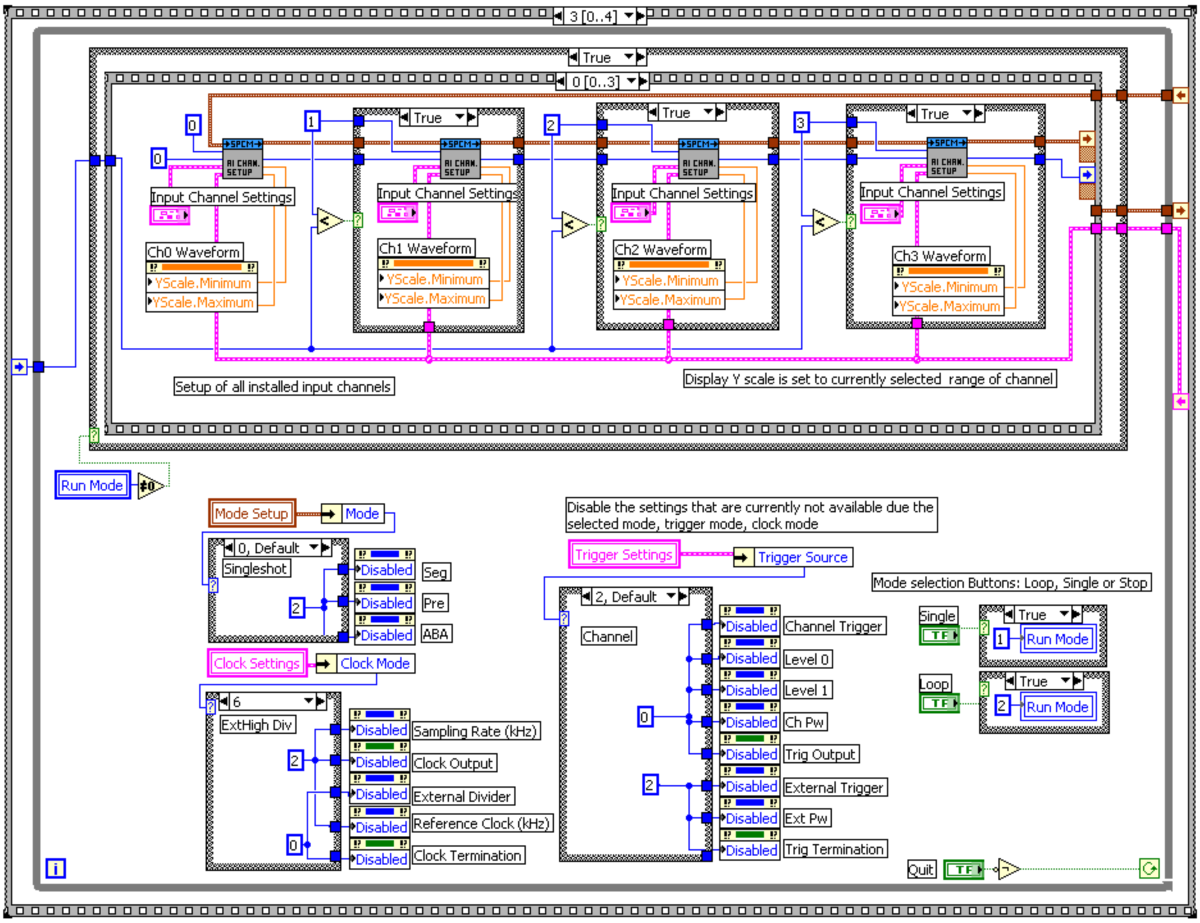
The list display below shows the timestamps that have been read and that correspond to the acquired segments. The number of timestamps is equal to the number of programmed segments. In the example there is enough space to show the timestamps of 8 acquired segments.

Remarks on the example

- The example is not optimized for display speed. Especially the data sorting and transforming of raw data to voltage levels takes some time.
- The example was done to show the use of the card as a scope. However the programmable memory is not limited in the example. Please keep in mind that one sample of data is converted to a 4 byte float value. It is not advised to use more memory than it is installed in the PC system as this may crash LabVIEW or slow down the system extremely.
- The example uses status polling to have complete control on the system and to keep the example simple to understand. However it is of course also possible to use one of the interrupt driver wait functions and change the example into a thread based program.

The example diagram

The following diagram shows the main loop of the example. All other sequence steps before this mainloop are only needed for initialization and for the initial setup of the user interface.



The bottom half handles the enabling/disabling of the different user interface components depending on the selected modes and also changed the run mode depending on the buttons that have been pressed.

The upper part contains the main acquisition sequence showing in the picture the first step of the acquisition. This sequence is called whenever the run mode is not in the stop state (not zero). The sequence contains the following steps:

Sequence 0 (shown above)

Setup of all installed input channels. The example handles up to 4 channels simultaneous. If less channels are installed the setup for the non-installed channels is not written to the driver. Second, this sequence step scales the waveform display graphs to match the currently selected input range and input offset. Therefore it uses the corresponding outputs of the „AI Chan. Setup“ VI

Sequence 1

Does the rest of the setup. Mode, clock and trigger is written to the driver.

Sequence 2

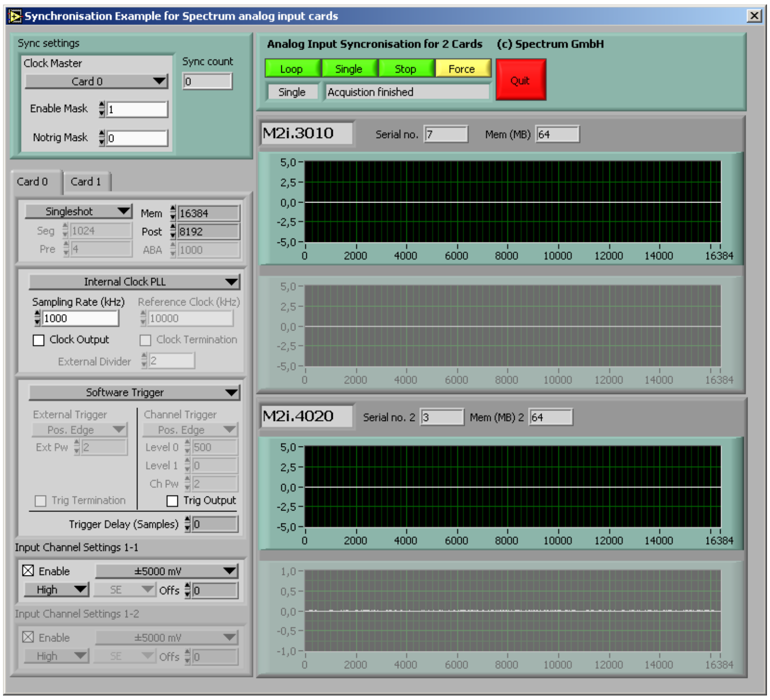
Starts the card and waits inside a loop for the acquisition ready flag of the driver. The loop permanently checks for the current state and display the acquisition state in the status window. Further the loop checks whether force trigger or stop buttons are pressed and then executes the corresponding command. The loop is ended if the acquisition has been finished, an error has occurred or stop/quit button has been pressed.

Sequence 3

The last sequence step reads out the data if no error has occurred. It also scales the waveform graph to the currently selected memory size. If an error is found the error message is displayed and the loop is stopped.

Analog Input Synchronization example (sync_analog_in.vi)

This example shows how to use the synchronisation option with two analog acquisition cards. To keep the example simple it only displays two channels of the cards. You are requested to modify this example to merge it with number of cards and channels required. The example is just done to show how the synchronisation works and how to program it.



The User Interface

The user interface allows the complete setup of both cards. Each card has its own setup page allowing mode, clock and trigger setup. The example shows the first two channels of two synchronized cards.

Buttons + Status

Loop — starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.

Single — makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.

Stop — stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.

Force — forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.

Quit — quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.

Status display — shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

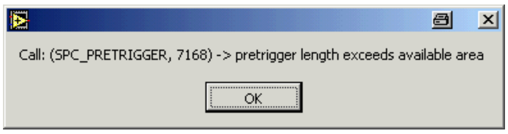
Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Analog Output (AO) examples

This chapter gives you a brief overview of the examples that come together with the driver. Please keep in mind that these are only examples to show how the driver can be programmed. Although most of these examples can also be used as complete and comfortable stand-alone programs that wasn't our intention. Therefore there might be some limits in the examples and some settings are not checked on LabVIEW example level but only on the level of the standard driver.

Encountering an error message as shown on the right is not a bug of the LabVIEW driver or the example but it is simply a setup that isn't valid. Please check all details of the hardware manual to see what was going wrong here. In our example one tries to use Multiple Recording and exceeds the available pretrigger length. The register name (SPC_PRETRIGGER) gives you a clue where to search inside the hardware manual.



The examples are delivered „as is" and they're not intended to become more powerful applications as this makes the understanding of the examples very difficult. If you encounter any general problems with the examples please contact our support.

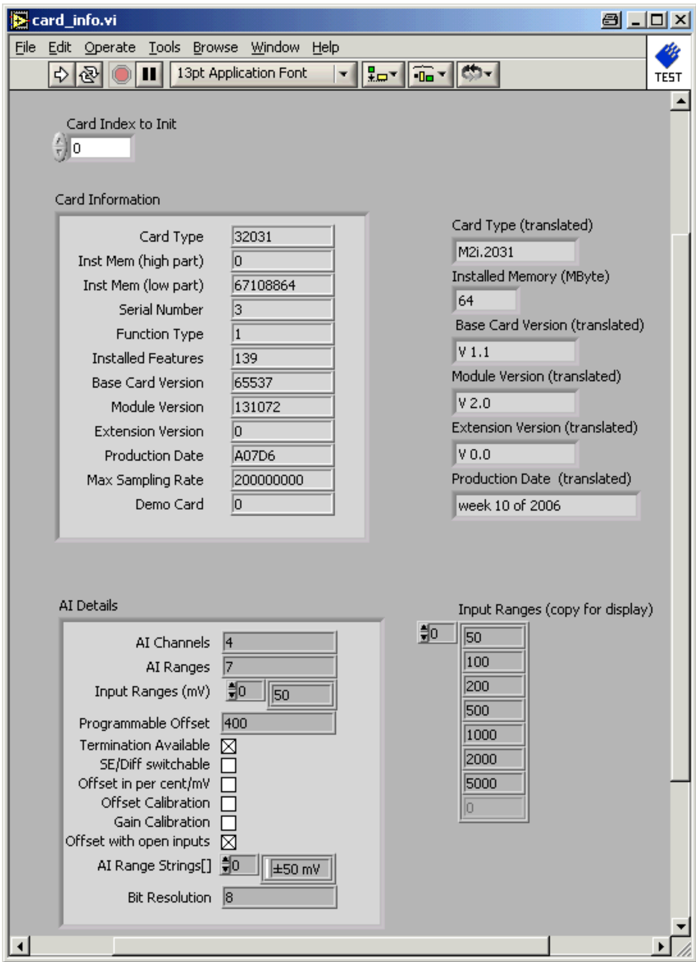
Please feel free to use the examples and to modify them for your own applications. Please keep in mind that we do not support any modified examples!

Card Information (card_info.vi)

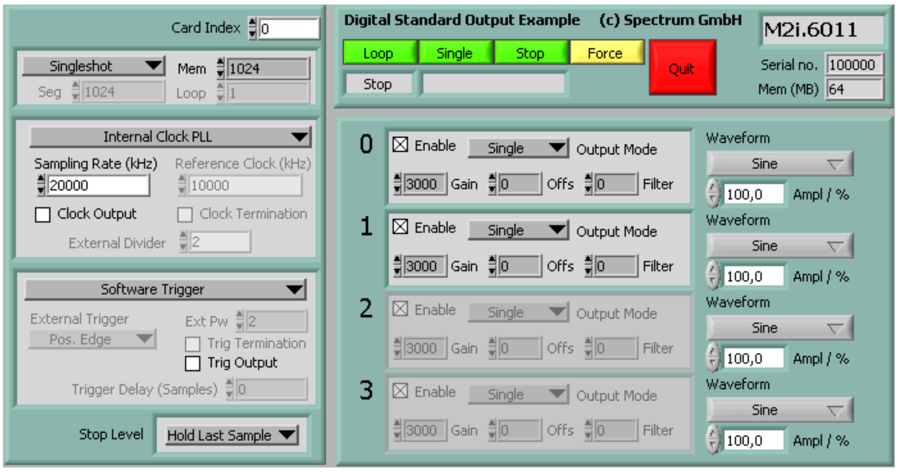
This VI shows how to initialize the card and how to read out further information on the card. The VI does the following steps:

- Open the driver
- Read out the common card information
- Translates some of the information using one of the tools that are included in the delivery
- Display the translated content like card type, versions and production date
- Read out the card type specific information (in our example the card is an analog acquisition card) and shows the information on screen
- Checks for an error
- Closes the driver again

This example can be used as a base for own programs that do not fit under one of the other examples that are explained on the following pages.



Analog Standard Output Example (analog_standard_out.vi)



This example shows how to use the output functionality of the analog card in standard mode. To keep the example simple there are only a very few different waveforms to program. You are requested to modify this example and to merge it with the data source that you need for your application. The example is just done to show how the output works and how to program it.

The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Start	starts the output with the current setup. Depending on the selected loop mode the card will stop automatically after some time of output or it will run until the user stops it by pressing the „stop" button.
Stop	stops the current output and changes to the stop mode. A new output can be started by using the „start" button again.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event. When using Multiple Replay this means that one segment is replayed after each click of the force trigger button.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current output state. The state is read from hardware.

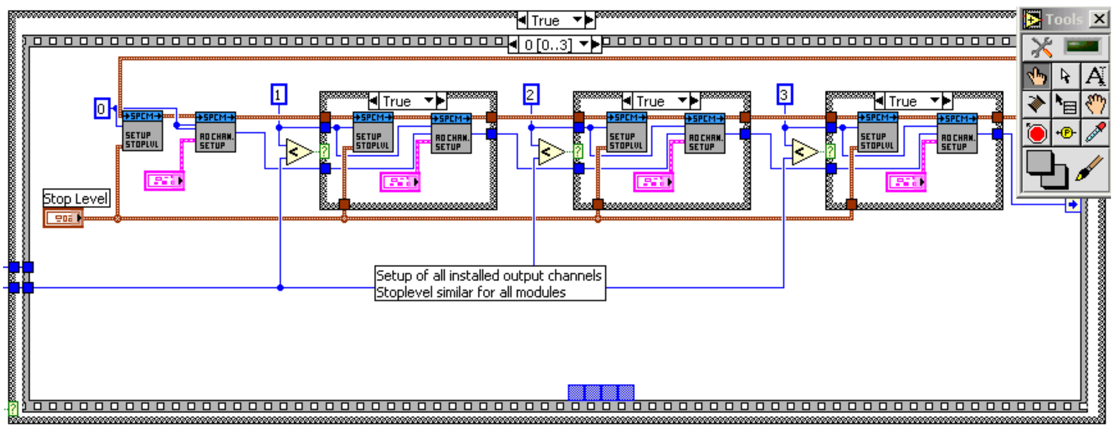
Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Waveform Setup

On the right hand side is a very small waveform setup for each channel. Each channel waveform can be defined in shape and amplitude in relation to the programmed output amplitude.

Diagram

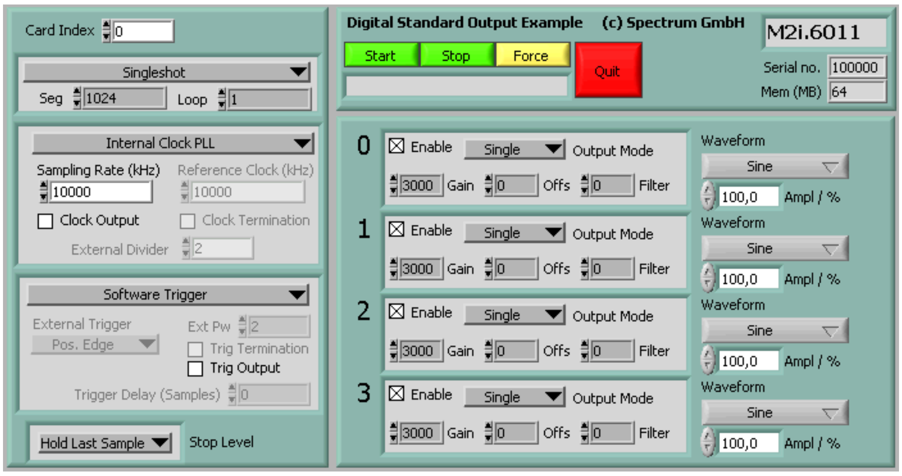


All setup is done in a chain forwading and error-checking a cluster with settings.

To allow the use of the example with all different types of arbitrary waveform generators, some parts are seperated by case structs and checked for the number of installed channels or the presence of separate features. This si shown in the diagram except fr the output channel setup allowing the support of a maximum of 4 synchronous output channels.

Analog FIFO Output Example (analog_stream_output.vi)

This example shows how to use the output functionality of the analog card in FIFO mode (continuous streaming). To keep the example simple there are only a very few different waveforms to program. You are requested to modify this example and to merge it with the data source that you need for your application. The example is just done to show how the output works and how to program it.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Start	starts the FIFO output with the current setup. Depending on the selected segment and loop size the card will stop automatically after replay of teh programmed ammount of data it will run until the user stops it by pressing the „stop" button.
Stop	stops the current output and changes to the stop mode. A new output can be started by using the „start" button again.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event. When using Multiple Replay this means that one segment is replayed after each click of the force trigger button.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current output state. The state is read from hardware.

Setup

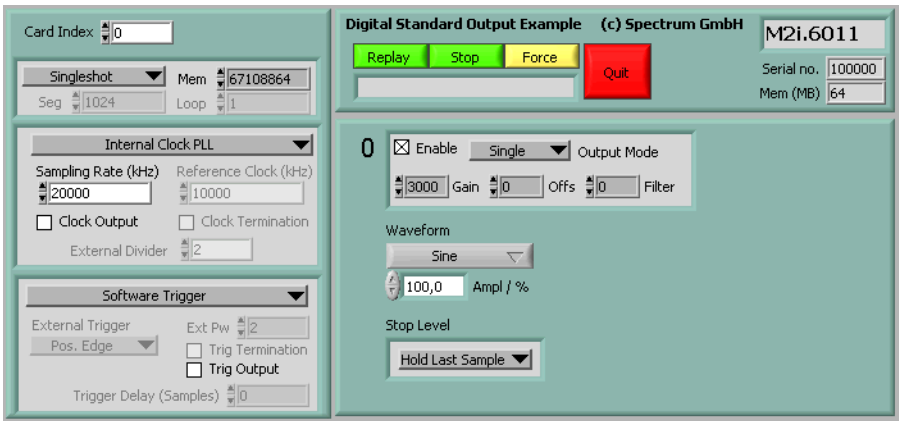
On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Waveform Setup

On the right hand side is a very small waveform setup for each channel. Each channel waveform can be defined in shape and amplitude in relation to the programmed output amplitude.

Analog long replay example (long_rep_analog_out.vi)

This example shows how to use the output functionality of the analog card with large memory. As the cards can have up to 4 GByte on-board memory one needs to generate/load and transfer this huge amount of data block by block. The example shows how this can be done by generating a simple waveform for the programmed memory. You are requested to modify this example and to merge it with the data source that you need for your application. The example is just done to show how the output works and how to program it.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

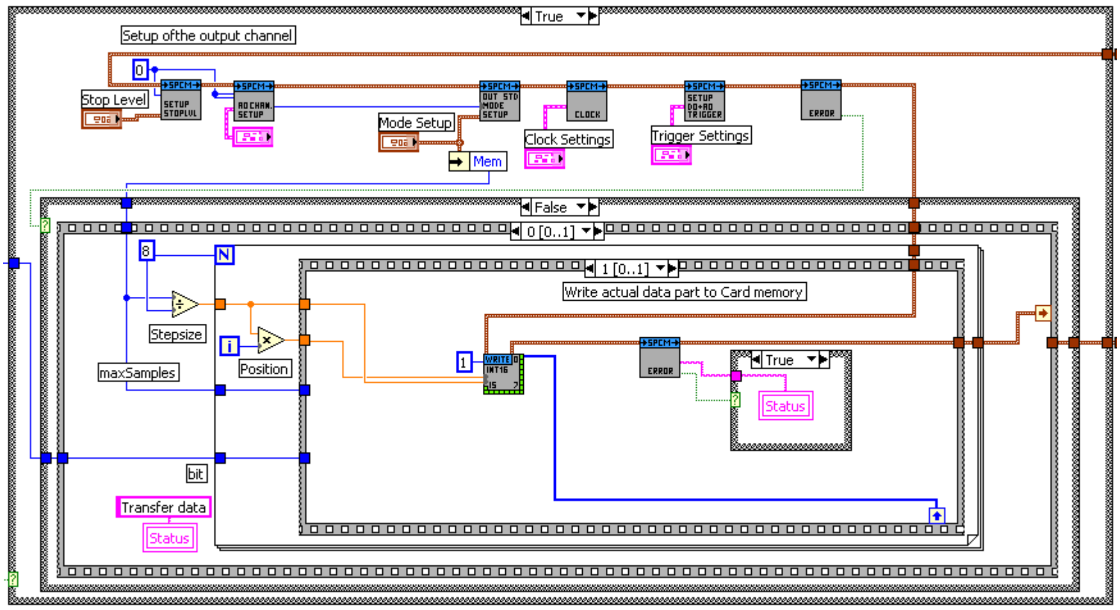
Start	starts the output with the current setup. Depending on the selected loop mode the card will stop automatically after some time of output or it will run until the user stops it by pressing the „stop" button.
Stop	stops the current output and changes to the stop mode. A new output can be started by using the „start" button again.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event. When using Multiple Replay this means that one segment is replayed after each click of the force trigger button.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current output state. The state is read from hardware.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Diagram

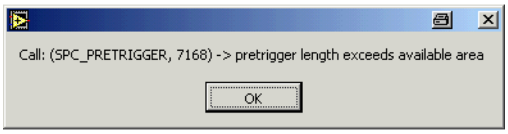
Data is programmed in 8 portions of equal size to the card. When using larger memory this stepsize should be increased to kepp the amount of used system memory low.



Digital Input (DI) examples

This chapter gives you a brief overview of the examples that come together with the driver. Please keep in mind that these are only examples to show how the driver can be programmed. Although most of these examples can also be used as complete and comfortable stand-alone programs that wasn't our intention. Therefore there might be some limits in the examples and some settings are not checked on LabVIEW example level but only on the level of the standard driver.

Encountering an error message as shown on the right is not a bug of the LabVIEW driver or the example but it is simply a setup that isn't valid. Please check all details of the hardware manual to see what was going wrong here. In our example one tries to use Multiple Recording and exceeds the available pretrigger length. The register name (SPC_PRETRIGGER) gives you a clue where to search inside the hardware manual.



The examples are delivered „as is" and they're not intended to become more powerful applications as this makes the understanding of the examples very difficult. If you encounter any general problems with the examples please contact our support.

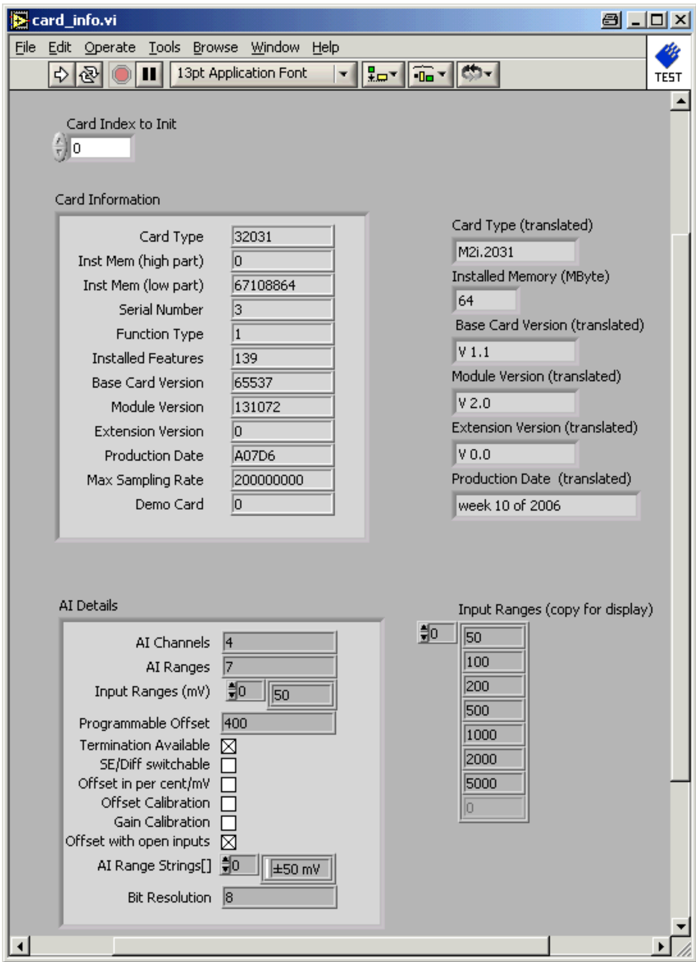
Please feel free to use the examples and to modify them for your own applications. Please keep in mind that we do not support any modified examples!

Card Information (card_info.vi)

This VI shows how to initialize the card and how to read out further information on the card. The VI does the following steps:

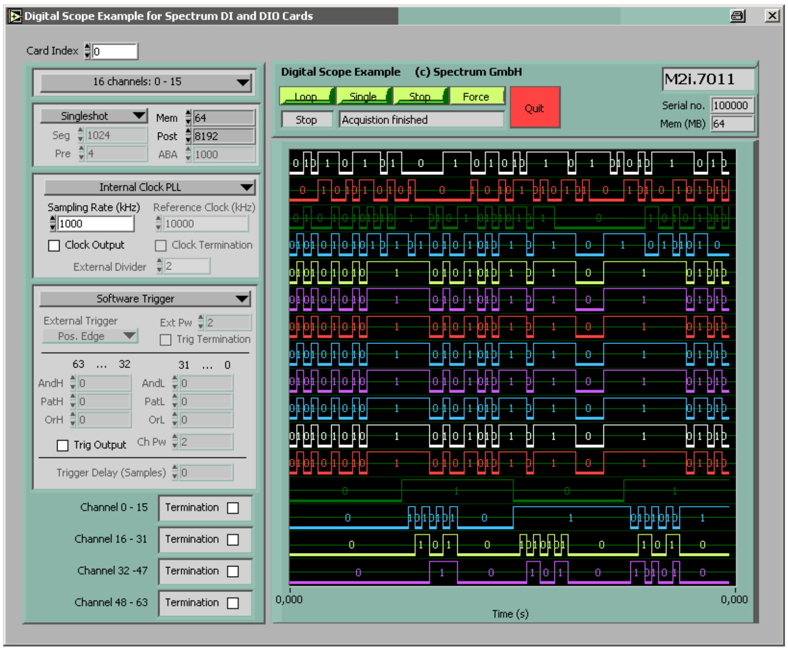
- Open the driver
- Read out the common card information
- Translates some of the information using one of the tools that are included in the delivery
- Display the translated content like card type, versions and production date
- Read out the card type specific information (in our example the card is an analog acquisition card) and shows the information on screen
- Checks for an error
- Closes the driver again

This example can be used as a base for own programs that do not fit under one of the other examples that are explained on the following pages.



Digital Scope/Logic Analyzer (digital_scope.vi)

This example gives you the chance to use the digital acquisition card as a simple digital scope / Logic Analyzer. It is possible to make single acquisitions as well as to run the card in a acquisition loop. All clock settings, trigger sources and all input channel settings can be tested and changed in the interface. The example covers all digital acquisitions cards with up to 64 digital channels independent of the maximum speed the card has.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Loop	starts the card in an acquisition loop as known from stand-alone scopes. The setup can be changed while the acquisition loop is running, the changed setup will be used on the next run.
Single	makes a single acquisition with the current setup. The card will stop after this single acquisition and can be restarted by another click on the single button or can be switched to loop mode by clicking the loop button.
Stop	stops the current acquisition and changes to the stop mode. A new acquisition can be started by using the single or loop button.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event resulting in an acquisition with no fixed phase relation to any outside signals.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Setup

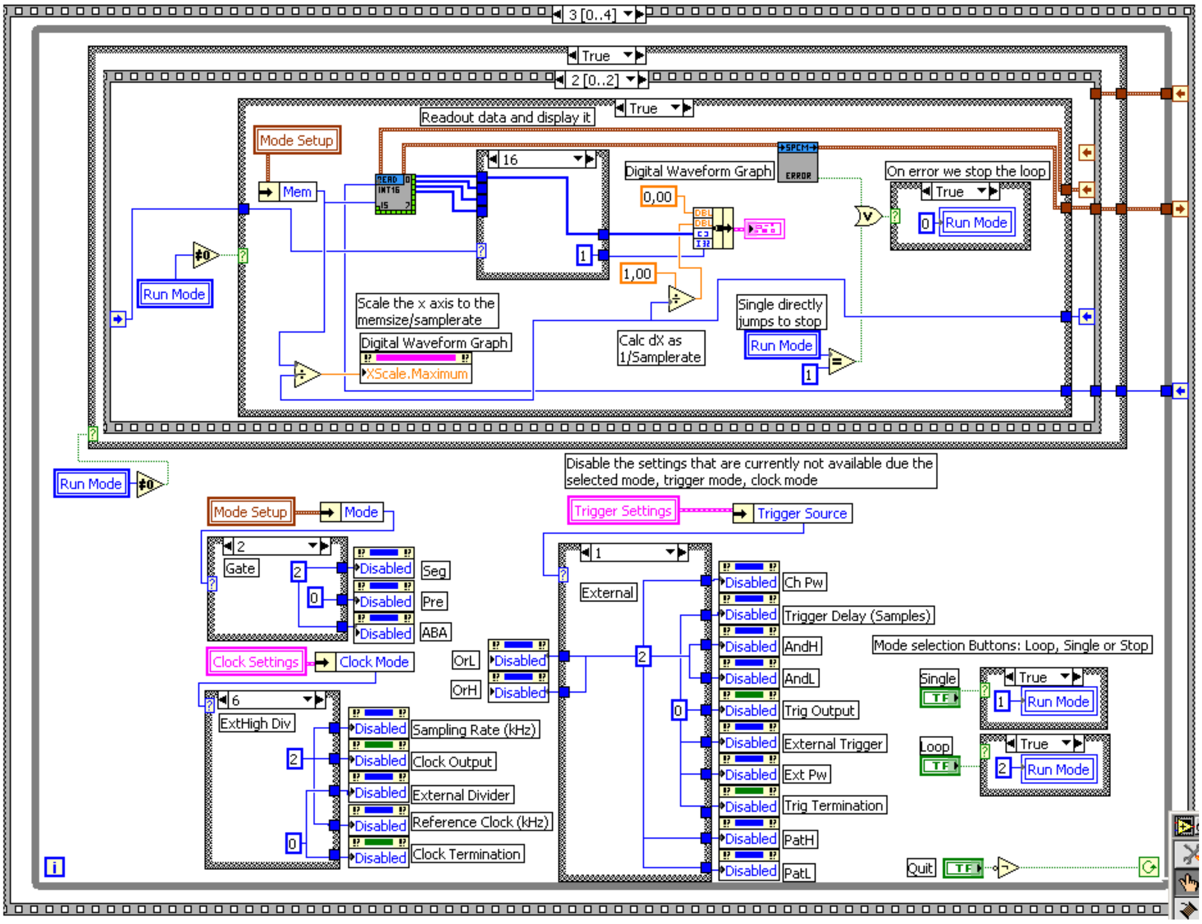
On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Remarks on the example

- The example is not optimized for display speed. Especially the data sorting and display as digital data may take some time.
- The example was done to show the use of the card as a scope. However the programmable memory is not limited in the example. Sorting and displaying huge amount of data may be a problem for the system. It is not advised to use more memory than it is installed in the PC system as this may crash LabVIEW or slow down the system extremely.
- The example uses status polling to have complete control on the system and to keep the example simple to understand. However it is of course also possible to use one of the interrupt driver wait functions and change the example into a thread based program.

The example diagram

The following diagram shows the main loop of the example. All other sequence steps before this mainloop are only needed for initialization and for the initial setup of the user interface.



The bottom half handles the enabling/disabling of the different user interface components depending on the selected modes and also changed the run mode depending on the buttons that have been pressed.

The upper part contains the main acquisition sequence showing in the picture the first step of the acquisition. This sequence is called whenever the run mode is not in the stop state (not zero). The sequence contains the following steps:

Sequence 0

Setup of all installed input channels and mode, clock, trigger. The example handles up to 64 channels simultaneous. If less channels are installed the setup for the non-installed channels is not written to the driver. Second, this sequence step scales the waveform display graphs to match the currently selected input range and input offset.

Sequence 1

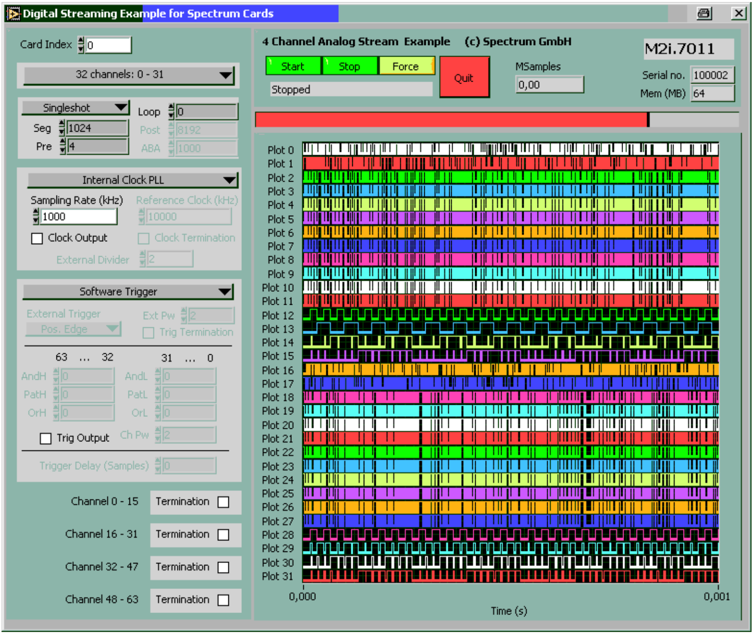
Starts the card and waits inside a loop for the acquisition ready flag of the driver. The loop permanently checks for the current state and display the acquisition state in the status window. Further the loop checks whether force trigger or stop buttons are pressed and then executes the corresponding command. The loop is ended if the acquisition has been finished, an error has occurred or stop/quit button has been pressed.

Sequence 2 (shown above)

The last sequence step reads out the data if no error has occurred. It also scales the waveform graph to the currently selected memory size. If an error is found the error message is displayed and the loop is stopped.

Digital FIFO Acquisition Example (digital_stream.vi)

This example shows a continuous acquisition using the FIFO mode. The example does a complete setup for the card and starts with continuous data streaming afterwards until it is stopped or the programmed loop counting value is reached.



User Interface

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Buttons + Status

Start	starts the card in FIFO acquisition using the current setup on the left.
Stop	stops the current acquisition
Force	forces a trigger if no trigger event is found by the hardware
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Waveform Graph

The digital waveform graph shows the data that is acquired continuously. The graph shows the last acquired block of data and has a more or less continuously display update.

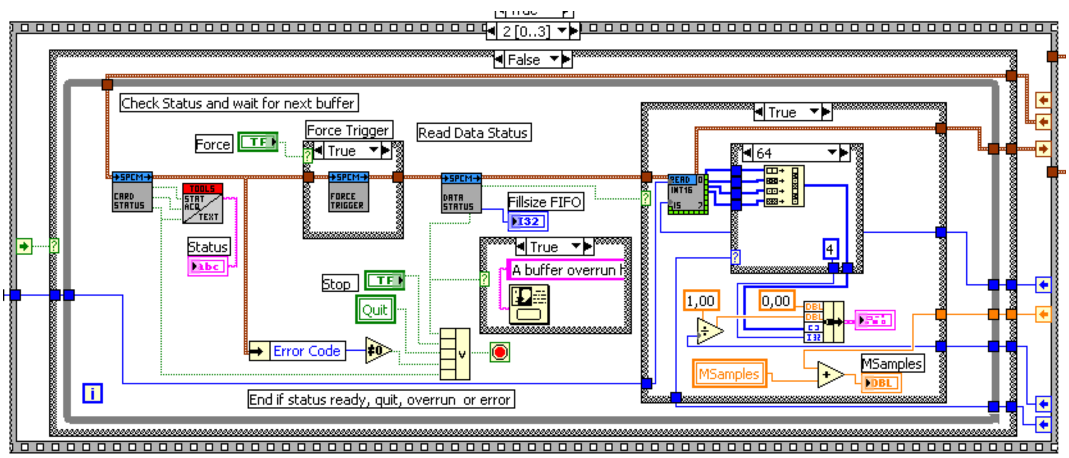
Above these graphs there is a fill size display that shows the current fill size of the hardware FIFO buffer. If the system is fast enough this example should be able to run with a few MHz including display without getting an overrun.

Remarks

The example was done to show how FIFO mode is running and to test the system using FIFO mode. It was never designed for maximum speed. To get maximum speed one is requested to use the dwDataRead_raw functions instead of the sorting functions found in this example. Also maximum speed can only be reached when not doing such complex displays as done here in this example.

Example diagram

This diagram except shows the main FIFO loop. It checks for the current card status and displays a status message. 2nd it checks the current data status, shows the current fill size and checks for overrun.

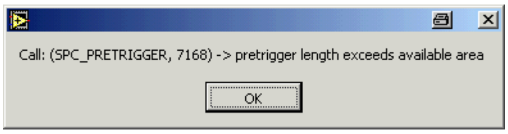


If the data status shows that a new block of data is available the Read int16 function is called and data is displayed in the digital waveform chart.

Digital Output (DO) examples

This chapter gives you a brief overview of the examples that come together with the driver. Please keep in mind that these are only examples to show how the driver can be programmed. Although most of these examples can also be used as complete and comfortable stand-alone programs that wasn't our intention. Therefore there might be some limits in the examples and some settings are not checked on LabVIEW example level but only on the level of the standard driver.

Encountering an error message as shown on the right is not a bug of the LabVIEW driver or the example but it is simply a setup that isn't valid. Please check all details of the hardware manual to see what was going wrong here. In our example one tries to use Multiple Recording and exceeds the available pretrigger length. The register name (SPC_PRETRIGGER) gives you a clue where to search inside the hardware manual.



The examples are delivered „as is" and they're not intended to become more powerful applications as this makes the understanding of the examples very difficult. If you encounter any general problems with the examples please contact our support.

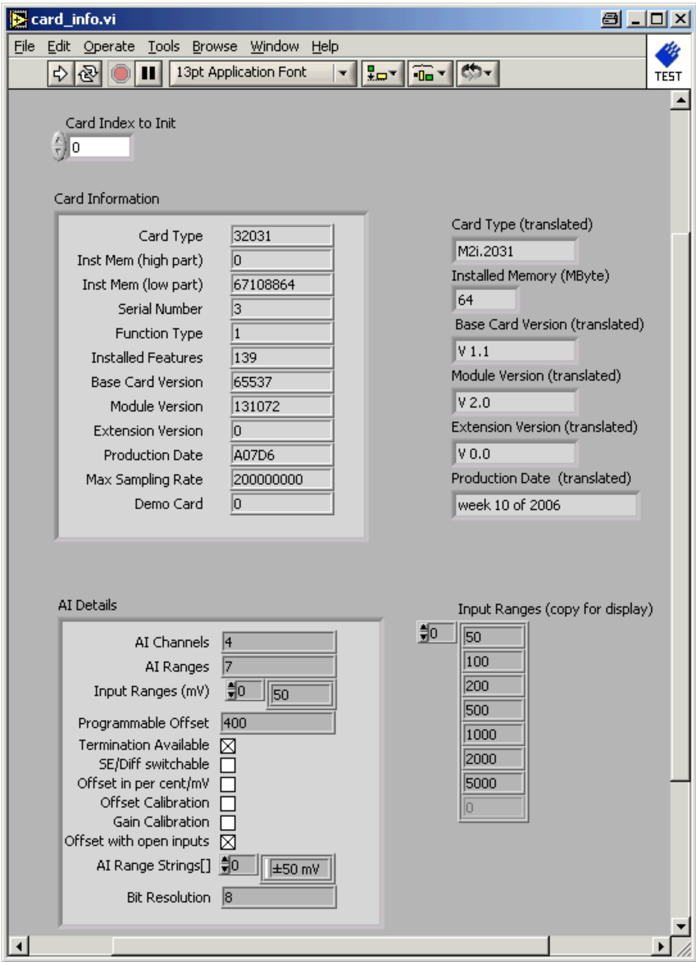
Please feel free to use the examples and to modify them for your own applications. Please keep in mind that we do not support any modified examples!

Card Information (card_info.vi)

This VI shows how to initialize the card and how to read out further information on the card. The VI does the following steps:

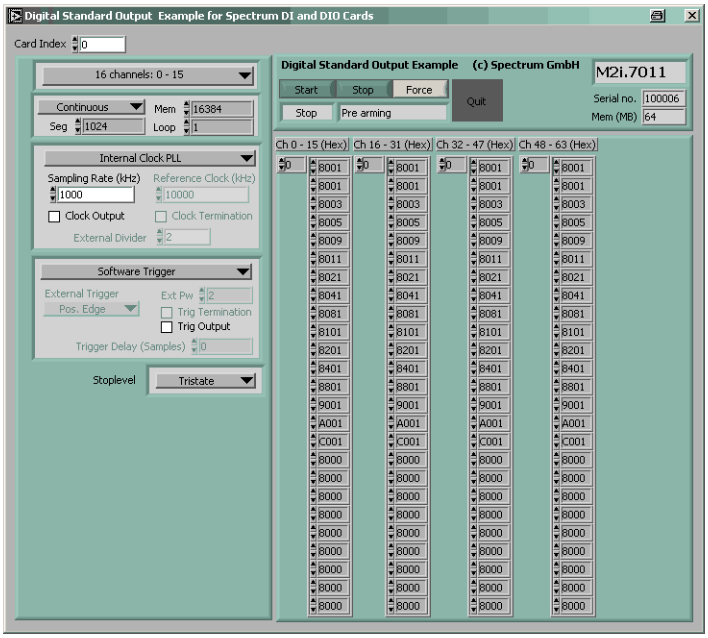
- Open the driver
- Read out the common card information
- Translates some of the information using one of the tools that are included in the delivery
- Display the translated content like card type, versions and production date
- Read out the card type specific information (in our example the card is an analog acquisition card) and shows the information on screen
- Checks for an error
- Closes the driver again

This example can be used as a base for own programs that do not fit under one of the other examples that are explained on the following pages.



Digital Standard Output Example (digital_standard_out.vi)

This example shows how to use the output functionality of the digital card in standard mode. To keep the example simple data is just generated from static arrays. You are requested to modify this example and to merge it with the data source that you need for your application. The example is just done to show how the output works and how to program it.



The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

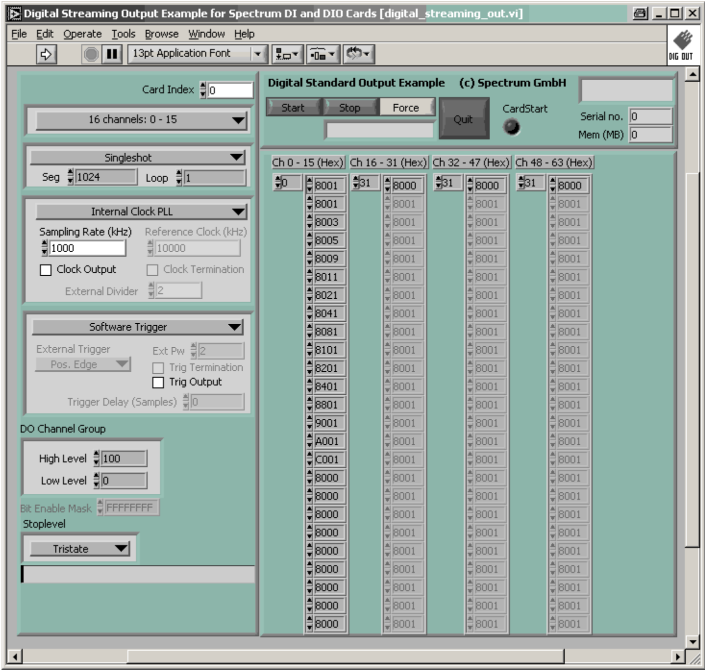
Start	starts the output with the current set-up. Depending on the selected loop mode the card will stop automatically after some time of output or it will run until the user stops it by pressing the „stop" button.
Stop	stops the current output and changes to the stop mode. A new output can be started by using the „start" button again.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event. When using Multiple Replay this means that one segment is replayed after each click of the force trigger button.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current output state. The state is read from hardware.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

Digital FIFO Output Example (digital_streaming_out.vi)

This example shows a continuous output using the FIFO mode. To keep the example simple data is taken from a static array.



User Interface

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

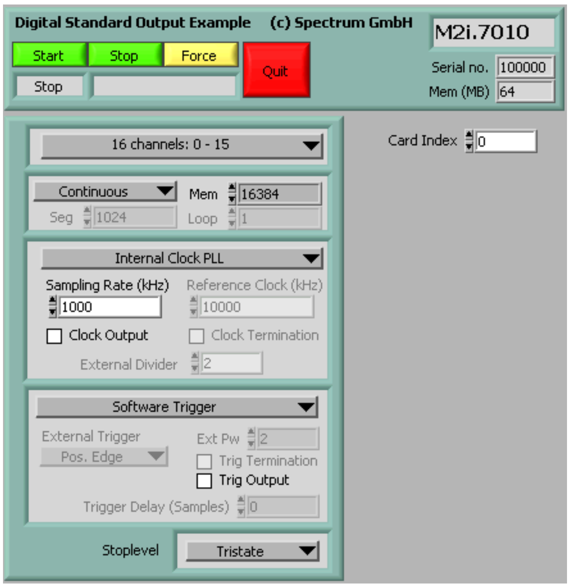
Buttons + Status

Start	starts the card in FIFO output using the current setup on the left.
Stop	stops the current output
Force	forces a trigger if no trigger event is found by the hardware
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current acquisition state. The acquisition state is read from hardware.

Remarks

The example was done to show how FIFO mode is running and to test the system using FIFO mode. It was never designed for maximum speed. To get maximum speed one is requested to use the dwDataWrite_raw functions instead of the sorting functions found in this example.

Digital long replay example (long_digital_rep.vi)



This example shows how to use the output functionality of the digital card with large memory. As the cards can have up to 4 GByte on-board memory one needs to generate/load and transfer this huge amount of data block by block. The example shows how this can be done by generating a simple ramp for the programmed memory. You are requested to modify this example and to merge it with the data source that you need for your application. The example is just done to show how the output works and how to program it.

The User Interface

The user interface was build to allow a fast start with all basic functions. Depending on the used mode and the availability of the card some of the settings may be disabled as they're not available at the moment. If you encounter any error messages from the driver please check the current setup very carefully by examining the hardware manual. The LabVIEW example didn't check for valid combinations as this is done inside the driver.

Card Index

The cards index tells which card of the system should be used for the scope example. Card numbering starts at 0.

Buttons + Status

Start	starts the output with the current setup. Depending on the selected loop mode the card will stop automatically after some time of output or it will run until the user stops it by pressing the „stop" button.
Stop	stops the current output and changes to the stop mode. A new output can be started by using the „start" button again.
Force	forces a trigger if no trigger event is found by the hardware. One click on this button forces one time a trigger event. When using Multiple Replay this means that one segment is replayed after each click of the force trigger button.
Quit	quits the example. Please keep in mind to use the quit button as this makes sure that the driver is correctly un-loaded from memory and is accessible for other software.
Status display	shows the current run mode as well as the current output state. The state is read from hardware.

Setup

On the left there are several setups in a column. Each of these setups directly corresponds with one library function that is explained in the chapter before. Please check these chapters to see the different possible settings.

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_SYNCPROTOCOL	Eh	14	Synchronization protocol versions of synchronized cards 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_TEMPERATURE	47h	71	The temperature of at least one of the card's sensors measures a temperature, that is too high for the hardware.
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_MEMINIT	10Dh	269	On-board memory initialization error. Power cycle the PC and try another PCIe slot (if possible). In case that the error persists, please contact Spectrum support for further assistance.
ERR_POWERSUPPLY	10Eh	270	On-board power supply error. Power cycle the PC and try another PCIe slot (if possible). In case that the error persists, please contact Spectrum support for further assistance.
ERR_ADCCOMMUNICATION	10Fh	271	Communication with ADC failed.P ower cycle the PC and try another PCIe slot (if possible). In case that the error persists, please contact Spectrum support for further assistance.
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.

error name	value (hex)	value (dec.)	error description
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.
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_CONFIGACCESS	208h	520	Internal error occured during config writes or reads. Please contact Spectrum support for further assistance.
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.