Collinear Laser Spectroscopy Data Acquisition System

This document describes the new MIDAS-based Data Acquisition System for Collinear Laser Spectroscopy. It includes an overview of the system, a description of the control windows and how to use them, the definition of the Event-by-Event data format, the file naming conventions used and a list of the Input/Output connections.

Overview

User's View

From a user's perspective the system is, quite simple. It generates output voltages, accepts a number of input signals and builds spectra and event files based on these signals and their timing. The main data acquisition programs run in a VME processor module. The user interacts with them via the MIDAS Graphical User Interface (GUI.)

The system outputs a voltage between 0 and 10Volts and holds it for a user-determined period of time. The voltage is then incremented and the process is repeated. The user decides how many of these voltage steps, or channels, there are. After the last channel the voltage is zeroed and the scan is repeated.

The system can operate in one of two modes "Singles Only" or "Event-by-Event".

Two hardware counters are available to be driven by pulsed signals that could be, for example, signals indicating that a detector has "fired". These scalers are read and stored at the end of each channel; they can be displayed as spectra and stored to disc. In Singles mode the only data taken is from these scalers.

If the user chooses Event-by-Event mode the system also responds to a strobe or trigger signal (possibly a "converted" signal from a TAC) that can occur at any time during a channel period. When the trigger does happen the system reads, via an ADC, a voltage between 0 and 10Volts applied to one of the input pins (this could be the TAC output). It also reads the value of an 8-bit digital input, probably a pattern register indicating which detector segment was hit. Like the scalers these inputs are histogrammed. They are also stored in the Event-by-Event buffer which is saved to disc at the end of a run.

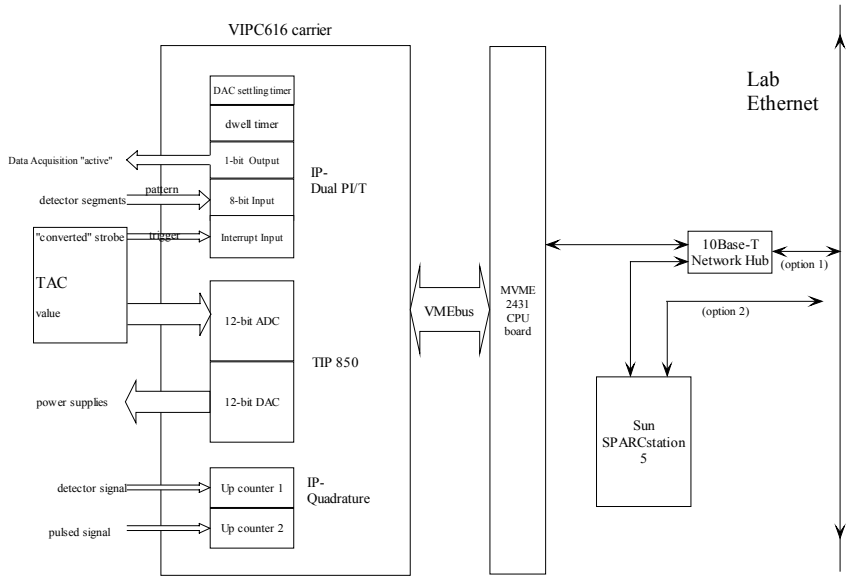
Computing View

From a computing (hardware & software) point of view it is rather more complex. For the sake of brevity some detail is missing from the description that follows.

The data acquisition hardware consists of 5 major components, a single board processor, a digital Input/Output/Timer module, a counter module, a Digital to Analogue to converter and an Analogue to Digital converter. These components all fit onto 2 double height VME cards.

The processor is a Motorola PowerPC VME module (a 2431) running the LynxOS Real-Time POSIX Operating System. The other modules are all standard Industry-Pack modules mounted on a Greenspring VIPC616 VME IP carrier board. The IO module is a Greenspring

Dual-PIT (IP-DPIT). The counter is a Greenspring quadrature decoder (IP-Quadrature). The DAC and the ADC are both part of a single TEWS-Datantechnik module (TIP-850).



The software consists of several autonomous parts: A device driver, a data collection module, MIDAS register and spectrum servers and the MIDAS Graphical User Interface (GUI). The MIDAS user interface runs in the Laser Group's SPARC based Sun workstation, the other parts run in the 2431.

The user interacts with the GUI, sending commands to the VME system, reading status variables and reading data in the form of spectra, which can be displayed or saved to disc.

The 2 MIDAS servers receive and respond to the commands sent from the GUI.

The data collection module reads data from the device driver and interprets it to build appropriate spectra in memory.

The device driver itself has several subsystems: Interrupt Service Routines (ISR), "threads", an "ioctl" interface and a "read" interface.

An ISR is called each time there is an interrupt; it deals with the hardware and notifies the relevant thread that the interrupt has occurred.

The threads can be thought of as semi-independent programs that wait for notification from an ISR of something happening and then take suitable action. The main or "Laser Thread" waits for interrupts from the timers that define the DAC and channel timings. At the end of a channel dwell period the counters are read and their values are sent to the read interface for the collection program. The DAC values are then set up for the next channel.

The "EbyE Thread" waits for interrupts from the IO module. Each time it receives one the thread reads the 8-bit digital register and the ADC. As with the main thread this data is then put in a software queue for the read interface to send to the collection program.

The "ioctls" are software interfaces to the device driver. The register server uses them to set and query the variables that control the behaviour of the threads.

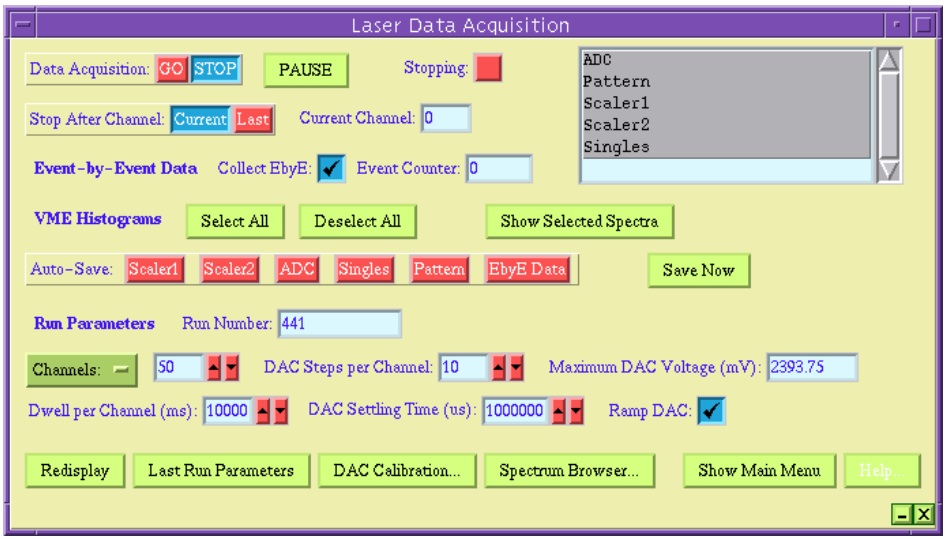
The data collection program repeatedly calls the device driver's read function. The data it receives is structured so that the program can tell if it is singles/counter data from the Laser Thread or event data from the EbyE Thread. In the first case it increments spectra based on the channel value and counter value. In the second case it both increments histograms of the ADC and digital values and stores the data in a buffer to be read out later and stored by the GUI. From a software point of view this buffer is also a spectrum. Readout of all the spectra is carried out by the MIDAS Spectrum Server.

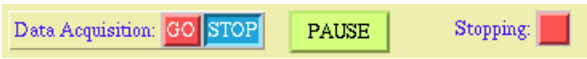
The GUI gives the user an interface to the components already described. By entering values and clicking buttons he or she can set parameters, start and stop the system and check on its state.

The MIDAS Graphical User Interface

When the user starts a MIDAS *laser-session* on a workstation he or she is initially presented with a single control window. The window contains a number of elements or "widgets" that are used to control and monitor the system. In this section of the document the widgets are listed and the function of each is described.

Main Data Acquisition Window





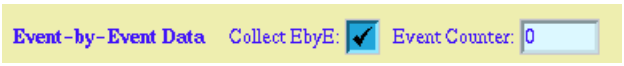
The **GO/STOP** buttons start and stop a Data Acquisition run. The widget also shows whether Data Acquisition is currently running or stopped. The system can be set to automatically save the data (histograms and event-by-event data) when a run stops.

The **PAUSE** button allows the system to be temporarily stopped during a scan. Data Acquisition will halt at the end of the current channel but data will not be saved. The system can continue from the next channel when the same button is clicked. The label on the button changes between PAUSE and CONTINUE depending on the state of the system.

Since the Data Acquisition never stops immediately after a stop request, the **STOPPING** widget is used to show that **STOP** or **PAUSE** has been clicked and Data Acquisition is stopping.

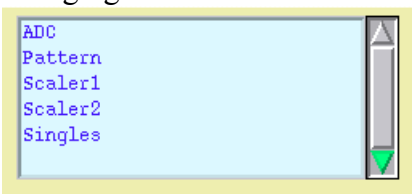
When a scan is stopped it will do so either at the end of the current channel or at the end of all channels in the current scan. The **STOPMODE** widget is used to switch between these alternatives.

A scan consists of a number of channels of a user-specified period. **CHANNEL** shows the current channel during a scan. Since the channel number is continuously changing the displayed value may be behind the real value.



The user uses the **EBYE** *tickbox* to include or exclude Event-by-Event data collection in the next run. This widget is not active during a run, when clicking it will have no effect.

During an Event-by-Event run a software counter numbers the events. The **EVENT** widget is used to keep track of the event number. Since the Event Number could be continuously changing the value shown is likely to be behind the current number.



The **SPECTRUM-LIST** shows the names of all the spectra currently in the VME server. Spectra can be selected for viewing by clicking on the names in the list. Saved spectra can be selected and viewed from the "Spectrum Browser" window.

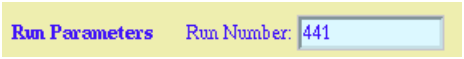
The **SELECT-ALL** button is a quick way of selecting all the spectra in the list.

The **DESELECT-ALL** button deselects all the spectra in the list.

Clicking **SHOW** displays the spectra that are selected in the list. Alternatively a spectrum will be displayed if its name is double-clicked. Spectra are displayed in a separate window.

By default no histograms are saved at the end of a run. **AUTO-SAVE** is used to select which, if any, histograms will be saved when the run ends. Event-by-Event data is collected in a spectrum called EventBuffer in the VME system. It is saved to an EbyE file on disc together with the normal spectra. The spectra, except the EbyE data, are held in Eurogam Unified Spectrum Format.

When **SAVE-NOW** is clicked the "Auto-Save" spectra are immediately saved. N.B. Spectra are saved in directory */home/expt/on-line/specs*.



Each time the **GO** button is clicked a new run number is automatically allocated (by adding one to the current value.) The spectra are named with the run number. **RUN-NUMBER** shows the number of the current run.

The **CHANNELS** widgets are used to specify the number of channels for the next run. The value can either be typed in or common values (50, 100, 200) can be chosen from the menu. During a scan the DAC voltages that are output depend on the number of channels and the number of steps in each channel. Although a new channel value can be entered during a run it does not take effect until the next run is started.

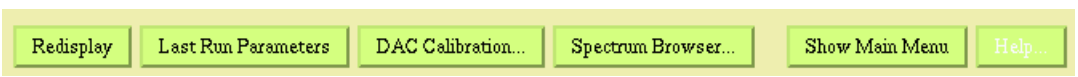
The DAC is, effectively, an 11-bit device with a full range of 2048 steps from 0 to 10V. Each step is equivalent to 4.89mV. The **DAC-STEP** widget is used to define the number of steps between each channel. Although a new value can be entered during a run it does not take effect until the next run is started.

The **DAC-MAX** widget shows the maximum voltage that could be output with the current settings i.e. the voltage of the last channel. This voltage is calculated from the number of channels and the DAC step. It could be greater than the DAC can actually be set to. If it is too large the next run will fail to start. A new value shown here does not take effect until the next run is started.

All the channels have a common dwell time; that is the time during which the DAC voltage is held steady and data collection takes place. The dwell time is measured in milliseconds and is set in the **DWELL** number widget. Although a new value can be entered during a run it does not take effect until the next run is started.

When the DAC voltage is moved to a new value between channels, a short settling time is added to allow the voltage of external equipment to stabilise. The settling time is measured in microseconds and is set in the **SETTLE** number widget. Although a new value can be entered during a run it does not take effect until the next run is started.

Some devices cannot accept large voltage swings from the controlling DAC. Use the **DAC-RAMP** *tickbox* if the DAC should be ramped more slowly between voltages. This widget affects the DAC immediately it is changed.



The **REDISPLAY** button is used to re-read and show the current run state settings. During a run the window is automatically redisplayed every 10 seconds.

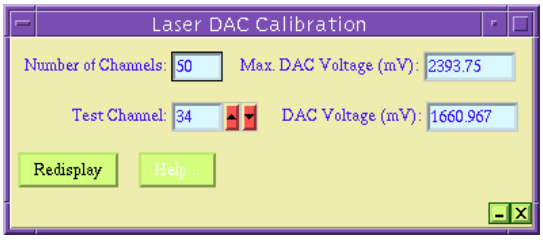
Every time a new run is started the run number, number of channels, DAC step settings, the dwell and settling times and whether to ramp the DAC are saved to file*. These values are reloaded every time a new *laser-session* is started. Sometimes, for example after a server crash or after entering incorrect new values, it is desirable force a reload of the last saved values. The **LAST** button is used for this purpose.

The **CALIBRATION** button opens the DAC Calibration window (described below.)

Click the **SPECTRUM-BROWSER** button to open a spectrum browser. The Spectrum Browser and Spectrum Display windows are used to select and display saved histograms.

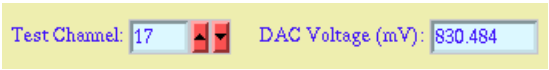
As well as the Laser Data Acquisition functions, the user has access to some standard MIDAS functions which are accessible through the **MAIN-MENU**.

DAC Calibration Window



The **CHANNELS** widget shows the number of channels for the run being calibrated. The value is set in the main window.

The **DAC-MAX** widget shows the maximum voltage that could be output with the current settings. This voltage is calculated from the number of channels and the DAC step; it could be greater than the DAC can actually generate. However, the system will never attempt to set the DAC beyond its range. The DAC in use has a 0 to 10V range.



The **TEST** widget is used to enter the number of the channel under test here. The test channel is the one being calibrated. Its range is 0 to one less than the number of channels. The DAC will generate a voltage depending on the DAC step and the value if the test channel.

VALUE shows the voltage that the DAC has been set to for the test channel. This value is the voltage the DAC is actually set to. It is never greater than the DAC's maximum (10V) even if the calculated maximum voltage is greater.

Using the System - An example

- Login to the Sun workstation as user *expt.*

* The present implementation saves these values in `/MIDAS/experiments/laser/run-parameters`

- In a cmdtool or other terminal window enter the command `laser-session`, which will start a number of windows. The main one will be the **Main Data Acquisition Window** described above.
- Check that the values for number of channels, DAC step, dwell and settle times are suitable. Change the **Ramp DAC** tick if necessary.
- Select **Collect EbyE data** if appropriate.
- Choose the spectra, and Event Buffer, which should be auto-saved at the end of the run.
- Optionally click on **DAC Calibration** and note the real voltages for your test channels.
- Click on the **GO** button in the main window.
- Select the spectra you are interested in from the list and click **SHOW.**

Event-by-Event Data

EbyE data collection

During data collection Event-by-Event data is stored in VME memory as a spectrum. If EventBuffer is selected for Auto-Save the latest data will be copied to disc every 2 minutes. At the end of the run the whole buffer will be copied to disc. On disc the EbyE data is held as a binary file of data only, it is not in Eurogam Spectrum format.

Format of an Event-by-Event Data File

Number of Events (n)
Event Block 0
Event Block 1
:
Event Block n-1

The Number of Events word is a 32-bit unsigned integer.

Format of an Event-by-Event Data Event Block

Block Header Item
Event Block Data Item
Event Block Data Item
:
Block Trailer

Format of Event-by-Event Block Items

Each Data Block Item is a 32-bit unsigned integer.
Each block consists of an 8-bit token that defines the type and a 24-bit data field.

Token	Data

A number of tokens have been defined. They have not all been implemented.

SINGLES_BLOCK	0xf1
EBYE_BLOCK	0xf2
END_BLOCK	0xff
EVENT_NUMBER	0xe0
SCALER_1	0xe1
SCALER_2	0xe2
SCALER_3	0xe3
SCALER_4	0xe4
SINGLES	0xe5
ADC_DATUM	0xe6
HIT_PATTERN	0xe7

Format of Block Delimiter Items

Token	Data
SINGLES_BLOCK	Channel number
EBYE_BLOCK	Channel number
END_BLOCK	0xffffff

Current Implementation

All events are in the following format.

Token	Data
EBYE_BLOCK	Channel number
ADC_DATUM	ADC data
HIT_PATTERN	Hit pattern
END_BLOCK	0xffffff

Example EbyE data file contents

File contents	Comment
00000003	3 events in the file
F2000000	EbyE data, channel 0
E600023D	ADC data, value 0x23d
E7000020	Hit pattern, segment 5 fired
FFFFFFFF	End of block
F2000000	EbyE data, channel 0
E600089A	ADC data, value 0x89a
E7000001	Hit pattern, segment 0 fired
FFFFFFFF	End of block
F2000001	EbyE data, channel 1
E6001234	ADC data, value 0x1234
E7000080	Hit pattern, segment 7 fired
FFFFFFFF	End of block

A Note on 32-bit integer formats

The data in an EbyE file is held as "big-endian 32-bit integers". i.e. it is generated on a PowerPC/SPARC system. If it is to be analysed on a little-endian system e.g. a PC (running no matter what Operating System - Windows, Linux, BeOS...) it must be converted before it will make sense.

File Conventions

The histograms in the VME system are named according to the data they hold. They are cleared and re-labelled at the start of each run.

ADC
Pattern
Scaler1
Scaler2
Singles
(and EventBuffer)

When they are saved to disc the name is prepended with Run*N, w*here *N* is the run number. E.g.

Run123.ADC
Run123.Pattern
Run123.Scaler1
Run123.Scaler2
Run123.Singles

During a run the data from the EventBuffer spectrum is copied to:
Run*N*.EbyEData.tmp

At the end of a run the Event-by-Event data is saved to:
Run*N*.EbyEData

By default the spectrum and EbyEData files are saved to */home/expt/on-line/specs*.

Industry-Pack Modules

The software controls input and output through 3 Industry-Pack (IP) modules mounted on a VME carrier board. Signals are taken to and from the system via 3 50-way IDC ribbon cables.

Module Function	**Manufacturer**	**Module Name**
VME Carrier Board	SBS Technologies	VIPC 616
Digital IO/Timers	SBS Technologies	IP-DualPI/T
Counters/Scalers	SBS Technologies	IP-Quadrature
DAC/ADC	TEWS-Datentechnik	TIP-850

The signals used by the system are shown in the table below.

Module	Pins	Signal Name	Function	Input/Output
IP-Dual PI/T	1-8	X, Port A	Hit Pattern	Input (TTL)
IP-Dual PI/T	17,19,21,23	X, Ground		
IP-Dual PI/T	20	X, H2	Strobe/Trigger	Input (TTL)
IP-Dual PI/T	25	Ground		
IP-Dual PI/T	42,44,46,48	Y, Ground		
IP-Dual PI/T	26	Y, Port A0	Active Out	Output (TTL)
IP-Dual PI/T	25	Ground		
IP-Quadrature	3	X1	Scaler1	Input (TTL)
IP-Quadrature	2,4,6,8	Ground		
IP-Quadrature	7	Y1	Ground	Ground
IP-Quadrature	15	X2	Scaler2	Input (TTL)
IP-Quadrature	14,16,18,20	Ground		
IP-Quadrature	19	Y2	Ground	Ground
TIP850	1	ADC 1 Input	ADC	Input (-10V-+10V
TIP850	4	AGND	Analogue Ground	
TIP850	27	DAC 1 Output	DAC	Output (-10V-+10V
TIP850	28	AGND	Analogue Ground	