measComp

unknown

2023-July-31

CONTENTS

author
: Mark Rivers, University of Chicago

This package provides EPICS drivers for the some of the USB and Ethernet I/O modules from Measurement Computing.

The software is located in the measComp github repository.

CHAPTER

ONE

REQUIRED MODULES

Required module	Required for
EPICS base	Base support
asyn	Driver and device support
autosave	Save/restore support
busy	Busy record support
mca	mca record support
scaler	Scaler record support.
seq	State notation language sequencer. Used in MCS mode with USB-CTR08 and for std.

The required versions of each of the above modules for a specific release of measComp can be determined from the measComp/configure/RELEASE file.

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Overview

This package provides EPICS drivers for the some of the USB and Ethernet I/O modules from Measurement Computing. Currently the USB-CTR04/08, and multi-function modules (E-1608, USB-1208LS, USB-1208FS, USB-1608G, USB-1608GX-2AO, USB-1808/1808X, USB-231, USB-2408-2AO, E-TC, TC-32, USB-TEMP, USB-TEMP-AI, and E-DIO24) are supported. The multi-function modules support analog input and/or output, temperature input (USB-2408-2AO, USB-TEMP, USB-TEMP-AI, E-TC, TC-32), digital input/output, pulse counters (all but TC-32), and pulse generators (USB-1608G and USB-1608GX-2AO).

Support for other modules is straightforward to add and can be done as the demand arises.

This module is supported on both Windows and Linux, 64-bit and 32-bit.

On Windows it uses the Measurement Computing "Universal Library" (UL), which is only available on Windows.

In R4-0 and later it uses the UL for Linux library from Measurement Computing for Linux drivers. This is an [open-source library available on Github](https://github.com/mccdaq/uldaq). The Linux Universal Library API is similar to the Windows UL API, but the functions have different names and different syntax.

UL for Windows and Linux support most current Measurement Computing models.

In versions prior to R4-0 the Linux support used the [low-level drivers from Warren Jasper](https://github.com/wjasper/Linux_Drivers). On top of these drivers the module provides a layer that emulates the Windows UL library from Measurement Computing. The EPICS drivers thus always use the Windows UL API and are identical on Linux and Windows. The E-1608, E-TC, E-TC32, E-DIO24, USB-1608G-2AO, USB-CTR08, USB-TEMP, USB-TEMP-AI and USB-31XX models are supported in these versions.

2.2 Driver for Multi-Function Devices

author
Mark Rivers, University of Chicago

Contents

2.2.1 Introduction

This is an EPICS driver for the multi-function devices from MeasurementComputing. These multi-function devices support support analog input, temperature input (thermocouple, RTD, thermistor, and semiconductor), analog output, binary I/O, counters, and timers. Not all devices have all of these capabilities.

The driver is written in C++, and consists of a class that inherits from asynPortDriver, which is part of the EPICS asyn module.

The driver is written to be general, so that it can be used with any Measurement Computing multi-function module. It uses the introspection capabilities of their UL library to query many of the device features. However, there are some features that cannot be queried, so the driver does require small modifications to be be used with a new model.

2.2.2 Supported models

The following models are currently supported.

E-1608

Fig. 1: **Photo of E-1608**

This module costs $525 and has the following features:

- 16-bit analog inputs
  - 8 single-ended channels or 4 differential channels
  - Programmable per-channel range: +-1V, +-2V, +-5V, +-10V
  - 250 kHz total maximum input rate, i.e. 1 channel at 250 kHz, 2 channels at 125 kHz, etc.
  - Internal or external trigger.
  - Internal or external clock for input signals.
  - Input FIFO, unlimited waveform length
- 16-bit analog outputs

- 2 channels, fixed +-10V range
- No output waveform capability

- Digital inputs/outputs
  - 8 signals, individually programmable as inputs or outputs
- Counter
  - 1 input
  - 10 MHz maximum rate, 32-bit register

More information can be found in the E-1608 product description.

The following is the main medm screen for controlling the E-1608.

E-TC

This module costs $505 and has the following features:

- Ethernet interface.
- 8 thermocouple inputs
  - 8 channels with cold-junction compensation. Types J, K, T, E, R, S, B, and N.
  - 4 samples/s.
- Digital inputs/outputs
  - 8 signals, individually programmable as inputs or outputs
- Counters
  - 1 input
  - 10 MHz maximum rate, 32-bit register

More information can be found in the E-TC product description.

The following is the main medm screen for controlling the E-TC.

TC-32

This module costs $1999 and has the following features:

- USB and Ethernet interfaces, either can be used.
- 32 thermocouple inputs
  - 32 channels with cold-junction compensation. Types J, K, T, E, R, S, B, and N.
  - 3 samples/s if reading all 32 channels, faster if reading fewer.
- Digital inputs
  - 8 digital inputs, switch-selectable pullup resistor
- Digital outputs
  - 32 digital inputs, switch-selectable pullup resistor

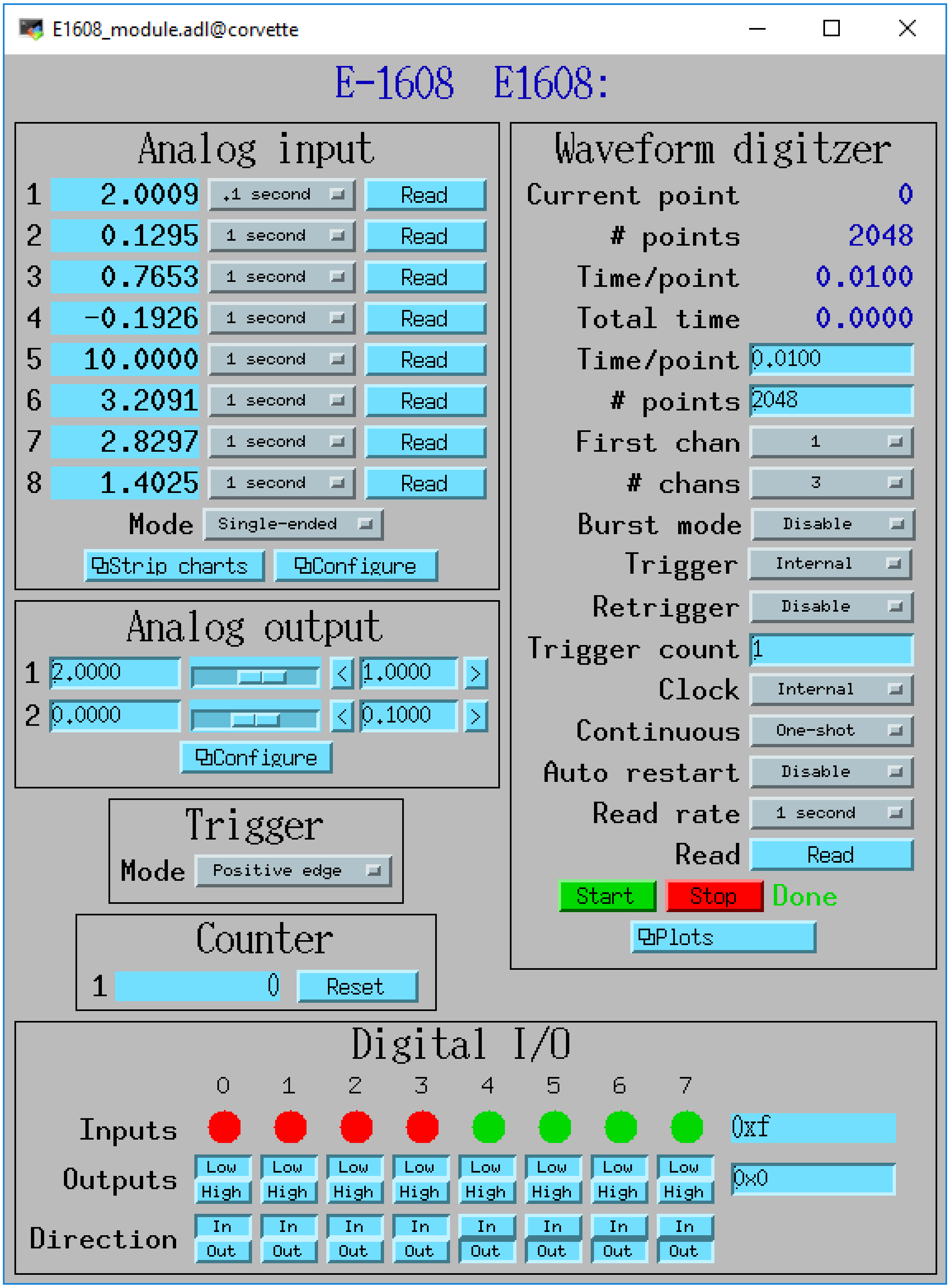

Fig. 2: **E1608_module.adl**

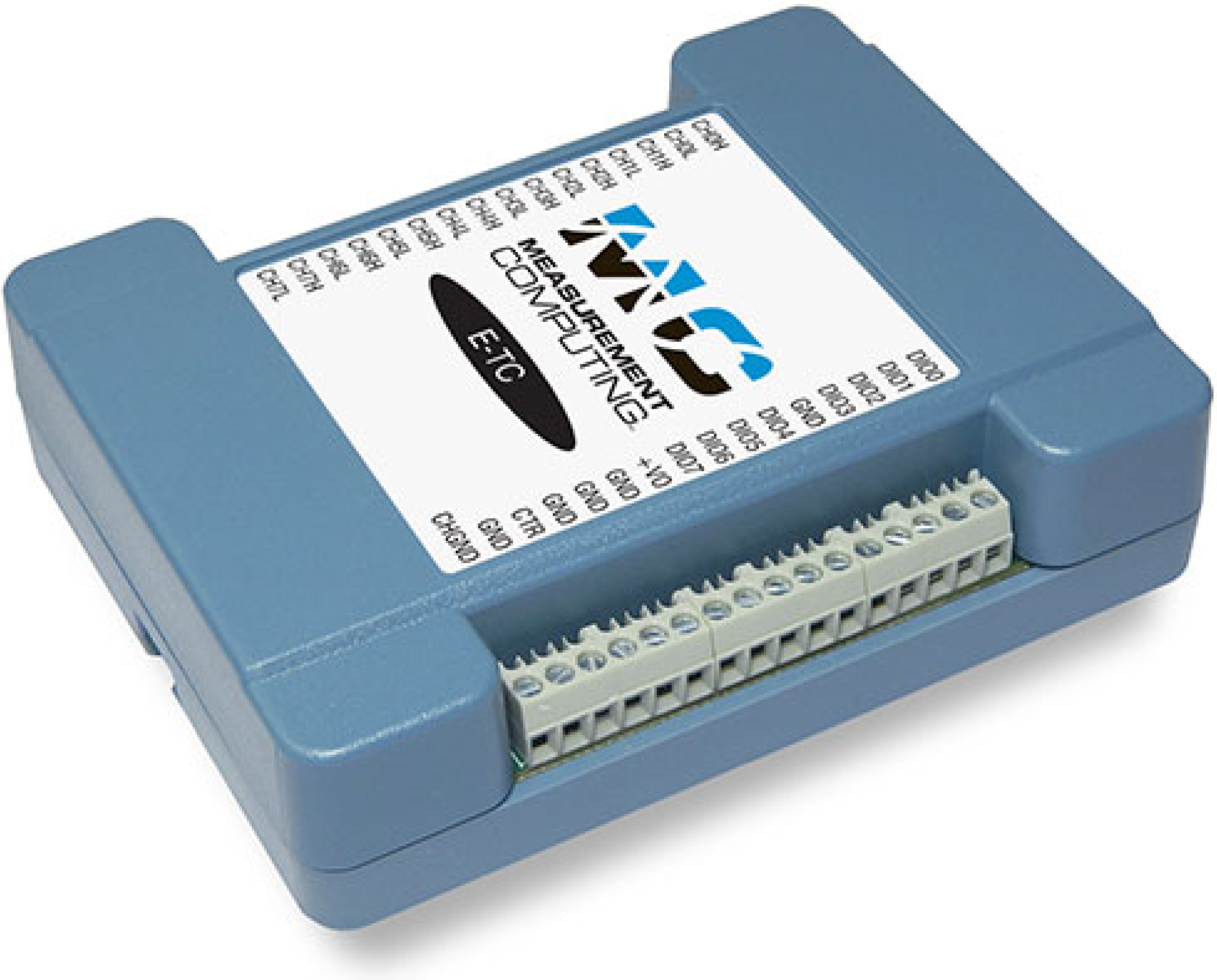

Fig. 3: **Photo of E-TC**

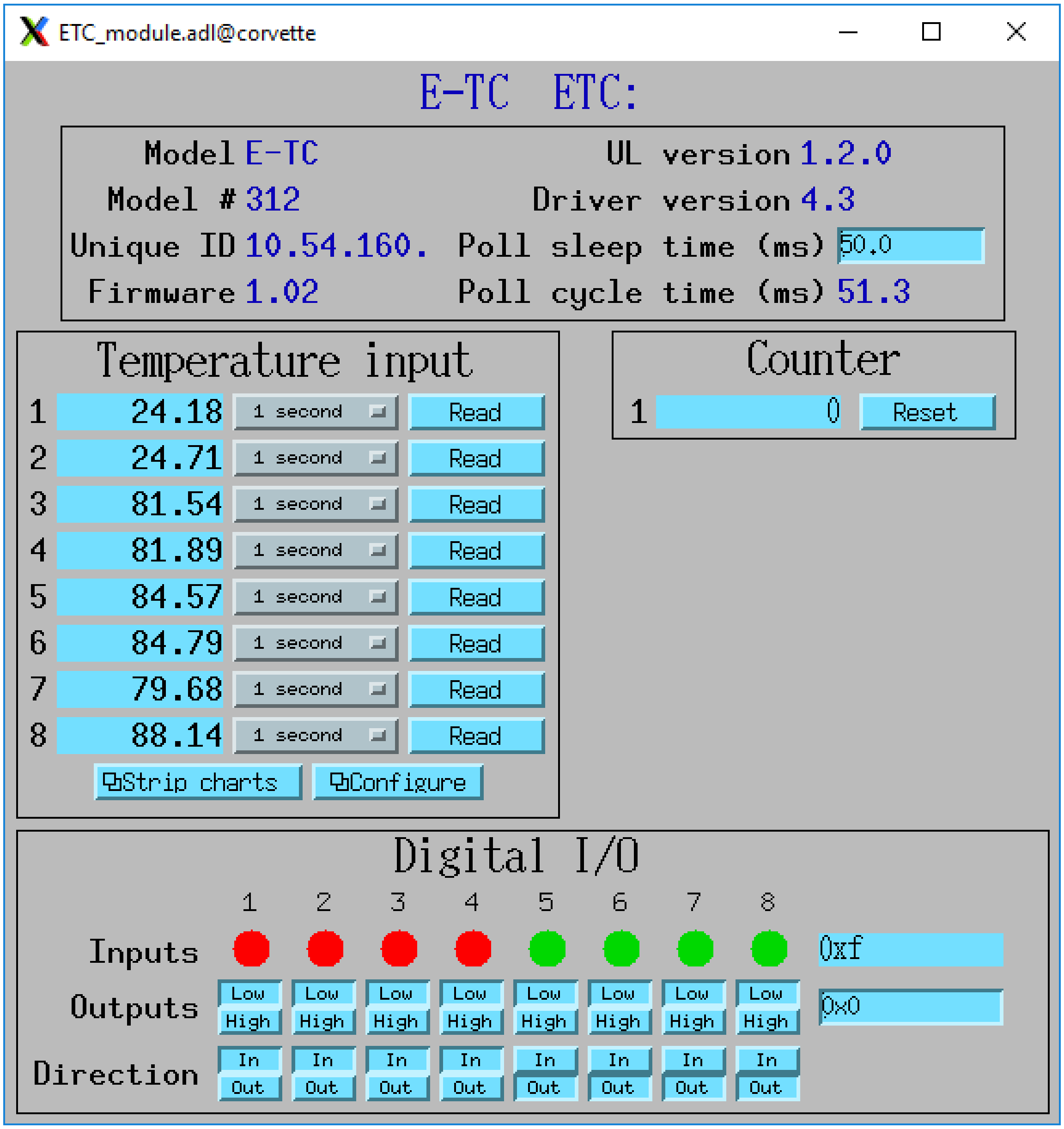

Fig. 4: **ETC_module.adl**

Fig. 5: **Photo of TC-32**

  - Each output can either be controlled by software or can be controlled by the alarm status of the corresponding thermocouple. Flexible alarm configuration, i.e. hysteresis.

More information can be found in the TC-32 product description.

The following is the main medm screen for controlling the TC-32.

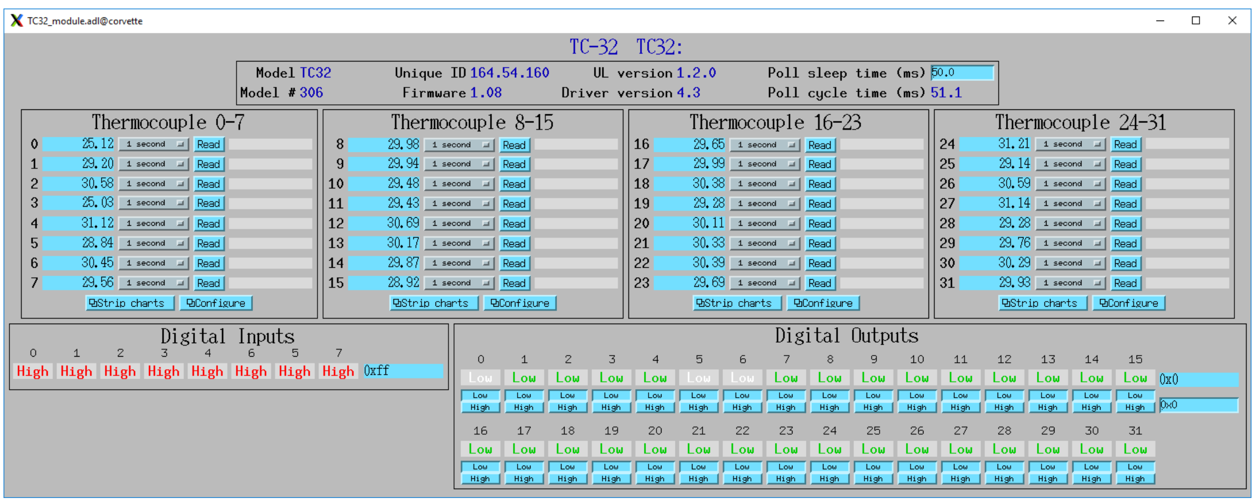

Fig. 6: **TC32_module.adl**

USB-1608G and USB-1608GX-2AO

This module costs $799 and has the following features:

- 16-bit analog inputs
  - 16 single-ended channels or 8 differential channels
  - Programmable per-channel range: +-1V, +-2V, +-5V, +-10V
  - 500 kHz total maximum input rate, i.e. 1 channel at 500 kHz, 8 channels at 62.5 kHz, etc.
  - Internal or external trigger. External trigger shared with analog outputs.
  - Internal or external clock, input and output signals.

Fig. 7: **Photo of USB-1608GX-2AO**

    - 4 kSample input FIFO, unlimited waveform length

- 16-bit analog outputs
    - 2 channels, fixed +-10V range
    - 500 kHz total maximum output rate, i.e. 1 channel at 500 kHz, 2 channels at 250 kHz
    - Internal or external trigger. External trigger shared with analog inputs.
    - Internal or external clock, input and output signals
    - 2 kSample output FIFO, unlimited waveform length
- Digital inputs/outputs
    - 8 signals, individually programmable as inputs or outputs
- Pulse generator
    - 1 output
    - 64MHz clock, 32-bit registers
    - Programmable period, width, number of pulses, polarity
- Counters
    - 2 inputs
    - 20 MHz maximum rate, 32-bit registers

More information can be found in the USB-1608GX-2AO product description.

The USB-1608G is very similar to the USB-1608GX-2AO except that it does not have any analog outputs and the analog inputs are limited to 250 kHz rather than 500 kHz. More information can be found in the USB-1608G product description.

The following is the main medm screen for controlling the USB-1608GX-2AO.

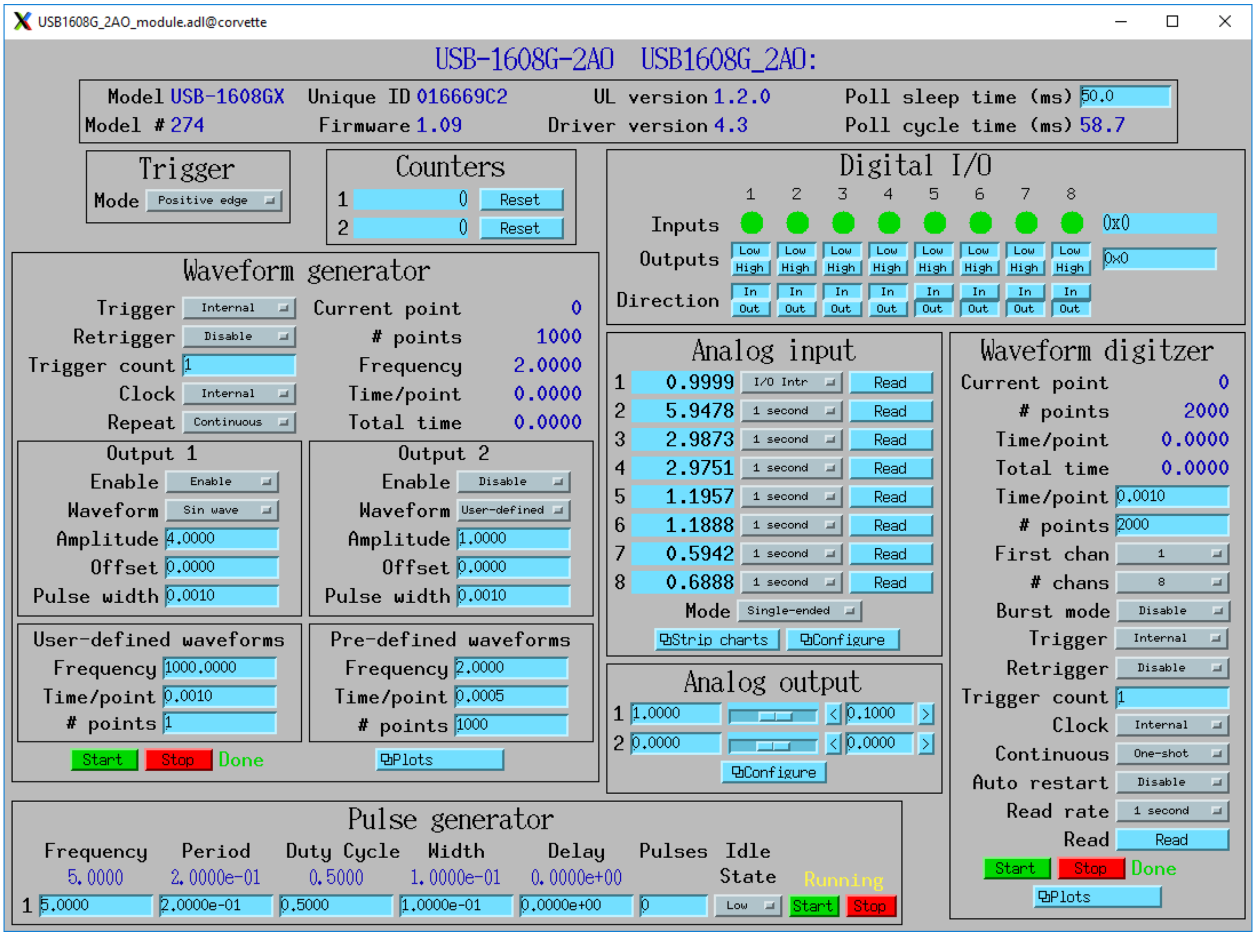

Fig. 8: **1608G_module.adl**

USB-1808 and USB-1808X

These modules cost $769 and $989 and have the following features:

- 18-bit analog inputs
  - 8 single-ended or differential channels
  - Programmable per-channel range:+-5V, +-10V, 0-5V, 0-10V
  - USB-1808: 125 kHz total maximum input rate, i.e. 1 channel at 125 kHz, 8 channels at 15.625 kHz, etc.
  - USB-1808X: 500 kHz total maximum input rate, i.e. 1 channel at 500 kHz, 8 channels at 62.5 kHz, etc.
  - Internal or external trigger. External trigger shared with analog outputs.
  - Internal or external clock, input and output signals.
  - 4 kSample input FIFO, unlimited waveform length
- 16-bit analog outputs
  - 2 channels, fixed +-10V range

Fig. 9: **Photo of USB-1808**

  - USB-1808: 250 kHz total maximum output rate, i.e. 1 channel at 250 kHz, 2 channels at 125 kHz
  - USB-1808X: 1000 kHz total maximum output rate, i.e. 1 channel at 1000 kHz, 2 channels at 500 kHz
  - Internal or external trigger. External trigger shared with analog inputs.
  - Internal or external clock, input and output signals
  - 2 kSample output FIFO, unlimited waveform length
- Digital inputs/outputs
  - 4 signals, individually programmable as inputs or outputs
- Pulse generator
  - 2 outputs
  - 100 MHz clock, 32-bit registers
  - Programmable period, width, number of pulses, polarity
- Counters
  - 2 inputs
  - 50 MHz maximum rate, 32-bit registers
- Quadrature encoder inputs
  - 2 inputs
  - 50 MHz maximum rate, 32-bit registers

More information can be found in the USB-1808 product description.

The following is the main medm screen for controlling the USB-1808.

USB-2408-2AO

This module costs $699 and has the following features:

- 24-bit analog inputs
  - 16 single-ended channels or 8 differential channels
  - Programmable per-channel range: 8 ranges from +-0.078V to +-10V
  - Thermocouple support for 8 channels with cold-junction compensation. Types J, K, T, E, R, S, B, or N.
  - 1 kHz total maximum input rate, i.e. 1 channel at 1 kHz, 8 channels at 125 Hz, etc.
  - Input FIFO, unlimited waveform length
- 16-bit analog outputs
  - 2 channels, fixed +-10V range
  - 1000 Hz total maximum output rate, i.e. 1 channel at 1000 Hz, 2 channels at 500 Hz
  - Output FIFO, unlimited waveform length
- Digital inputs/outputs
  - 8 signals, individually programmable as inputs or outputs
- Counters
  - 2 inputs

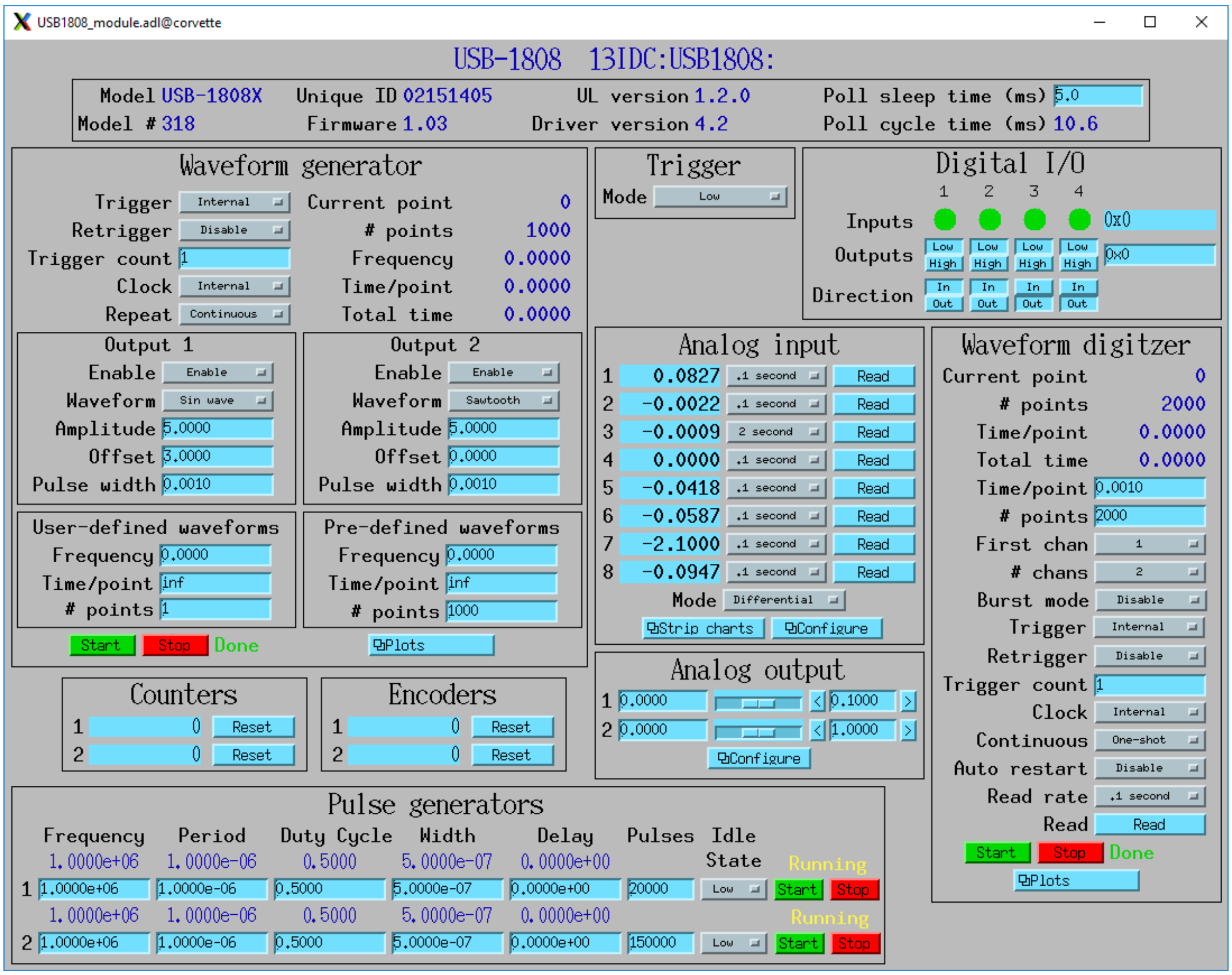

Fig. 10: **1808_module.adl**

Fig. 11: **Photo of Photo of USB-2408-2AO**

– 1 MHz maximum rate, 32-bit registers

More information can be found in the USB-2408-2AO product description.

The following is the main medm screen for controlling the USB-2408-2AO.

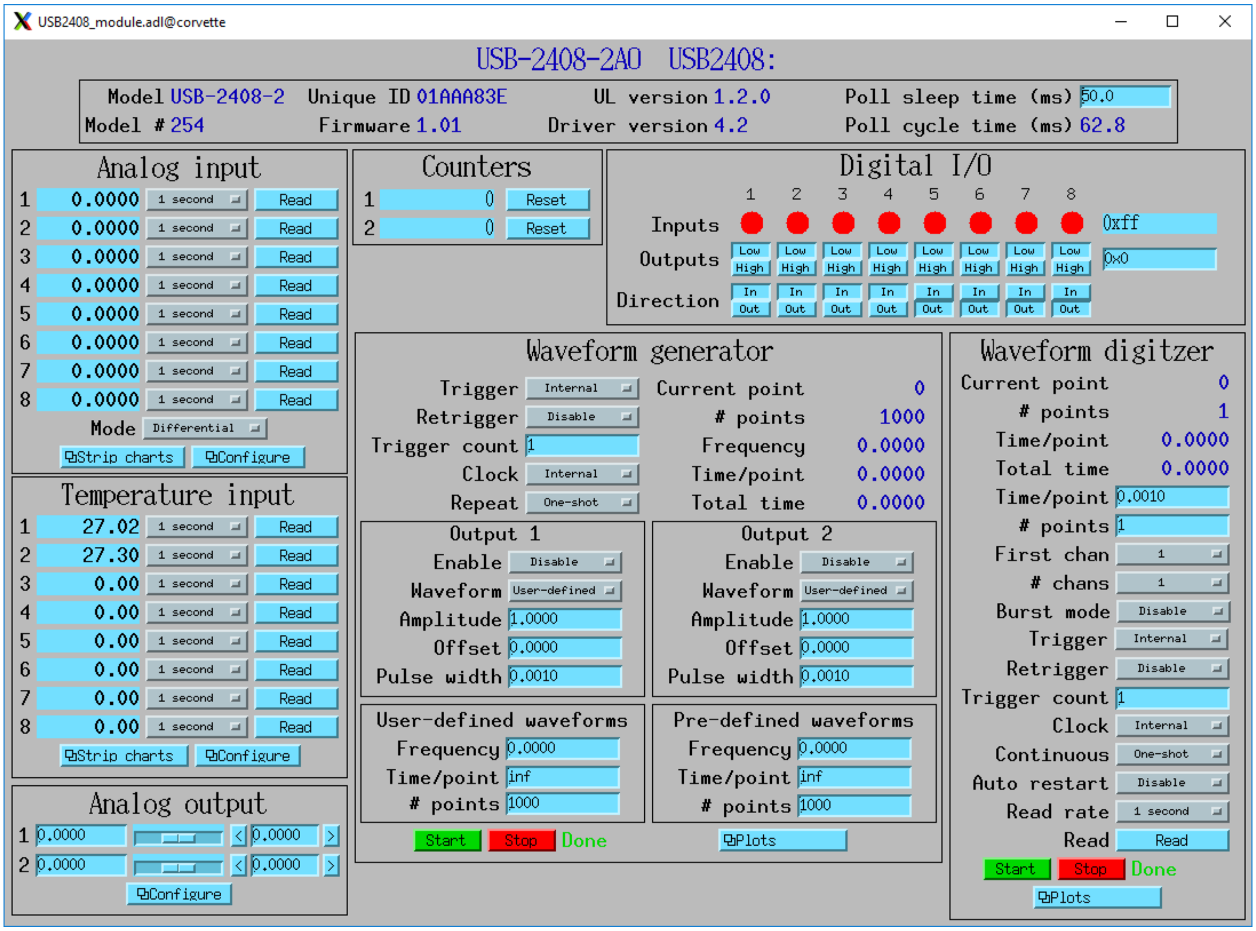

Fig. 12: **2408_module.adl**

USB-TEMP and USB-TEMP-AI

The USB-TEMP costs $605 and the USB-TEMP-AI costs $795. They have the following features:

- Temperature inputs
  - 8 temperature inputs on USB-TEMP, 4 on USB-TEMP-AI. These can be platinum resistance thermometers (RTD), thermocouples, thermistors, or semiconductor sensors.
  - Thermocouple support has cold-junction compensation. Types J, K, T, E, R, S, B, or N.
  - 2 samples/s per channel.
- 24-bit analog inputs (USB-TEMP-AI only)
  - 4 channels
  - Programmable per-channel range: 4 ranges from +-1.25V to +-10V
- Digital inputs/outputs

Fig. 13: **Photo of Photo of USB-TEMP**

- 8 signals, individually programmable as inputs or outputs

- Counters
  - 1 input
  - 1 MHz maximum rate, 32-bit register

More information can be found in the USB-TEMP product description.

The USB-TEMP and USB-TEMP-AI behave differently from all other Measurement Computing devices. On Windows InstaCal is used to select the temperature sensor type (RTD, thermocouple, etc.) and the RTD wiring configuration. Those settings are written into non-volatile memory on the device, and cannot be changed with EPICS. However, they **can** be changed with EPICS on Linux, so they are exposed in the OPI screen.

The following is the main medm screen for controlling the USB-TEMP-AI.

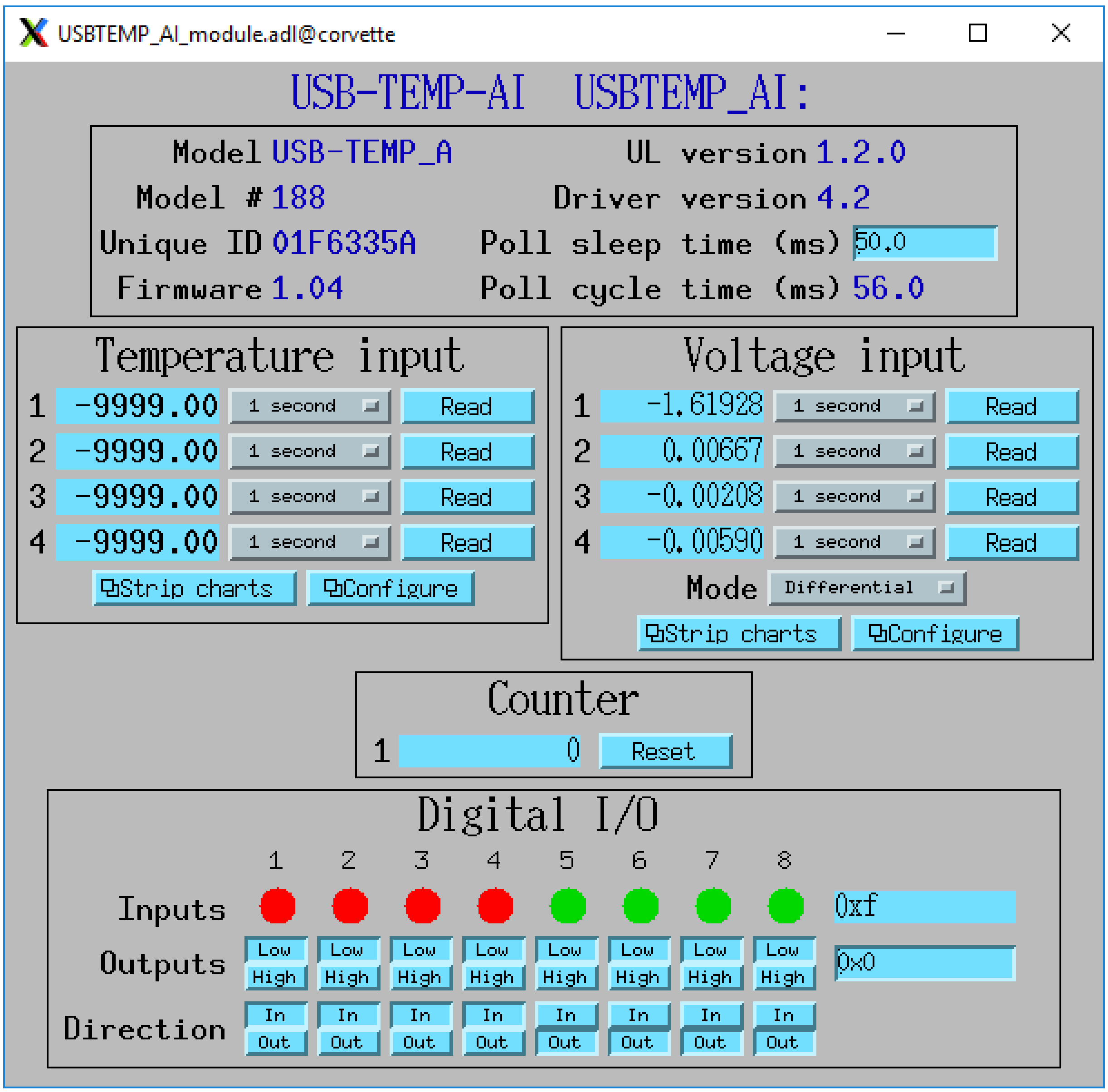

Fig. 14: **USBTEMP_AI_module.adl**

The following is the screen for configuring the temperature inputs.

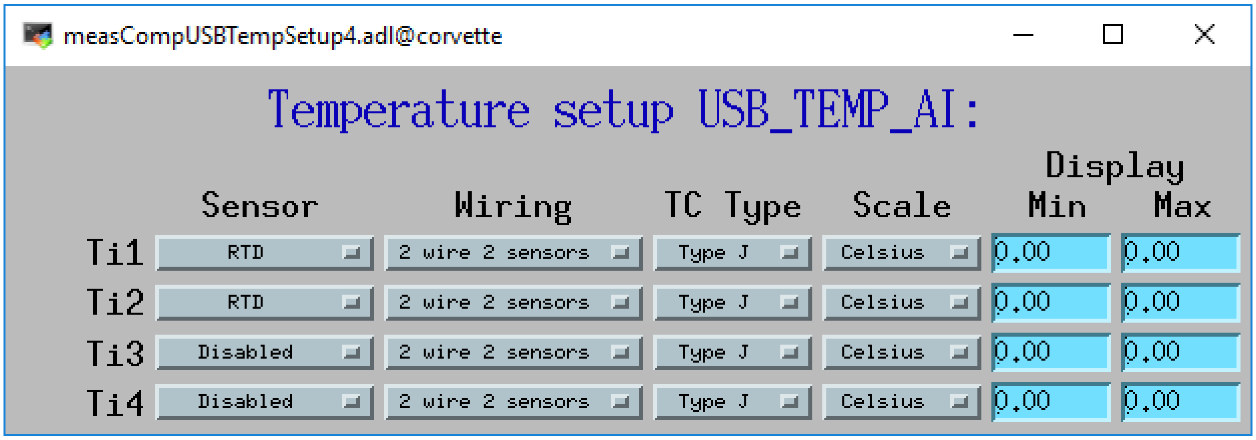

Fig. 15: **measCompUSBTempSetup4.adl**

USB-1208LS

This module costs $129 and has the following features:

- 12-bit analog inputs
  - 4 differential channels
  - Programmable per-channel range: 8 ranges from +-1V to +-20V
  - 50 Hz maximum sampling rate. The module has a trigger input that allows higher sampling rates, but this is not yet supported in the EPICS driver.
- 10-bit analog outputs
  - 2 channels, fixed 0 to +5V range
  - 100 Hz maximum input rate
- Digital inputs/outputs
  - 16 signals, programmable as inputs or outputs in groups of 8
- Counters
  - 1 input
  - 1 MHz maximum rate, 32-bit register

More information can be found in the USB-1208LS product description.

The USB-1208HS , USB-1208FS-Plus and USB-231 are similar devices but with higher performance. These are also supported.

The following is the main medm screen for controlling the USB-1208LS.

Fig. 16: **Photo of USB-1208LS**

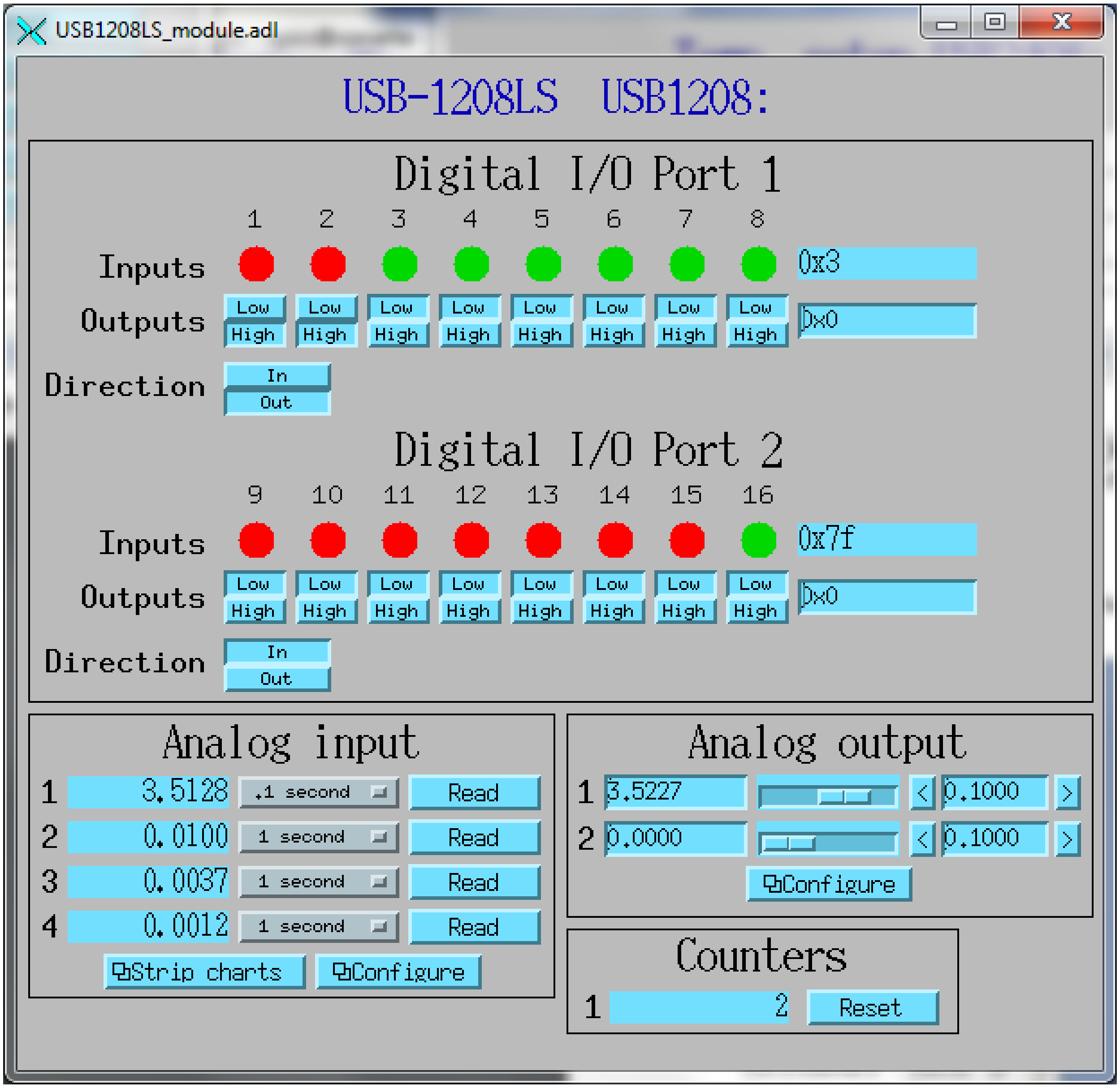

Fig. 17: **USB1208LS_module.adl**

E-DIO24

Fig. 18: **Photo of E-DIO24**

This module costs $320 and has the following features:

- Digital inputs/outputs
  - 24 signals, individually programmable as inputs or outputs
- Counters
  - 1 input
  - 10 MHz maximum rate, 32-bit register

More information can be found in the E-DIO24 product description.

The following is the main medm screen for controlling the E-DIO24.

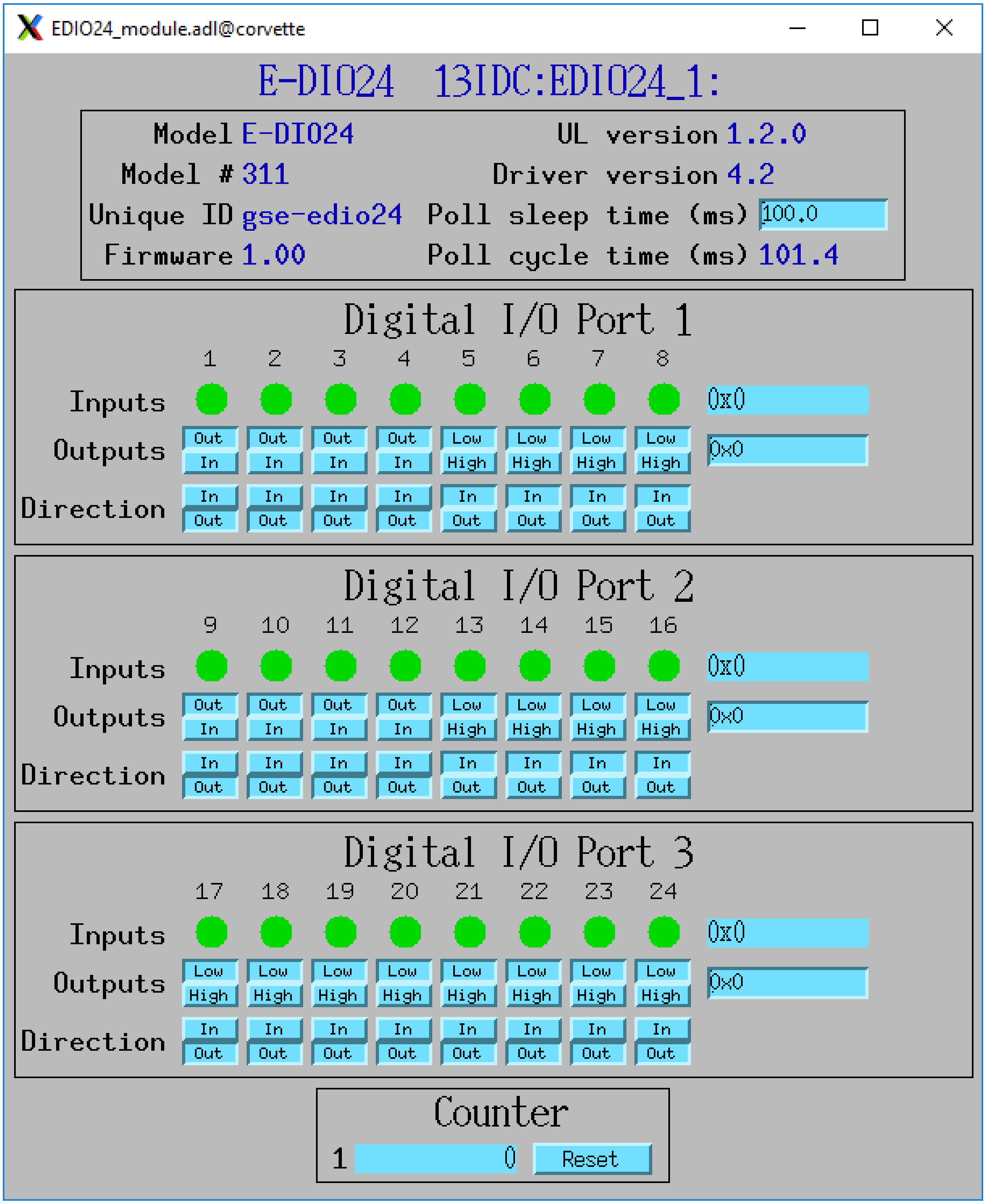

Fig. 19: **EDIO24_module.adl**

USB-3100

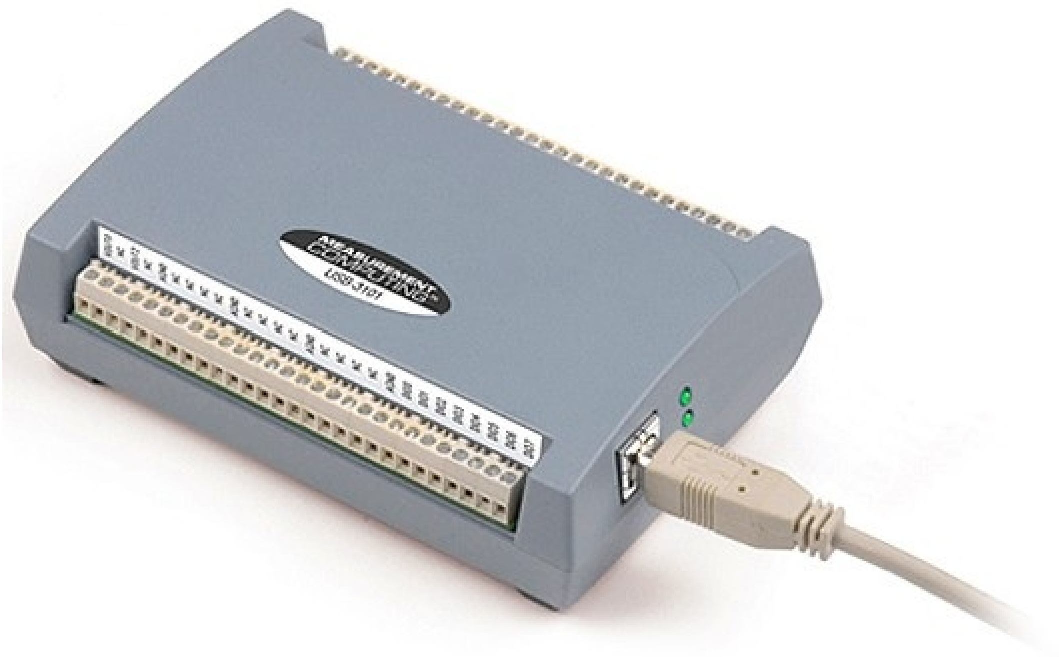

Fig. 20: **Photo of USB-3101**

This series of module costs from $330 (USB-3101) to $660 (USB-3106) depending on the number of channels and the output type, and has the following features:

- 16-bit analog outputs
  - 4, 8 or 16 channels, individually programmable range 0-10V or +-10V.
  - Some models provide 0-20 mA current output as well as voltage output
  - Some models have high-drive voltage output (+-40 mA)
  - 100 Hz maximum output rate
- Digital inputs/outputs
  - 8 signals, individually programmable as inputs or outputs
- Counters
  - 1 input

  - 1 MHz maximum rate, 32-bit register

More information can be found in the USB-3100 series product description.

The following is the main medm screen for controlling the USB-3104 8-channel unit.

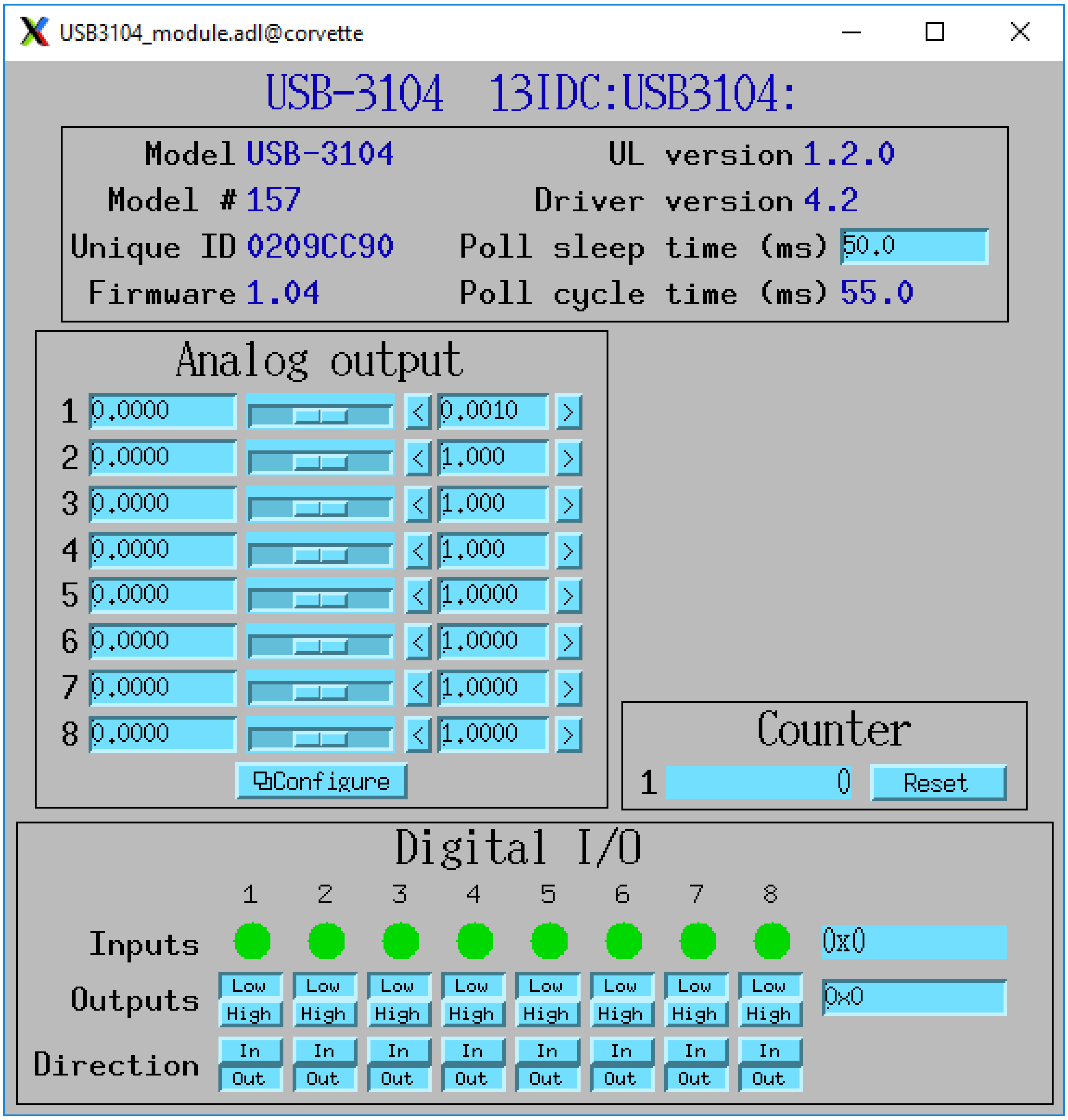

Fig. 21: **USB3104_module.adl**

The following is the medm screen for configuring the analog outputs on the USB-3104 8-channel unit.

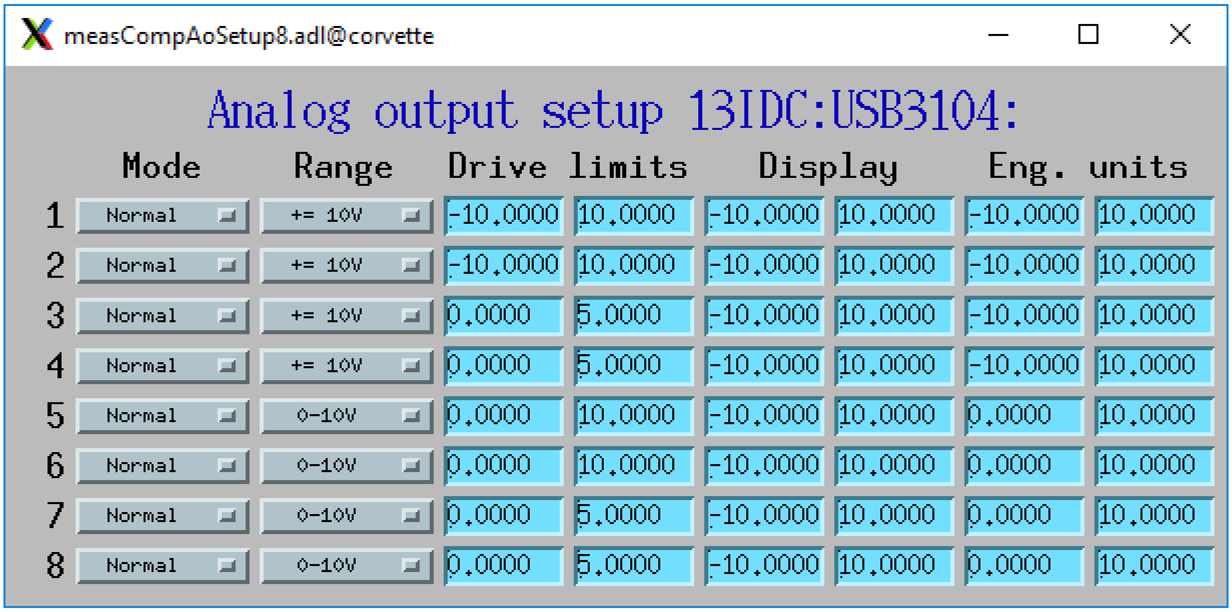

Fig. 22: **USB3104_setup.adl**

2.2.3 Configuration

The following lines are needed in the EPICS startup script for the multifunction driver.

```
## Configure port driver
# MultiFunctionConfig(portName,        # The name to give to this asyn port driver
#                     uniqueID,        # For USB the serial number.  For Ethernet the MAC address or IP address.
#                     maxInputPoints,  # Maximum number of input points for waveform digitizer
#                     maxOutputPoints) # Maximum number of output points for waveform generator
MultiFunctionConfig("1608G_1", 1, 1048576, 1048576)
dbLoadTemplate("1608G.substitutions.big")
```

The uniqueID is a string that identifies the device to be controlled.

- For USB devices the uniqueID is the serial number, which is printed on the device (e.g. "01F6335A").
- For Ethernet devices the uniqueID can either be the MAC address (e.g. "00:80:2F:24:53:DE"), or the IP address (e.g. "10.54.160.63", or the IP DNS name (e.g. "gse-e1601-1"). The MAC address, IP address or IP name can be used for devices on the local subnet, while the IP address or IP name must be used for devices on other subnets.

The measComp module comes with example iocBoot/ directories that contain example startup scripts and example substitutions files for each supported model.

2.2.4 Databases

The following tables list the database template files that are used with the multi-function modules.

Overall Device Functions

These are the records defined in measCompDevice.template. This database is loaded once for each module.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
$(P)ModelName	stringin	asynOctetRead	MODEL_NAME	The model name of this device, e.g. "USB-1808X".
$(P)ModelNumber	longin	asynInt32	MODEL_NUMBER	The model number of this device, e.g. 318.
$(P)FirmwareVersion	stringin	asynOctetRead	FIRMWARE_VERSION	The firmware version, e.g. "1.03".
$(P)UniqueID	stringin	asynOctetRead	UNIQUE_ID	The unique ID of this device, e.g. "02151405"
$(P)ULVersion	stringin	asynOctetRead	UL_VERSION	The version of the UL library on Linux or Windows, e.g. "1.2.0".
$(P)DriverVersion	stringin	asynOctetRead	DRIVER_VERSION	The version of the EPICS driver, e.g. "4.3".
$(P)PollTimeMS	ai	asynFloat64	POLL_TIME_MS	The actual time for the last poll cycle in ms.
$(P)PollSleepMS	ao	asynFloat64	POLL_SLEEP_MS	The time to sleep at the end of each poll cycle in ms.
$(P)LastErrorMessage	waveform	asynOctetRead	LAST_ERROR_MESSAGE	The last error message from the driver.

The medm sub-screen that displays these records. The main screen for every module contains a subscreen like this.

Fig. 23: **measCompDevice.adl**

Analog I/O Functions

These are the records defined in measCompAnalogIn.template. This database is loaded once for each analog input channel

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)	ai	asynInt32	ANALOG_IN_VALUE	Analog input value. This is converted from the 16-bit unsigned integer device units from the driver to engineering units using the EGUL and EGUF fields. This value is polled in the driver at the polling frequency set by PollSleepMS. The asynInt32Average device support is used, so that the ai value is the average of all the readings from the poller since the last time the record processed. For example, if the poller is running at 100 Hz and the ai record SCAN field is "0.2 seconds" then 20 values will be averaged each time the record processes. If SCAN=I/O Intr then the device support will average the number of values specified in the SVAL field of the record. If SVAL<=1 then the record will processes on each callback, so there is no averaging.
(P)(R)Range	mbbo	asynInt32	ANALOG_IN_RANGE	Input range for this analog input channel. Choices are determined at run time based on the model in use.
(P)(R)Type	mbbo	asynInt32	ANALOG_IN_TYPE	Input type (e.g. "Volts", "TC deg", etc.) for this analog input channel. Choices are determined at run time based on the model in use.

The following is the medm screen for controlling the analog input records for the USB-1608GX-2AO. Note that the engineering units limits (EGUL and EGUF) do not have to be in volts, they can be in any units such as "percent", "degrees", etc.

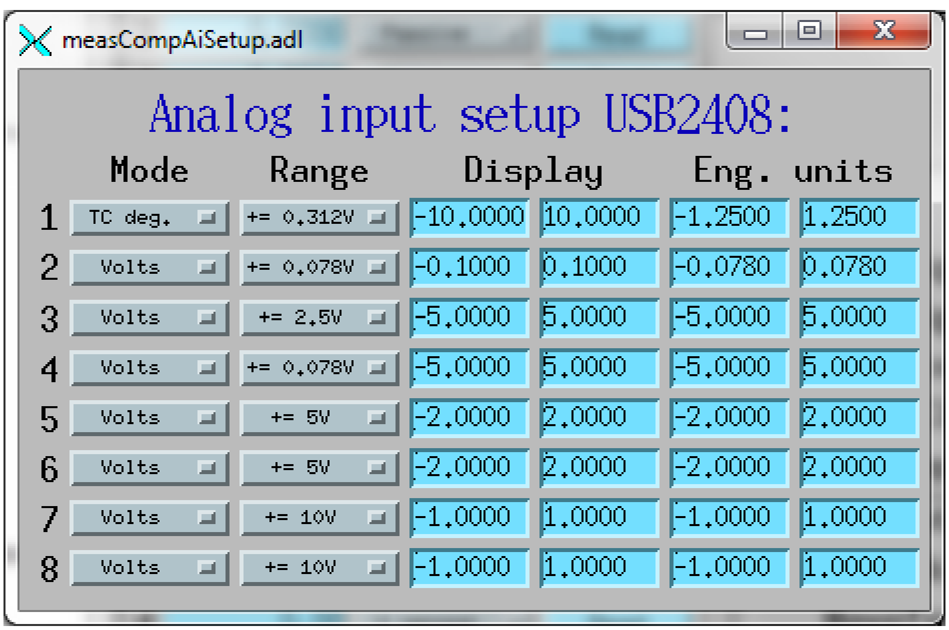

Fig. 24: **measCompAiSetup.adl**

These are the records defined in measCompAnalogOut.template. This database is loaded once for each analog output channel

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)	ai	asynInt32	ANALOG_OUT_VALUE	Analog output value. This is converted from engineering units to the 16-bit unsigned integer device units for the driver using the EGUL and EGUF fields.
(P)(R)Range	mbbo	asynInt32	ANALOG_OUT_RANGE	Output range for this analog output channel. Choices are determined at run time based on the model in use.
(P)(R)Return	ai	asynInt32	ANALOG_OUT_VALUE	Analog output value to return to at the end of a pulse. This is converted from engineering units to the 16-bit unsigned integer device units for the driver using the EGUL and EGUF fields.
(P)(R)Pulse	bo	N.A.	N.A.	Choices are “Normal” and “Pulse”. In Normal mode the Return record is ignored. In Pulse mode the $(P)($R) output is written to to hardware, followed immediately by writing the (P)(R)Return value.
(P)(R)TweakVal	ao	N.A.	N.A.	The amount by which to tweak the out when the Tweak record is processed.
(P)(R)TweakUp	calcout	N.A.	N.A.	Tweaks the output up by TweakVal.
(P)(R)TweakDown	calcout	N.A.	N.A.	Tweaks the output down by TweakVal.

The following is the medm screen for controlling the analog output records for the USB-1608GX-2AO. Note that the engineering units limits (EGUL and EGUF) do not have to be in volts, they can be in any units such as “percent”, “degrees”, etc. The drive limits can be more restrictive than the full +-10V output range of the analog outputs.

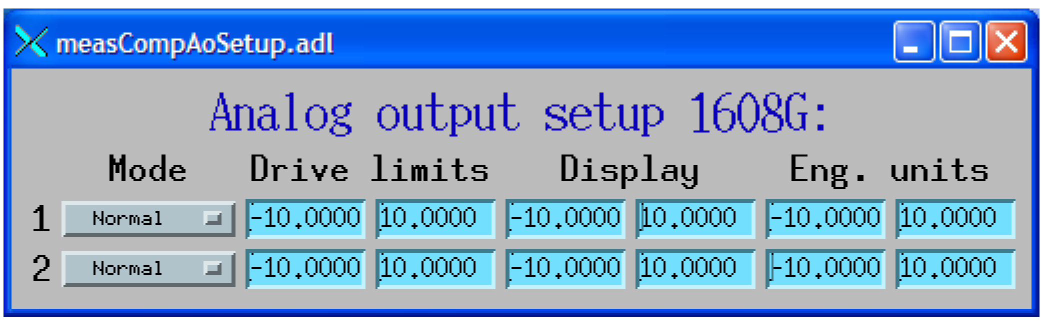

Fig. 25: **measCompAoSetup.adl**

Temperature Functions

These are the records defined in measCompTemperatureIn.template. This database is loaded once for each temperature input channel.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)	ai	asynFloat64	TEMPERATURE_IN_VALUE	Temperature input value. This field should be periodically scanned, since it is not currently polled in the driver, so I/O Intr scanning cannot be used.
(P)(R)Scale	mbbo	asynInt32	TEMPERATURE_SCALE	Temperature scale (units) for this temperature input channel. Choices are “Celsius” (0), “Fahrenheit” (1), “Kelvin” (2), “Volts” (4), and “Noscale” (5).
(P)(R)TCType	mbbo	asynInt32	THERMOCOUPLE_TYPE	Thermocouple type. Choices are “Type J” (1), “Type K” (2), “Type T” (3), “Type 4” (4), “Type R” (5), “Type S” (6), “Type B” (7), “Type N” (8)
(P)(R)Filter	mbbo	asynInt32	TEMPERATURE_FILTER	Temperature filter. Choices are “Filter” (0) and “No filter” (0x400)

The following is the main medm screen for configuring the analog/temperature inputs on the USB-2408-2AO.

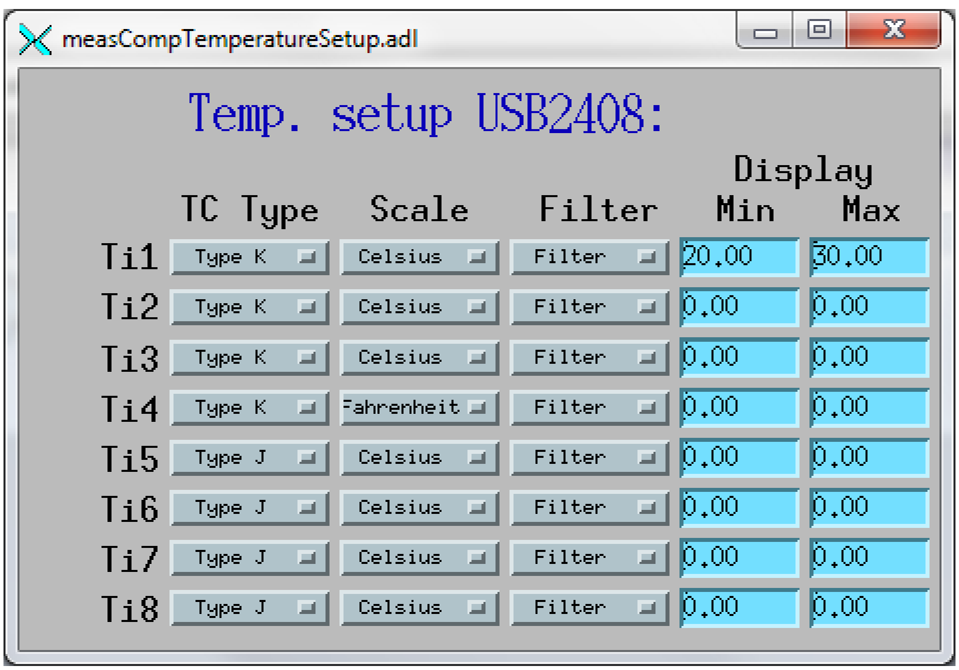

Fig. 26: **measCompTemperatureSetup.adl**

Digital I/O Functions

These are the records defined in the following files:

- measCompBinaryIn.template. This database is loaded once for each binary I/O bit.
- measCompLongIn.template. This database is loaded once for each binary I/O register.
- measCompBinaryOut.template. This database is loaded once for each binary I/O bit.
- measCompLongOut.template. This database is loaded once for each binary I/O register.
- measCompBinaryDir.template. This database is loaded once for each binary I/O bit.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)	bi	asynUInt32Digital	DIGITAL_INPUT	Digital input value. The MASK parameter in the INP link defines which bit is used. The binary inputs are polled by the driver poller thread, so these records should have SCAN="I/O Intr".
(P)(R)	longin	asynUInt32Digital	DIGITAL_INPUT	Digital input value as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used. The binary inputs are polled by the driver poller thread, so this record should have SCAN="I/O Intr".
(P)(R)	bo	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value. The MASK parameter in the INP link defines which bit is used.
(P)(R)_RBV	bi	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value readback. The MASK parameter in the INP link defines which bit is used.
(P)(R)	longout	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used.
(P)(R)_RBV	longin	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value readback as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used.
(P)(R)	bo	asynUInt32Digital	DIGITAL_DIRECTION	Direction of this I/O line, "In" (0) or "Out" (1). The MASK parameter in the INP link defines which bit is used.

Pulse Generator Functions

Note: These are called "timers" in Measurement Computing's documentation.

These are the records defined in measCompPulseGen.template. This database is loaded once for each pulse generator.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)Run	bo	asynUInt32	PULSE_RUN	"Run" (1) starts the pulse generator, "Stop" (0) stops the pulse generator. Note that ideally this record should go back to 0 when the pulse generator is done, if it is outputting a finite number of pulses (see Count record). But unfortunately the Measurement Computing library does not have a way to query the status of the timer to see if it is done, so this is not possible.
(P)(R)Period	ao	asynFloat64	PULSE_PERIOD	Pulse period, in seconds. The time between pulses can be defined either with the Period or with the Frequency; whenever one record is changed the other is updated with the new calculated value.
(P)(R)Frequency	ao	N.A.	N.A.	Pulse frequency, in seconds. The Frequency calculates a new value of the Period, and sends the period value to the driver.
(P)(R)Width	ao	asynFloat64	PULSE_WIDTH	Pulse width, in seconds. The allowed range is 15.625 ns to (Period-15.625 ns).
(P)(R)Delay	ao	asynFloat64	PULSE_DELAY	Initial pulse delay in seconds after Run is set to 1.
(P)(R)Count	longout	asynInt32	PULSE_COUNT	Number of pulses to output. If the Count is 0 then the pulse generator runs continuously until Run is set to 0.
(P)(R)IdleState	bo	asynInt32	PULSE_IDLE_STATE	The idle state of the pulse output line, "Low" (0) or "High" (1). This determines the polarity of the pulse, i.e. positive going or negative going.

Waveform Digitizer Functions

These records are defined in the following files: - measCompWaveformDig.template. This database is loaded once per module. - measCompWaveformDigN.template. This database is loaded for each digitizer input channel.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)NumPoints	longout	asynInt32	WAVEDIG_NUM_POINTS	Number of points to digitize. This cannot be more than the value of maxInputPoints that was specified in USB1608GConfig.
(P)(R)FirstChan	mbbo	asynInt32	WAVEDIG_FIRST_CHAN	First channel to digitize. “1” (0) to “8” (7). The database currently assumes differential inputs, so only 8 inputs are available, though this can easily be extended to 16.
(P)(R)NumChans	mbbo	asynInt32	WAVEDIG_NUM_CHANS	Number of channels to digitize. “1” (0) to “8” (7). The maximum valid number is 8-FirstChan+1. The database currently assumes differential inputs, so only 8 inputs are available, though this can easily be extended to 16.
(P)(R)TimeWF	waveform	asynFloat32Array	WAVEDIG_TIME_WF	Timebase waveform. These values are calculated when Dwell or NumPoints are changed. It is typically used as the X-axis in plots.
(P)(R)CurrentPoint	longin	asynInt32	WAVEDIG_CURRENT_POINT	The current point being collected. This does not always increment by 1 because the device can transfer data in blocks.
(P)(R)Dwell	ao	asynFloat64	WAVEDIG_DWELL	The time per point in seconds. The minimum time is 2 microseconds times NumChans.
(P)(R)TotalTime	ai	asynFloat64	WAVEDIG_TOTAL_TIME	The total time to digitize NumChans*NumPoints.
(P)(R)ExtTrigger	bo	asynInt32	WAVEDIG_EXT_TRIGGER	The trigger source, “Internal” (0) or “External” (1).
(P)(R)ExtClock	bo	asynInt32	WAVEDIG_EXT_CLOCK	The clock source, “Internal” (0) or “External” (1). If External is used then the Dwell record does not control the digitization rate, it is controlled by the external clock. However Dwell should be set to approximately the correct value if possible, because that controls what type of data transfers the device uses.
(P)(R)Continuous	bo	asynInt32	WAVEDIG_CONTINUOUS	Values are “One-shot” (0) or “Continuous” (1). This controls whether the device stops when acquisition is complete, or immediately begins another acquisition. Typically “One-shot” is used, because the driver is currently not double-buffered, so data could be overwritten before the driver has a chance to read the data. One exception is when using Retrigger=Enable and TriggerCount less than NumPoints. In that case each trigger will only collect TriggerCount samples, and one wants to use Continuous so that it collects the next TriggerCount samples on the next trigger input.
(P)(R)AutoRestart	bo	asynInt32	WAVEDIG_AUTO_RESTART	Values are “Disable” (0) and “Enable” (1). This controls whether the driver automatically starts another acquire when the previous one completes. This is different from Continuous mode described above, because this is a software restart that only happens after the driver has read the buffer from the previous acquisition.
(P)(R)Retrigger	bo	asynInt32	WAVEDIG_RETRIGGER	Values are “Disable” (0) and “Enable” (1). This controls whether the device rearms the trigger input after a trigger is received.
(P)(R)TriggerCount	longout	asynInt32	WAVEDIG_TRIGGER_COUNT	This controls how many samples are collected on each trigger input. 0 means collect NumPoint samples. If TriggerCount is less than NumPoints, Retrigger=Enable and Continuous=Enable then each time a trigger is received TriggerCount samples will be collected.
(P)(R)BurstMode	bo	asynInt32	WAVEDIG_BURST_MODE	Values are “Disable” (0) and “Enable” (1). This controls whether the device digitizes all NumChans channels as quickly as possible during each sample, or whether it digitizes successive channels at evenly spaced time intevals during the Dwell time. Enabling BurstMode means that all channels are digitized 2 microseconds apart. This can reduce the accuracy if the channels have very different voltages because of the settling time and slew rate limitations of the system.
(P)(R)Run	busy	asynInt32	WAVEDIG_RUN	Values are “Stop” (0) and “Run” (1). This starts and stops the waveform digitizer.
(P)(R)ReadWF	busy	asynInt32	WAVEDIG_READ_WF	Values are “Done” (0) and “Read” (1). This reads the waveform data from the device buffers into the waveform records. Note that the driver always reads device when acquisition stops, so for quick acquisitions this record can be Passive. To see partial data during

This is a plot of a digitized waveform captured of someone speaking into a microphone.

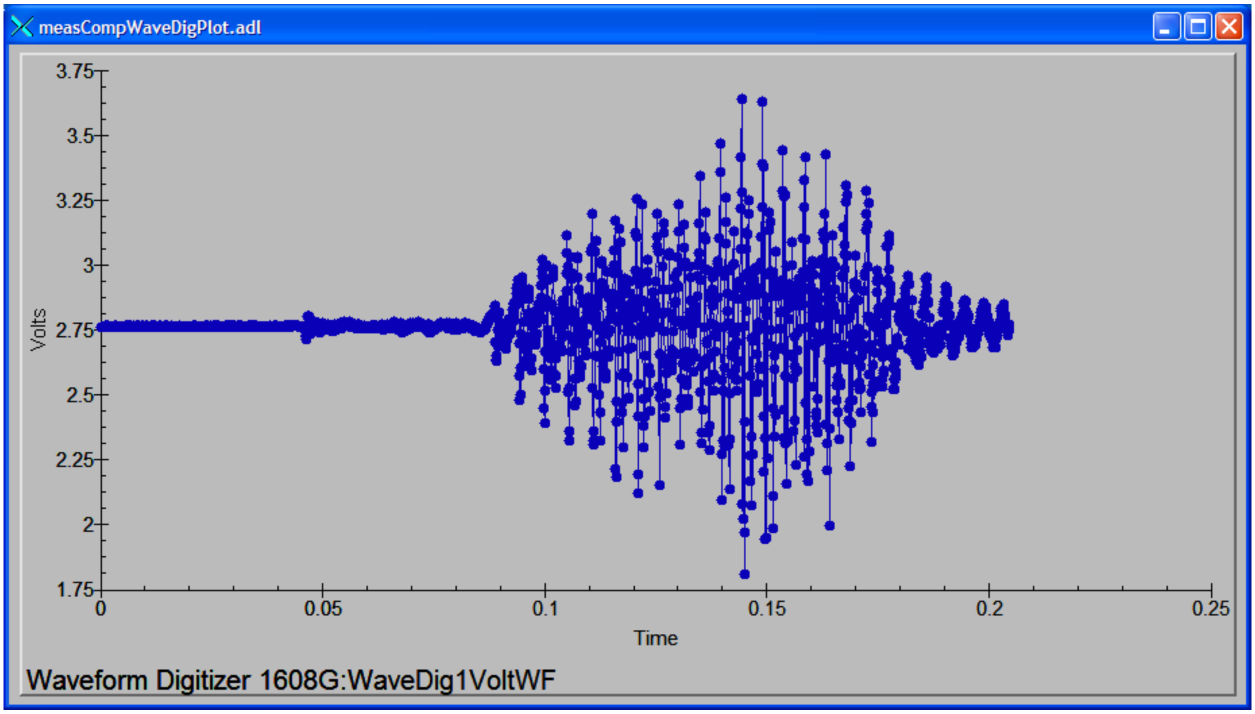

Fig. 27: **Waveform digitizer plot**

Waveform Generator Functions

These records are defined in the following files: - measCompWaveformGen.template. This database is loaded once per module. - measCompWaveformGenN.template. This database is loaded for each waveform generator output channel.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)NumPoints	longin	asynInt32	WAVEGEN_NUM_POINTS	Number of points output waveform. The value of this record is equal to UserNumPoints if user-defined waveforms are selected, or IntNumPoints if internal predefined waveforms are selected.
(P)(R)UserNumPoints	longout	asynInt32	WAVEGEN_USER_NUM_POINTS	Number of points in user-defined output waveforms. This cannot be more than the value of maxOutputPoints that was specified in USB1608GConfig.
(P)(R)IntNumPoints	longout	asynInt32	WAVEGEN_INT_NUM_POINTS	Number of points in internal predefined output waveforms. This cannot be more than the value of maxOutputPoints that was specified in USB1608GConfig.
(P)(R)UserTimeWF	waveform	asynFloat32Array	WAVEDIG_USER_TIME_WF	Timebase waveform for user-defined waveforms. These values are calculated when UserDwell or UserNumPoints are changed. It is typically used as the X-axis in plots.
(P)(R)IntTimeWF	waveform	asynFloat32Array	WAVEGEN_INT_TIME_WF	Timebase waveform for internal predefined waveforms. These values are calculated when IntDwell or IntNumPoints are changed. It is typically used as the X-axis in plots.
(P)(R)CurrentPoint	longin	asynInt32	WAVEGEN_CURRENT_POINT	The current point being output. This does not always increment by 1 because the device can transfer data in blocks.
(P)(R)Frequency	ai	asynFloat64	WAVEGEN_FREQUENCY	The output frequency (waveforms/second). The value of this record is equal to UserFrequency if user-defined waveforms are selected, or IntFrequency if internal predefined waveforms are selected.
(P)(R)Dwell	ai	asynFloat64	WAVEGEN_DWELL	The output dwell time or period (seconds/sample). The value of this record is equal to UserDwell if user-defined waveforms are selected, or IntDwell if internal predefined waveforms are selected.
(P)(R)UserDwell	ao	asynFloat64	WAVEGEN_USER_DWELL	The output dwell time or period (seconds/sample) for user-defined waveforms. This record is automatically changed if UserFrequency is modified.
(P)(R)IntDwell	ao	asynFloat64	WAVEGEN_INT_DWELL	The output dwell time or period (seconds/sample) for internal predefined waveforms. This record is automatically changed if IntFrequency is modified.
(P)(R)UserFrequency	ao	N.A.	N.A.	The output frequency (waveforms/second) for user-defined waveforms. This record computes UserDwell and writes to that record. This record is automatically changed if UserDwell is modified.
(P)(R)IntFrequency	ao	N.A.	N.A.	The output frequency (waveforms/second) for internal predefined waveforms. This record computes IntDwell and writes to that record. This record is automatically changed if IntDwell is modified.
(P)(R)TotalTime	ai	asynFloat64	WAVEGEN_TOTAL_TIME	The total time to output the waveforms. This is Dwell*NumPoints.
(P)(R)ExtTrigger	bo	asynInt32	WAVEGEN_EXT_TRIGGER	The trigger source, “Internal” (0) or “External” (1).
(P)(R)ExtClock	bo	asynInt32	WAVEGEN_EXT_CLOCK	The clock source, “Internal” (0) or “External” (1). If External is used then the Dwell record does not control the output rate, it is controlled by the external clock. However Dwell should be set to approximately the correct value if possible, because that controls what type of data transfers the device uses.
(P)(R)Continuous	bo	asynInt32	WAVEGEN_CONTINUOUS	Values are “One-shot” (0) or “Continuous” (1). This controls whether the device stops when the output waveform is complete, or immediately begins again at the start of the waveform.
(P)(R)Retrigger	bo	asynInt32	WAVEGEN_RETRIGGER	Values are “Disable” (0) and “Enable” (1). This controls whether the device rearms the trigger input after a trigger is received.
(P)(R)TriggerCount	longout	asynInt32	WAVEGEN_TRIGGER_COUNT	This controls how many values are output on each trigger input. 0 means output NumPoints samples. If TriggerCount is less than NumPoints, Retrigger=Enable and Continuous=Enable then each time a trigger is received TriggerCount samples will be output.
(P)(R)Run	busy	asynInt32	WAVEGEN_RUN	Values are “Stop” (0) and “Run” (1). This starts and stops the waveform generator.
(P)(R)UserWF	wave-	asyn-	WAVE-	This waveform record contains the user-defined waveform generator

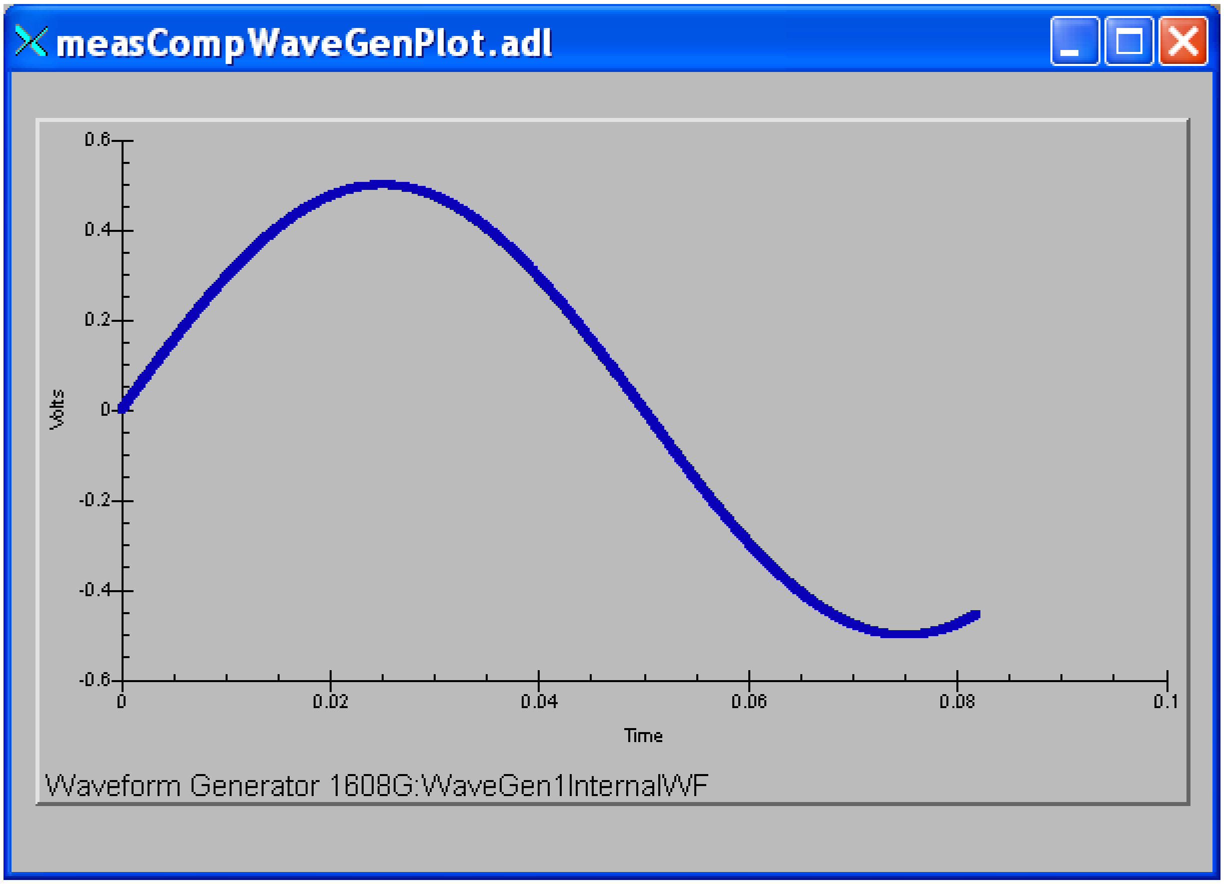

Fig. 28: **Plot of an internal predefined waveform (sin wave)**

Trigger Functions

These records are defined in measCompTrigger.template. This database is loaded once per module.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)Mode	mbbo	asynInt32	TRIGGER_MODE	The mode of the external trigger input. Choices are "Positive edge", "Negative edge", "High", and "Low".

2.2.5 Box for USB-CTR08, USB-3104, and USB-1808X

The following photo is a box we built to house the USB-CTR08, USB-3104, and USB-1808X and provide BNC I/O connections.

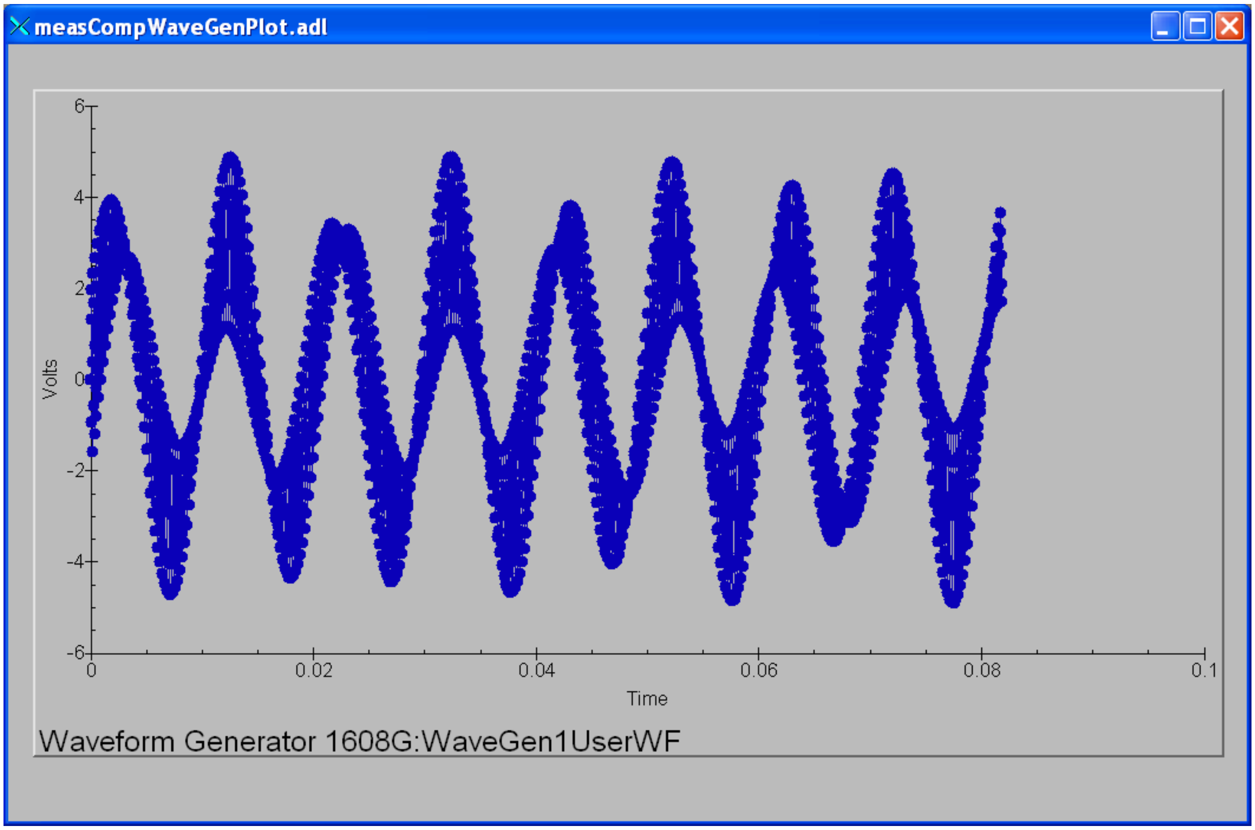

Fig. 29: **Plot of a user-defined waveform (sum of sin and cos waves)**

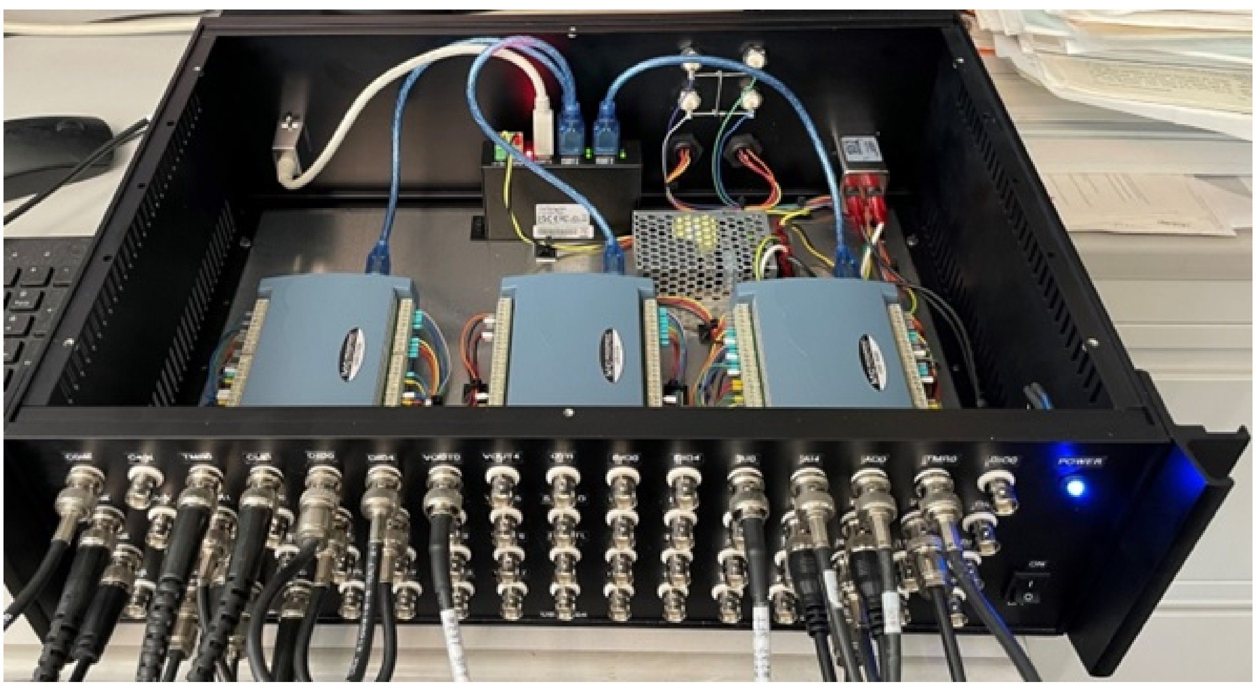

Fig. 30: **GSECARS designed box for USB-CTR08, USB-3104, and USB-1808X**

2.2.6 Box for USB-2408-2AO

The following photos show a box we built to house the USB-2408-2AO and provide I/O connections.

This is the top view.

Fig. 31: **Top view of USB-2408-2AO box**

These are the side views.

2.2.7 Performance measurements

The following summarizes a simple test of the precision and accuracy of the analog outputs and analog inputs of the USB-1608GX-2AO. The test configuration was with Analog Output 0 connected to Analog Input 0, and also to a Keithley 2700 digital multimeter. The Keithley is a 6.5 digit (22 bit) device, so it can be used to measure the accuracy of the USB-1608GX-2AO analog output, and provide the "true" value to measure the accuracy of the analog input. The 1608GX analog inputs records and the Keithley input had SCAN=0.1 second, so new readings were being made at 10Hz. The following IDL test program was used to drive the analog output from -10V to +10V in 0.1V steps. 10 readings were made of the 1608GX analog inputs, and one reading of the Keithley at each voltage step. These tests were done with the +-10V range of the analog outputs and analog inputs. Since these are 16-bit devices, one bit is 20V/65536 = 0.000305 volts.

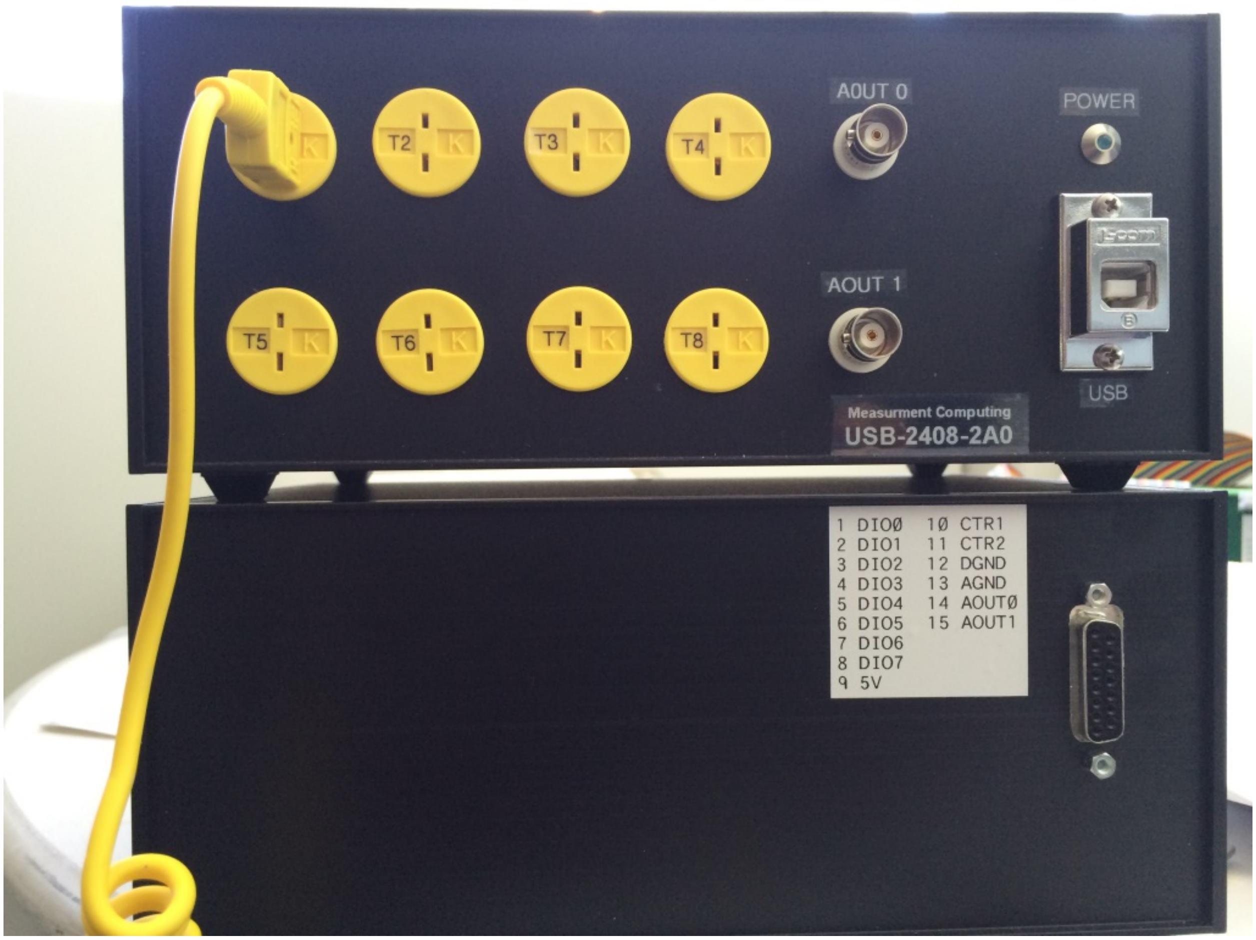

Fig. 32: **Side views of USB-2408-2AO box**

```
pro test_analog_performance_1608,  ao=ao, ai=ai, min_volts=min_volts, max_volts=max_
↪volts, $
                                   step_volts=step_volts, num_samples=num_samples,␣
↪delay=delay, $
                                   keithley=keithley, results

  if (n_elements(ao)          eq 0) then ao          = '1608G:Ao1'
  if (n_elements(ai)          eq 0) then ai          = '1608G:Ai1'
  if (n_elements(min_volts)   eq 0) then min_volts   = -10.0
  if (n_elements(max_volts)   eq 0) then max_volts   =  10.0
  if (n_elements(step_volts)  eq 0) then step_volts  = 0.1
  if (n_elements(num_samples) eq 0) then num_samples = 10
  if (n_elements(delay)       eq 0) then delay       = 0.1
  if (n_elements(keithley)    eq 0) then keithley    = '13LAB:DMM2Dmm_raw.VAL'

  output = min_volts
  samples = dblarr(num_samples)
  num_points = ((max_volts - min_volts) / step_volts + 0.5) + 1
  results = dblarr(4, num_points)
  for i=0, num_points-1 do begin
    output = min_volts + i*step_volts
    t = caput(ao, output)
    wait, 2*delay
    for j=0, num_samples-1 do begin
      wait, delay
      t = caget(ai, temp)
      samples[j] = temp
    endfor
    m = moment(samples)
    results[0,i] = output
    results[1,i] = m[0]
    results[2,i] = sqrt(m[1])
    t = caget(keithley, temp)
    results[3,i] = temp
    print, results[0,i], results[1,i], results[2,i], results[3,i]
  endfor
end
```

The following plot shows the difference of the nominal USB-1608GX-2AO analog output voltage from the Keithley 2700 reading. The mean error is 0.000312V, or just over 1 bit. The RMS error is 0.000203V, or less than 1 bit.

The following plot shows the difference of the mean of 10 readings of the 1608GX analog input voltage from the Keithley 2700 reading. The mean error is 0.000106V, less than 1 bit. The RMS error is 0.000259V, also less than 1 bit.

The following plot shows the standard deviation of 10 readings of the 1608GX analog input voltage. The values range from about 0.001V (~3 bits) at +-10V to less than 0.0003V (1 bit) between -2 and +2V.

The following table contains all of the results from the tests.

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
-10.00000	-9.99930	0.00084
-9.90000	-9.89978	0.00130
-9.80000	-9.79986	0.00126

Table 1 – continued from previous page

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
-9.70000	-9.69964	0.00134
-9.60000	-9.60018	0.00123
-9.50000	-9.50057	0.00099
-9.40000	-9.40020	0.00117
-9.30000	-9.30010	0.00080
-9.20000	-9.20046	0.00105
-9.10000	-9.09996	0.00118
-9.00000	-9.00035	0.00122
-8.90000	-8.90016	0.00079
-8.80000	-8.80061	0.00118
-8.70000	-8.69996	0.00138
-8.60000	-8.60044	0.00112
-8.50000	-8.50004	0.00098
-8.40000	-8.39973	0.00103
-8.30000	-8.29975	0.00132
-8.20000	-8.19965	0.00108
-8.10000	-8.09986	0.00115
-8.00000	-8.00040	0.00079
-7.90000	-7.90021	0.00088
-7.80000	-7.79950	0.00107
-7.70000	-7.69998	0.00099
-7.60000	-7.60018	0.00092
-7.50000	-7.49990	0.00080
-7.40000	-7.39986	0.00097
-7.30000	-7.29992	0.00101
-7.20000	-7.20006	0.00085
-7.10000	-7.09953	0.00100
-7.00000	-7.00060	0.00088
-6.90000	-6.89986	0.00097
-6.80000	-6.79988	0.00089
-6.70000	-6.69984	0.00107
-6.60000	-6.60017	0.00091
-6.50000	-6.49958	0.00088
-6.40000	-6.40043	0.00105
-6.30000	-6.30005	0.00088
-6.20000	-6.20008	0.00085
-6.10000	-6.10016	0.00076
-6.00000	-6.00052	0.00068
-5.90000	-5.89963	0.00077
-5.80000	-5.80050	0.00076
-5.70000	-5.70013	0.00066
-5.60000	-5.60006	0.00066
-5.50000	-5.50008	0.00082
-5.40000	-5.39989	0.00090
-5.30000	-5.29982	0.00081
-5.20000	-5.19997	0.00087
-5.10000	-5.10021	0.00048
-5.00000	-5.00011	0.00054
-4.90000	-4.89986	0.00071

Table 1 – continued from previous page

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
-4.80000	-4.79976	0.00070
-4.70000	-4.69960	0.00082
-4.60000	-4.60090	0.00054
-4.50000	-4.50050	0.00072
-4.40000	-4.40012	0.00076
-4.30000	-4.30039	0.00045
-4.20000	-4.20005	0.00066
-4.10000	-4.10010	0.00068
-4.00000	-4.00012	0.00062
-3.90000	-3.90018	0.00060
-3.80000	-3.80002	0.00059
-3.70000	-3.70019	0.00049
-3.60000	-3.60027	0.00056
-3.50000	-3.50042	0.00063
-3.40000	-3.40017	0.00048
-3.30000	-3.30043	0.00045
-3.20000	-3.20034	0.00064
-3.10000	-3.10027	0.00066
-3.00000	-3.00047	0.00043
-2.90000	-2.90025	0.00060
-2.80000	-2.80021	0.00044
-2.70000	-2.70033	0.00038
-2.60000	-2.60011	0.00058
-2.50000	-2.50001	0.00063
-2.40000	-2.40015	0.00051
-2.30000	-2.29960	0.00043
-2.20000	-2.20050	0.00041
-2.10000	-2.10040	0.00048
-2.00000	-2.00012	0.00054
-1.90000	-1.90018	0.00044
-1.80000	-1.80026	0.00044
-1.70000	-1.70025	0.00062
-1.60000	-1.60043	0.00041
-1.50000	-1.50054	0.00044
-1.40000	-1.40035	0.00037
-1.30000	-1.30001	0.00043
-1.20000	-1.20006	0.00035
-1.10000	-1.10024	0.00048
-1.00000	-1.00035	0.00052
-0.90000	-0.90056	0.00036
-0.80000	-0.80052	0.00050
-0.70000	-0.70011	0.00041
-0.60000	-0.60029	0.00036
-0.50000	-0.50056	0.00035
-0.40000	-0.40031	0.00032
-0.30000	-0.30042	0.00030
-0.20000	-0.20053	0.00048
-0.10000	-0.10037	0.00041
0.00000	0.00018	0.00030

Table 1 – continued from previous page

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
0.10000	0.09986	0.00046
0.20000	0.19995	0.00032
0.30000	0.30005	0.00035
0.40000	0.39979	0.00046
0.50000	0.49979	0.00032
0.60000	0.60008	0.00028
0.70000	0.69941	0.00041
0.80000	0.79979	0.00019
0.90000	0.89986	0.00037
1.00000	0.99956	0.00032
1.10000	1.09966	0.00051
1.20000	1.19982	0.00045
1.30000	1.29940	0.00041
1.40000	1.39959	0.00041
1.50000	1.49990	0.00035
1.60000	1.59969	0.00035
1.70000	1.69979	0.00052
1.80000	1.80029	0.00016
1.90000	1.89944	0.00050
2.00000	1.99966	0.00047
2.10000	2.09973	0.00045
2.20000	2.19980	0.00041
2.30000	2.29984	0.00044
2.40000	2.40006	0.00023
2.50000	2.49934	0.00032
2.60000	2.59937	0.00038
2.70000	2.69963	0.00054
2.80000	2.79994	0.00032
2.90000	2.90010	0.00033
3.00000	3.00026	0.00021
3.10000	3.09990	0.00027
3.20000	3.19976	0.00041
3.30000	3.30022	0.00022
3.40000	3.39977	0.00061
3.50000	3.49990	0.00045
3.60000	3.59991	0.00068
3.70000	3.69952	0.00039
3.80000	3.79974	0.00052
3.90000	3.89969	0.00043
4.00000	3.99994	0.00029
4.10000	4.09967	0.00042
4.20000	4.19974	0.00063
4.30000	4.29950	0.00058
4.40000	4.39973	0.00066
4.50000	4.50001	0.00055
4.60000	4.60005	0.00048
4.70000	4.70014	0.00043
4.80000	4.79982	0.00059
4.90000	4.89995	0.00069

Table 1 – continued from previous page

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
5.00000	4.99925	0.00059
5.10000	5.09960	0.00066
5.20000	5.19963	0.00087
5.30000	5.29952	0.00072
5.40000	5.39925	0.00084
5.50000	5.49926	0.00059
5.60000	5.59918	0.00065
5.70000	5.70004	0.00073
5.80000	5.79989	0.00081
5.90000	5.89972	0.00087
6.00000	6.00000	0.00076
6.10000	6.10001	0.00038
6.20000	6.19986	0.00047
6.30000	6.29947	0.00071
6.40000	6.39973	0.00077
6.50000	6.49986	0.00068
6.60000	6.60005	0.00091
6.70000	6.69947	0.00085
6.80000	6.79939	0.00065
6.90000	6.89924	0.00083
7.00000	6.99989	0.00074
7.10000	7.09972	0.00091
7.20000	7.20012	0.00074
7.30000	7.30004	0.00073
7.40000	7.39934	0.00061
7.50000	7.50002	0.00073
7.60000	7.60003	0.00074
7.70000	7.69967	0.00101
7.80000	7.79947	0.00089
7.90000	7.89972	0.00094
8.00000	8.00027	0.00083
8.10000	8.09934	0.00090
8.20000	8.19971	0.00095
8.30000	8.29963	0.00112
8.40000	8.39997	0.00073
8.50000	8.49903	0.00089
8.60000	8.59962	0.00080
8.70000	8.69950	0.00109
8.80000	8.79945	0.00084
8.90000	8.89973	0.00111
9.00000	8.99980	0.00083
9.10000	9.09993	0.00071
9.20000	9.19966	0.00098
9.30000	9.29918	0.00090
9.40000	9.39910	0.00097
9.50000	9.49987	0.00106
9.60000	9.59890	0.00102
9.70000	9.70004	0.00110
9.80000	9.79974	0.00105

Table 1 – continued from previous page

1608GX analog output (nominal)	1608GX analog input (mean of 10 readings)	Std. Dev. of 10 1608GX analog input re
9.90000	9.89935	0.00112
10.00000	9.99951	0.00058

Suggestions and Comments to:
Mark Rivers : (rivers@cars.uchicago.edu)

2.3 Driver for the USB-CTR08

author
Mark Rivers, University of Chicago

Contents

2.3.1 Introduction

This is an EPICS driver for the USB-CTR04 and USB-CTR08 counter/timer modules from MeasurementComputing.

The driver is written in C++, and consists of a class that inherits from asynPortDriver, which is part of the EPICS asyn module.

This module has the following features:

- Digital inputs/outputs
  - 8 signals, individually programmable as inputs or outputs
- Pulse generators. 4 pulse generators each with

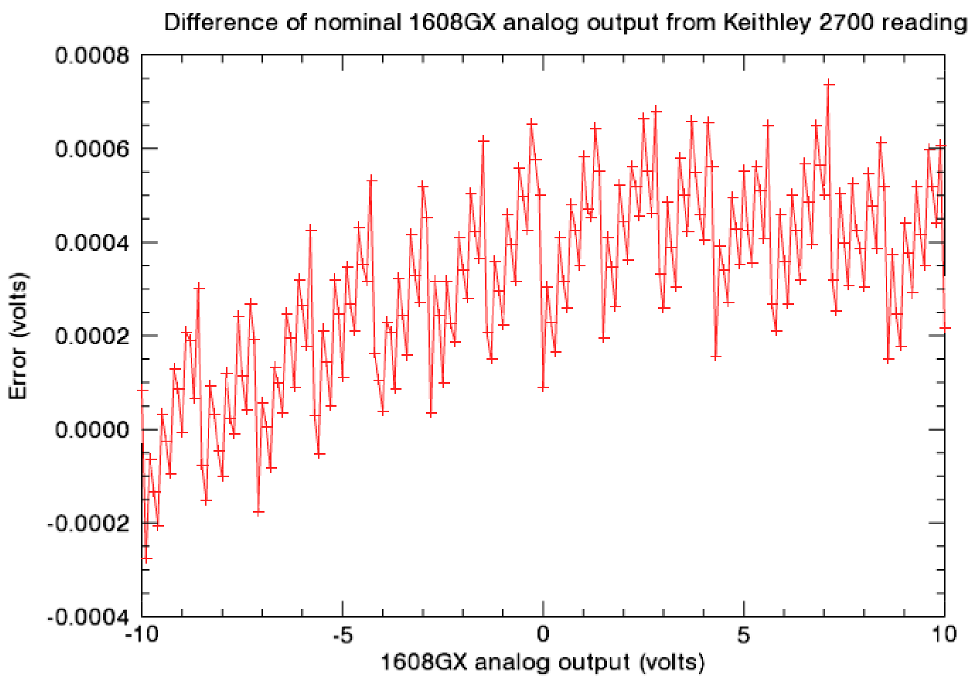

Fig. 33: **USB-1608GX-2AO analog output voltage error**

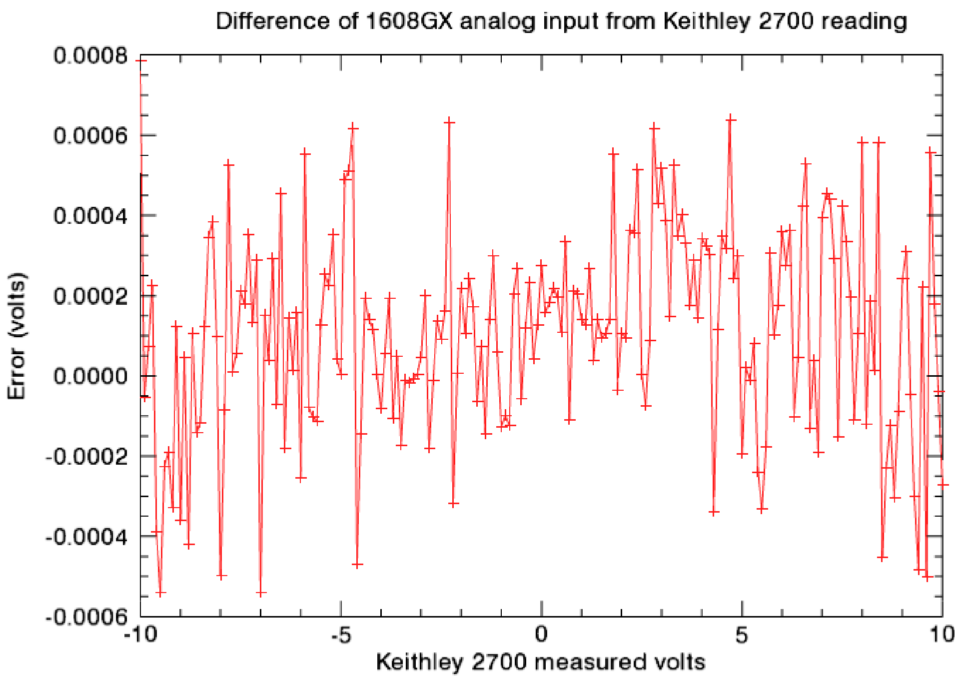

Fig. 34: **USB-1608GX-2AO analog input voltage error**

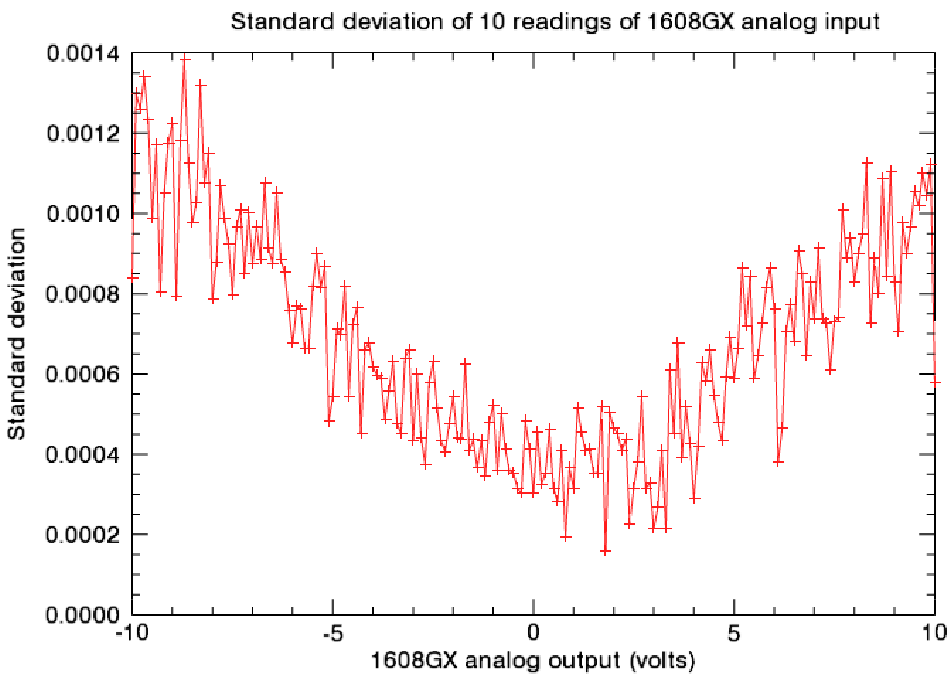

Fig. 35: **USB-1608GX-2AO analog input standard deviation**

Fig. 36: **Photo of USB-CTR08**

- 48MHz clock, 32-bit registers
- Programmable period, width, number of pulses, polarity
- Counters. 8 counters (USB-CTR08) or 4 counters (USB-CTR04)
  - 48 MHz maximum count rate
  - Support for EPICS scaler record (similar to Joerger VSC and SIS3820)
  - Support for Multi-Channel Scaler (MCS) mode, similar to SIS3820.

2.3.2 Configuration

The following lines are needed in the EPICS startup script for the USBCTR.

```
# This line is for Linux only
cbAddBoard("USB-CTR", "")

## Set the minimum sleep time to 1 ms
asynSetMinTimerPeriod(0.001)

## Configure port driver
# USBCTRConfig(portName,       # The name to give to this asyn port driver
#              boardNum,       # The number of this board assigned by the Measurement␣
↪Computing Instacal program
#              maxTimePoints)  # Maximum number of time points for MCS
USBCTRConfig("$(PORT)", 0, 2048, .01)

#asynSetTraceMask($(PORT), 0, TRACE_ERROR|TRACE_FLOW|TRACEIO_DRIVER)

dbLoadTemplate("USBCTR.substitutions")

# This loads the scaler record and supporting records
dbLoadRecords("$(SCALER)/db/scaler.db", "P=USBCTR:, S=scaler1, DTYP=Asyn Scaler,␣
↪OUT=@asyn(USBCTR), FREQ=10000000")

# This database provides the support for the MCS functions
dbLoadRecords("$(MEASCOMP)/measCompApp/Db/measCompMCS.template", "P=$(PREFIX), PORT=
↪$(PORT)")

# Load either MCA or waveform records below
# The number of records loaded must be the same as MAX_COUNTERS defined above

# Load the MCA records
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)1, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 0),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)2, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 1),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)3, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 2),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)4, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 3),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)5, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 4),  PREC=3, CHANS=$(MAX_POINTS)")
```

(continues on next page)

(continued from previous page)

```
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)6, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 5),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)7, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 6),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)8, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 7),  PREC=3, CHANS=$(MAX_POINTS)")
#dbLoadRecords("$(MCA)/mcaApp/Db/simple_mca.db", "P=$(PREFIX), M=$(RNAME)9, ␣
↪DTYP=asynMCA, INP=@asyn($(PORT) 8),  PREC=3, CHANS=$(MAX_POINTS)")

# This loads the waveform records
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)1, ␣
↪INP=@asyn($(PORT) 0),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)2, ␣
↪INP=@asyn($(PORT) 1),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)3, ␣
↪INP=@asyn($(PORT) 2),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)4, ␣
↪INP=@asyn($(PORT) 3),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)5, ␣
↪INP=@asyn($(PORT) 4),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)6, ␣
↪INP=@asyn($(PORT) 5),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)7, ␣
↪INP=@asyn($(PORT) 6),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)8, ␣
↪INP=@asyn($(PORT) 7),  CHANS=$(MAX_POINTS)")
dbLoadRecords("$(MCA)/mcaApp/Db/SIS38XX_waveform.template", "P=$(PREFIX), R=$(RNAME)9, ␣
↪INP=@asyn($(PORT) 8),  CHANS=$(MAX_POINTS)")

asynSetTraceIOMask($(PORT),0,2)
#asynSetTraceFile("$(PORT)",0,"$(MODEL).out")

< save_restore.cmd
save_restoreSet_status_prefix($(PREFIX))
dbLoadRecords("$(AUTOSAVE)/asApp/Db/save_restoreStatus.db", "P=$(PREFIX)")

iocInit

seq(USBCTR_SNL, "P=$(PREFIX), R=$(RNAME), NUM_COUNTERS=$(MAX_COUNTERS), FIELD=$(FIELD)")
create_monitor_set("auto_settings.req",30)
```

The measComp module comes with an example iocBoot/iocUSBCTR directory that contains and example startup script and example substitution files.

2.3.3 Databases

The following tables list the database template files that are used with the USB-CTR04/08.

Digital I/O Functions

These are the records defined in the following files:

- measCompBinaryIn.template. This database is loaded once for each binary I/O bit.
- measCompLongIn.template. This database is loaded once for each binary I/O register.
- measCompBinaryOut.template. This database is loaded once for each binary I/O bit.
- measCompLongOut.template. This database is loaded once for each binary I/O register.
- measCompBinaryDir.template. This database is loaded once for each binary I/O bit.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)	bi	asynUInt32Digital	DIGITAL_INPUT	Digital input value. The MASK parameter in the INP link defines which bit is used. The binary inputs are polled by the driver poller thread, so these records should have SCAN="I/O Intr".
(P)(R)	longin	asynUInt32Digital	DIGITAL_INPUT	Digital input value as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used. The binary inputs are polled by the driver poller thread, so this record should have SCAN="I/O Intr".
(P)(R)	bo	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value. The MASK parameter in the INP link defines which bit is used.
(P)(R)_RBV	bi	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value readback. The MASK parameter in the INP link defines which bit is used.
(P)(R)	longout	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used.
(P)(R)_RBV	longin	asynUInt32Digital	DIGITAL_OUTPUT	Digital output value readback as a word, rather than individual bits. The MASK parameter in the INP link defines which bits are used.
(P)(R)	bo	asynUInt32Digital	DIGITAL_DIRECTION	Direction of this I/O line, "In" (0) or "Out" (1). The MASK parameter in the INP link defines which bit is used.

Pulse Generator Functions

Note: These are called "timers" in Measurement Computing's documentation.

These are the records defined in measCompPulseGen.template. This database is loaded once for each pulse generator.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)Run	bo	asynUInt32	PULSE_RUN	"Run" (1) starts the pulse generator, "Stop" (0) stops the pulse generator. Note that ideally this record should go back to 0 when the pulse generator is done, if it is outputting a finite number of pulses (see Count record). But unfortunately the Measurement Computing library does not have a way to query the status of the timer to see if it is done, so this is not possible.
(P)(R)Period	ao	asynFloat64	PULSE_PERIOD	Pulse period, in seconds. The time between pulses can be defined either with the Period or with the Frequency; whenever one record is changed the other is updated with the new calculated value.
(P)(R)Frequency	ao	N.A.	N.A.	Pulse frequency, in seconds. The Frequency calculates a new value of the Period, and sends the period value to the driver.
(P)(R)Width	ao	asynFloat64	PULSE_WIDTH	Pulse width, in seconds. The allowed range is 15.625 ns to (Period-15.625 ns).
(P)(R)Delay	ao	asynFloat64	PULSE_DELAY	Initial pulse delay in seconds after Run is set to 1.
(P)(R)Count	longout	asynInt32	PULSE_COUNT	Number of pulses to output. If the Count is 0 then the pulse generator runs continuously until Run is set to 0.
(P)(R)IdleState	bo	asynInt32	PULSE_IDLE_STATE	The idle state of the pulse output line, "Low" (0) or "High" (1). This determines the polarity of the pulse, i.e. positive going or negative going.

Scaler Record Support

The USBCTR driver supports the EPICS scaler record via the devScalerAsyn.c device support originally from the synApps std module but which has been moved into the scaler module. It supports up to 8 channels. The following wiring connections must be made in order for counters 1-8 to be stopped by counter 0, as is normally desired.

- Counter 0 Output must be connected to the Gate input on Counters 1-7.

The .PR1 preset is performed in hardware via the Counter 0 Output and Counters 1-7 gates. Counters 1-7 can also be set as preset counters, and the scaler record will stop counting when any of these preset values (.PR2-.PR8) are exceeded. However, unlike the .PR1 preset, these presets are done in software in the driver polling routine. The device sends readings at 100 Hz, and whenever a preset is exceeded counting is stopped. Each of the counters will have counted for exactly the same amount of time, but the actual count time could be up to 0.01 seconds longer than the time when the preset was reached.

Counter 0 is normally used as the preset counter, and is connected to a fixed frequency source. Any of the on-board pulse generators can be used to provide this frequency source, for example. It is important to set the scaler record .FREQ field to be the value of the Frequency_RBV of the pulse generator (the actual frequency) and not the Frequency field (the requested frequency) since these can differ, particularly at frequencies >1 MHz.

Multi-Channel Scaler (MCS) Support

The USBCTR driver provides multi-channel scaler support very similar to the SIS3820 driver in the synApps mca module. The support has the following properties:

- The number of counters being used in MCS mode can be selected with the FirstCounter and LastCounter records. Each can range from 0 to 7; LastCounter must be greater than or equal to FirstCounter. The number of active counters can thus range from 1 to 8.
- The minimum dwell time, either with internal or external channel advance, is 250 ns times the number of active counters. For example if only 2 counters are being used, the clock input on Counter 0 and a signal on Counter 1, then the minimum dwell time is 500 ns. If all 8 counters are being used then the minimum dwell time is 2 microseconds.
- Either MCS or waveform records can be used to hold the time series data.
- There is no limitation on the length of the waveform or mca records, only the size of system RAM.
- An external channel advance signal can be used directly by connecting it to the External Clock Input (CLKI)on the USB-CTR module. The minimum dwell time (period) of this signal is described above.
- An external channel advance can be "prescaled" (frequency divided by N) by connecting it to a counter input. This counter is assigned to the PrescaleCounter record. The Counter Output of the PrescaleCounter must be connected to the External Clock Input on the USB-CTR module. I have asked Measurment Computing to consider adding a prescale register for the CLKI signal in a future firmware version, but I don't know if this will be done.
- To achieve the shortest dwell times the counter must be read in 16-bit mode rather than 32-bit mode. This is handled automatically by the driver. If the dwell time is less than 100 microseconds the counters are read in 16-bit mode, while for longer dwell times they are read in 32-bit mode. There is no possible loss of data when reading in 16-bit mode because at the maximum count rate of 48 MHz only 4800 counts can occur in 100 microseconds, which is much less than the 16-bit limit. NOTE: When using external channel advance the Dwell record should be set to the approximate time between external pulses. This will cause the correct 32-bit/16-bit switch to occur so that the minimum dwell time can be reached and so the counters don't overflow 16-bits for longer dwell times.

The following record are defined in measCompMCS.template. This database is loaded once per module.

EPICS record name	EPICS record type	asyn interface	drvInfo string	Description
(P)(R)SNL_Connected	bi	N.A.	N.A.	This record is 1 (“Connected”) if all PVs have connected in the USBCTR_SNL State Notation Language program.
(P)(R)EraseAll	bo	asynInt32	MCA_ERASE	Erases the MCS data, setting the arrays and the elapsed times to 0.
(P)(R)EraseStart	bo	asynInt32	MCA_ERASE	Erases the MCS data and then starts MCS acquisition by forward linking to StartAll.
(P)(R)StartAll	bo	asynInt32	MCA_START_ACQUIRE	Starts MCS acquisition.
(P)(R)Acquiring	busy	N.A.	N.A.	Busy record is 1 (“Acquiring”) when MCS is acquiring and 0 (“Done”) when done..
(P)(R)StopAll	bo	asynInt32	MCA_STOP_ACQUIRE	Stops MCS acquisition.
(P)(R)PresetReal	ao	asynFloat64	MCA_PRESET_REAL	Preset real time. If non-zero acquisition will stop after this time.
(P)(R)ElapsedReal	ai	asynFloat64	MCA_ELAPSED_REAL	Elapsed real time.
(P)(R)ReadAll	bo	N.A	N.A.	Forces a read of all of the array data. This is done by the SNL program.
(P)(R)NuseAll	longout	asynInt32	MCA_NUM_CHANNELS	The number of time points to acquire.
(P)(R)CurrentChannel	longin	asynInt32	MCS_CURRENT_POINT	The current time point in the acquisition.
(P)(R)Dwell	ao	asynFloat64	MCA_DWELL_TIME	The dwell time per time point in internal channel advance mode.
(P)(R)ChannelAdvance	bo	asynInt32	MCA_CH_ADV_SOURCE	The channel advance source. 0=”Internal” uses DWELL record, 1=”External” uses External Clock Input on USB-CTR module.
(P)(R)Prescale	bo	asynInt32	MCA_PRESCALE	The prescale factor for the external channel advance source. To use Prescale the external clock must be input to the counter channel selected by PrescaleCounter, and the output of the PrescaleCounter counter channel must be connected to the External Clock Input. Note that due to hardware limitations Prescale must be > 1. For no prescaling the external channel advance source must be connected directly to the External Clock Input.
(P)(R))MCSCounterNEnable (N=1-8)	bo	asynInt32	N.A.	Enable counter N in MCS mode. Choices are “No” (0) and “Yes” (1).
(P)(R))MCSDIOEnable	bo	asynInt32	N.A.	Enable collecting digital I/O word in MCS mode. Choices are “No” (0) and “Yes” (1).
(P)(R)PrescaleCounter	mbbo	asynInt32	MCS_PRESCALE_COUNTER	The counter channel to use for prescaling the external channel advance in MCS mode. 0=”CNTR0” … 7=”CNTR7”.
(P)(R)Point0Action	mbbo	asynInt32	MCS_POINT0_ACTION	Controls how the first time point in the MCS scan is handled. The USB-CTR always reads the current scaler counts as soon as MCS acquisition begins, rather than after the first channel advance occurs. This record selects one of the following 3 modes: • “Clear” (0) In this mode the scalers are cleared to 0 before they are read. This means that the counts in first time point for each counter will be 0. • “No clear” (1) In this mode the scalers are not cleared before they are read. This means that there will normally be a large number of counts in the first time point, since the counters will have been counting since they were last cleared. • “Skip” (2) In this mode the first time point will be skipped, i.e. not read into the mca or waveform records. The first time point will thus contain the counts after MCS acquisition was started until the first channel advance signal is received, either internal or external. This is probably the mode that will be most useful. However, it does require N+1 channel advance signals rather than N. This is handled by the driver for internal channel advance. But for external channel advance the user must ensure that N+1 pulses are sent. For example if NUseAll=2000 then 2001 pulses must be sent before acquisition will stop.

medm screens

The following is the main medm screen for controlling the USB-CTR04/08.

The following is the medm screen for the EPICS scaler record using the USB-CTR04/08.

The following is the medm screen for controlling the MCS mode of the USB-CTR04/08.

2.3.4 Wiring to BCDA BC-020 LEMO Breakout Panels

The following photos show the BCDA BC-020 LEMO breakout panels wired to the USB-CTR08. A BC-020 with a BC-087 daughter card (left) is used for the 8 counter signals, and a BC-020 with wire-wrapping (right) is used for digital I/O, timer output, clock I/O, etc. .

Wiring table

```
      Digital I/O and other signals using wire-wrap connections

50-pin ribbon      USB-1608GX      BC-020       EPICS Function
connector pin    screw terminal   connector
 1                 DIO0             J1          Digital I/O bit 0
 2                  GND             J1 shell    Ground
 3                 DIO1             J2          Digital I/O bit 1
 4                  GND             J2 shell    Ground
 5                 DIO2             J3          Digital I/O bit 2
 6                  GND             J3 shell    Ground
 7                 DIO3             J4          Digital I/O bit 3
 8                  GND             J4 shell    Ground
 9                 DIO4             J5          Digital I/O bit 4
10                  GND             J5 shell    Ground
11                 DIO5             J6          Digital I/O bit 5
12                  GND             J6 shell    Ground
13                 DIO6             J7          Digital I/O bit 6
14                  GND             J7 shell    Ground
15                 DIO7             J8          Digital I/O bit 7
16                  GND             J8 shell    Ground
17                 TMR0             J9          Pulse generator 0 output
18                  GND             J9 shell    Ground
19                 TMR1            J10          Pulse generator 1 output
20                  GND            J10 shell    Ground
21                 TMR2            J11          Pulse generator 2 output
22                  GND            J11 shell    Ground
23                 TMR3            J12          Pulse generator 3 output
24                  GND            J12 shell    Ground
25                 TRIG            J13          Trigger input for MCS
26                  GND            J13 shell    Ground
27                 CLKI            J14          External channel advance input
28                  GND            J14 shell    Ground
29                 CLK0            J15          Clock output
30                  GND            J15 shell    Ground
31                  +VO            J16          +5 volt output
32                  GND            J16 shell    Ground
```

(continues on next page)

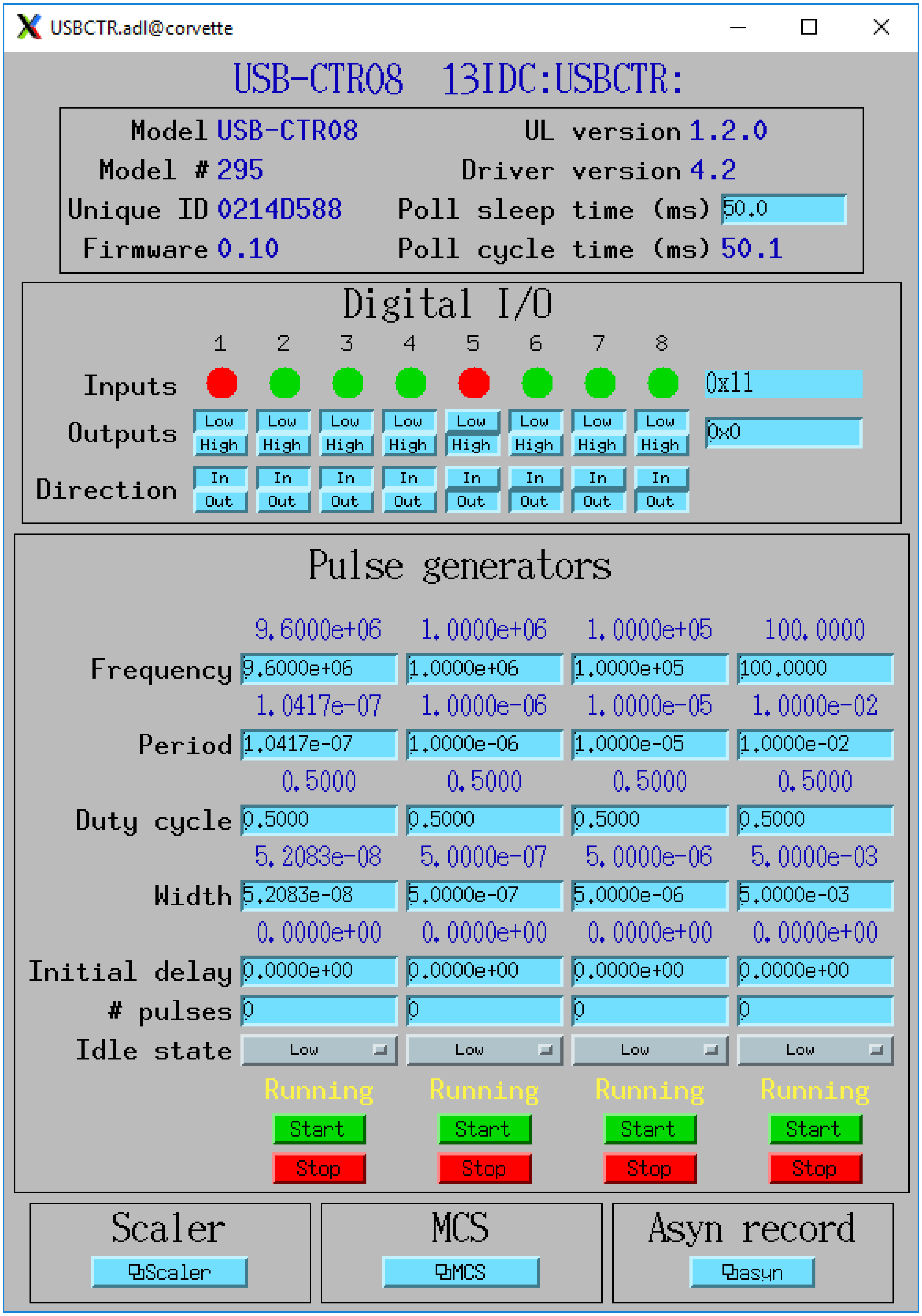

Fig. 37: **USBCTR.adl**

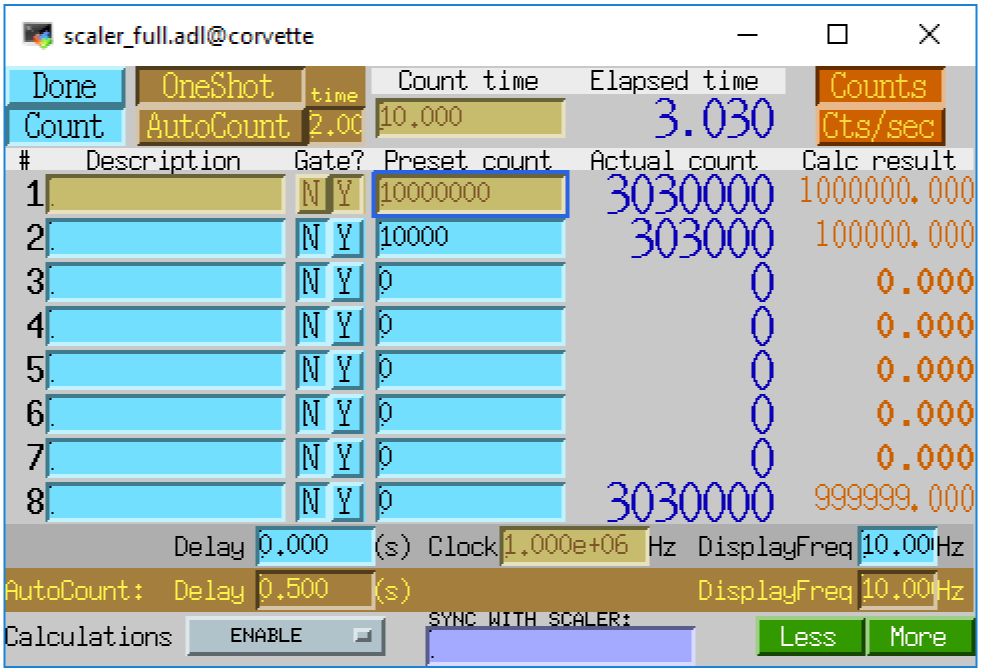

Fig. 38: **scaler_full.adl**

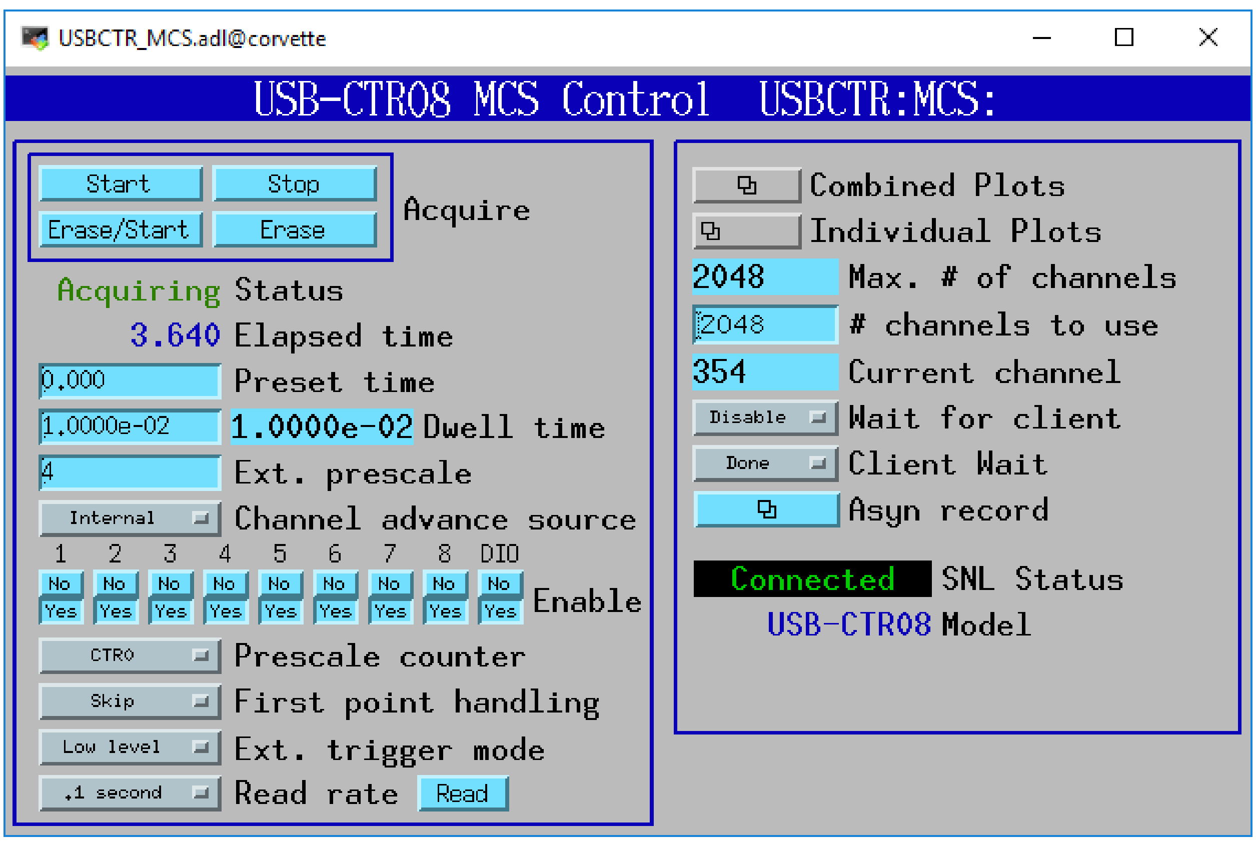

Fig. 39: **USBCTR_MCS.adl**

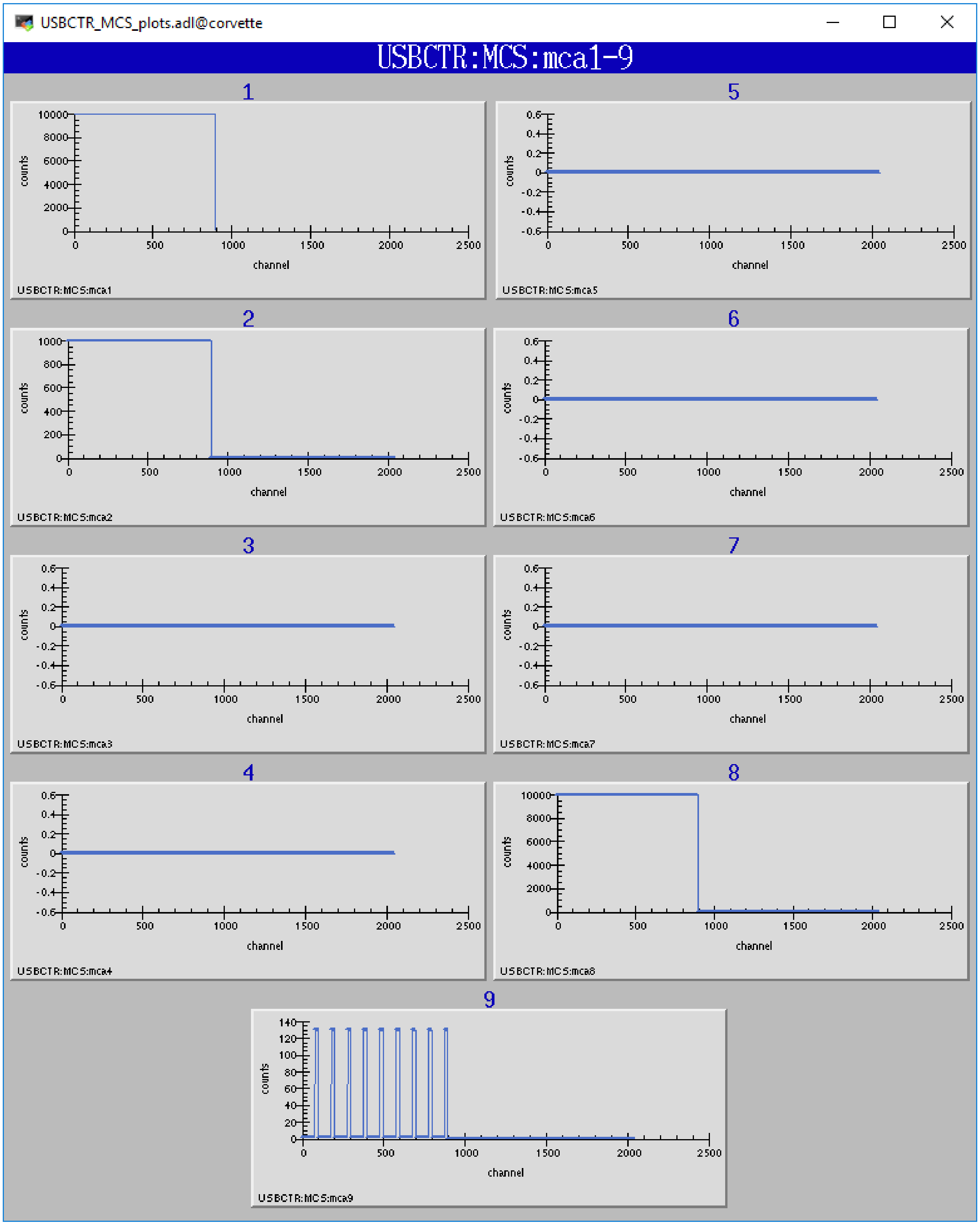

Fig. 40: **USBCTR_MCS_plots.adl**

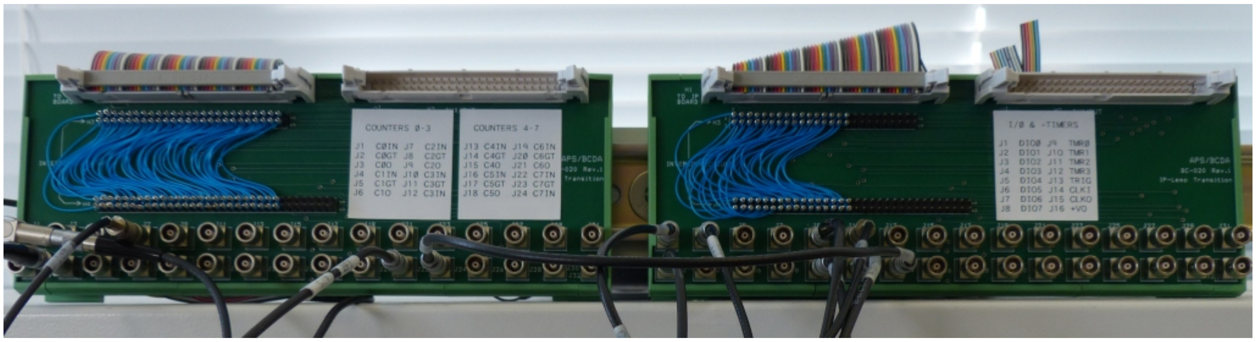

Fig. 41: **BC-020 LEMO breakout panels with USBCTR-08**

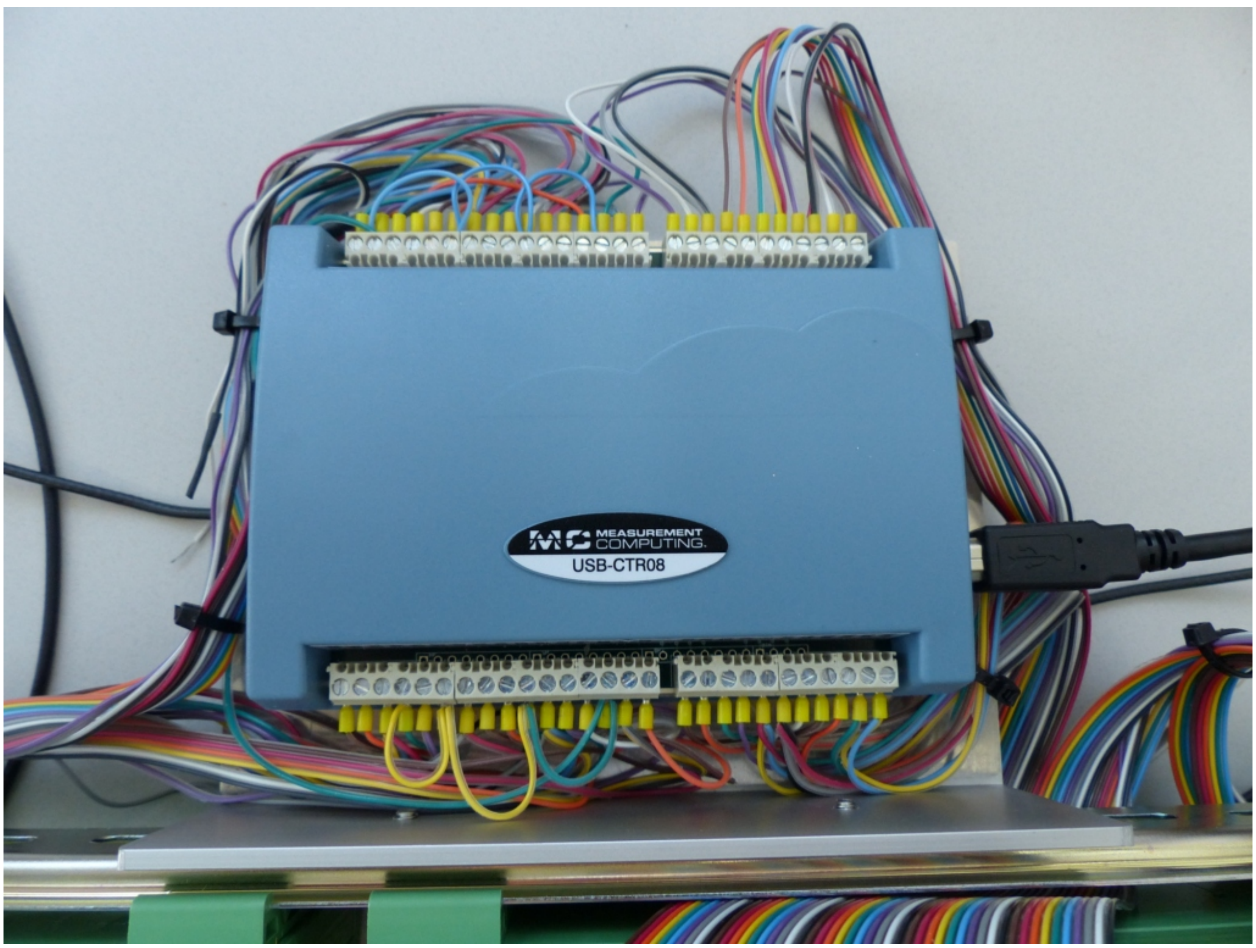

Fig. 42: **Top view of USBCTR-08 with BC-020 LEMO breakout panels**

(continued from previous page)

```
         Counter I/O using wire-wrap connections

50-pin ribbon      USB-CTR08      BC-020   EPICS Function
connector pin    screw terminal   connector
 1               C0IN               J1          Scaler 1 input
 2                GND               J1 shell    Ground
 3               C0GT               J2          Scaler 1 gate input
 4                GND               J2 shell    Ground
 5                C0O               J3          Scaler 1 output
 6                GND               J3 shell    Ground
 7               C1IN               J4          Scaler 2 input
 8                GND               J4 shell    Ground
 9               C1GT               J5          Scaler 2 gate input
10                GND               J5 shell    Ground
11                C1O               J6          Scaler 2 output
12                GND               J6 shell    Ground
13               C2IN               J7          Scaler 3 input
14                GND               J7 shell    Ground
15               C2GT               J8          Scaler 3 gate input
16                GND               J8 shell    Ground
17                C2O               J9          Scaler 3 output
18                GND               J9 shell    Ground
19               C3IN              J10          Scaler 4 input
20                GND              J10 shell    Ground
21               C3GT              J11          Scaler 4 gate input
22                GND              J11 shell    Ground
23                C4O              J12          Scaler 4 output
24                GND              J12 shell    Ground
25               C4IN              J13          Scaler 5 input
26                GND              J14 shell    Ground
27               C4GT              J14          Scaler 5 gate input
28                GND              J14 shell    Ground
29                C4O              J15          Scaler 5 output
30                GND              J15 shell    Ground
31               C5IN              J16          Scaler 6 input
32                GND              J16 shell    Ground
33               C5GT              J17          Scaler 6 gate input
34                GND              J17 shell    Ground
35                C5O              J18          Scaler 6 output
36                GND              J18 shell    Ground
37               C6IN              J19          Scaler 7 input
38                GND              J19 shell    Ground
39               C6GT              J20      Scaler 7 gate input
40                GND              J20 shell    Ground
41                C6O              J21          Scaler 7 output
42                GND              J21 shell    Ground
43               C7IN              J22          Scaler 8 input
44                GND              J22 shell    Ground
45               C7GT              J23          Scaler 8 gate input
46                GND              J23 shell    Ground
```

(continues on next page)

(continued from previous page)

```
47                  C7O              J24          Scaler 8 output
48                  GND              J24 shell    Ground

In addition to these connections counter 0 output (C0O) was connected to the gate
inputs of counters 1-7 (C1GT - C7GT) at the module screw terminals.
This is cheaper and simpler than using LEMO tees and short cables on the BC-020 module.
```

2.3.5 Performance measurements

The binary input bits are polled at 100 Hz, and the input records have SCAN=I/O Intr. There is thus a worse-case latency of 0.01 seconds in detecting a transition on these bits.

If the scaler record is run under the following conditions:

- Counter 0 Output connected to the Gate Input of Counters 1-7
- Pulse generator 0 frequency=32 MHz, connected to Counter 0 input
- Pulse generator 1 frequency=32 MHz, connected to Counter 1 input
- Pulse generator 2 frequency=32 MHz, connected to Counter 2 input
- Pulse generator 3 frequency=32 MHz, connected to Counter 3 input
- Scaler record .FREQ field = 3.2e7
- Scaler record preset time = 1.0 second
- Only scaler channel 1 is preset (.G1=Y, .G2-.G8=N)

After each count cycle .S1=32000000 counts exactly, .S2-.S4=32000000 += 1 count. There is thus no cross-talk with all channels running at 32 MHz, and the gate signals are working as designed.

If Pulse Generator 2 is changed to 3.2 MHz, .PR2 is set to 1600000, and .G2 is set to Y, then the scaler is stopped by channel 2 in the software polling routine. In this case it counts for exactly 0.50 seconds. However, if .PR2 is increased to 1600001 then it counts for 0.51 seconds. This corresponds to the worst case error due to the 100 Hz rate at which the scaler values are read. Note that all counters are active for exactly 0.51 seconds, so the counts all accurately reflect this count time. The count time is just slightly longer than requested due to the finite polling interval.

In MCS mode the measured minimum dwell time in both internal and external channel advance mode agrees with the datasheet, i.e. 250 ns * number of active counters. I was not able to measure any dead time between time bins in MCS mode. When sending exactly 8000000 pulses at 8 MHz to channel 0 with a 1 ms internal dwell time the total number of counts in the MCA record was 8000000. This means that no pulses were lost during the 1000 channel advances that happened during this time.

2.3.6 Restrictions

- The EPICS driver only uses the Totalize mode of the counters. With the scaler record it does a one-shot totalize, while in the MCS mode it totalizes into time-bins. The USB-CTR08 is also capable of running in 3 other modes.
    1. In Period mode it measures the time between the rising or falling edges of successive input pulses.
    2. In Pulse Width measurement mode it measures the time between the rising and falling edges of a each pulse.
    3. In Timing Mode it measures the time between an event on the counter input and another event on the counter gate.

None of these modes are currently supported by the EPICS driver, but they could be added in a future release.

- In Totalize mode each counter has many options in how it works: count up/down, gate clears counter, gate controls counter direction, preset counts where the output signal goes high/low, polarity of the output, etc. These options are not currently exposed in the EPICS driver.
- The EPICS driver only works in 32-bit counter depth mode. The USB-CTR08 can count with a 64-bit counter depth. asyn does not currently have support for 64-bit integer data types, so this cannot be supported.
- To work with the scaler record the counter 0 output must be wired to the gate inputs of counters 1-7 as discussed above.

Suggestions and Comments to:
Mark Rivers : (rivers@cars.uchicago.edu)