NEW EPICS DRIVERS FOR KECK TCS UPGRADE

J. Johnson, K. Tsubota, J. Mader W.M. Keck Observatory, Kamuela, HI, USA

Abstract

MOCOAAB05 describes how the telescope control system at the W.M. Keck Observatory is being upgraded. A key change is that the control system is moving from a VxWorks/VME platform to a RHEL/PC+COTS solution. The control system will continue to use EPICS but will move from R3.13.10 to R3.14. Upgrading from VxWorks/EPICS has resulted in the need for a number of new drivers. This paper assumes that the reader is familiar with EPICS, Device Support and the ASYN driver framework [1] and will focus on the domain specific portions of the drivers. The primary focus will be on the drivers for the following hardware: Heidenhain Encoder Interface Box (EIB) [2], National Instruments RIO [3] and Symmetricom BC635 [4]. Throughout this paper the Telescope Control System Upgrade will be referred to as TSCU.

OVERVIEW

Fig. 1 provides a high level overview of the drivers for TCSU. The ASYN framework [1] and StreamDevice are being reused from EPICS in addition to the PMAC [5] and EtherNet/IP [6] drivers. The remainder were developed at Keck. Each driver is documented with technical background material specific to the driver and also includes information such as to how to build, how to use and how to test (including db examples). All drivers use the asynPortDriver framework, typically support the asynInt32 and asynFloat64 interfaces and follow the standard EPICS *CONFIG.Defs* build convention.

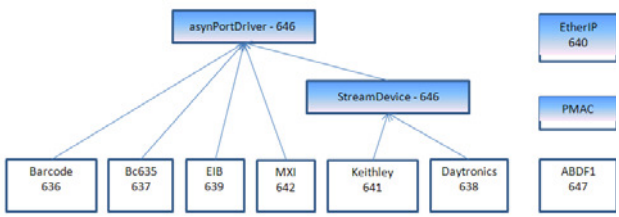

Figure 1: EPICS drivers for TCSU.

National Instruments provide a Linux solution for RIO communication, Symmetricom provide a Linux driver and API for the BC635 card and Heidenhain provide a Linux driver and API for their Encoder Interface Box (EIB). Keck provided an EPICS driver adaptation on top of these. All other drivers were ported directly from VxWorks to ASYN and two were redone to use StreamDevice.

HEINDENHAIN ENCODER DRIVER

For TCSU, we are using the Heidenhain EIB 749 encoder interface box for precise position measurement for the Azimuth and Elevation. A maximum of four Heidenhain encoders with sinusoidal incremental signals (1 VPP) or EnDat can be connected to the EIB 749. The EIB subdivides the periods of the incremental signals 4096-fold for measured-value generation reduced by adjusting the sinusoidal incremental signals. Internal or external triggers can be used for axis-specific storage of the measured values.

A standard Ethernet interface using TCP or UDP communication is available for data output. The method of transmission can be set via the operating mode. Heidenhain provide a Linux driver and user API which is well documented. The EIB 749 supports the following operating modes: Polling, Soft Real-Time, Streaming and Recording

All modes except for polling are associated with one or more triggers. A trigger can be external via hardware or internal using the EIB clock. The driver uses Soft Real-Time where the position data is transported with UDP packets from the EIB 749 to the client. This occurs parallel to the TCP communication via the standard Ethernet interface. With each trigger event, a data packet is sent to the client automatically and stored locally in a FIFO managed by the Linux driver.

From here, the application can read out the data within a program loop or register a callback function to be executed when data is available. The driver uses the callback method which allows it to be woken each time there are N or more items in the FIFO. This allows for easy oversampling and optional filtering. I/O INTR and Scan are supported by the driver.

The driver reports

- The unit hostname, firmware and ip address
- General configuration such as nominal increments, system line count etc.
- Trigger counts
- Per head information that includes: Head Status, the EIB sampling timestamp, Head Signal Strength, the incremental position, Absolute Reference Information (A, B & C), if there is an error, if the head is referenced

Commands include: Reference 1 or more heads, Reset the EIB and Reconnect to the EIB.

Signal strength is returned as 4 bytes from the EIB, 2 bytes for the A signal and 2 for the B signal. These are adjusted by the driver to the "zero-value of 0x800" of the signal, so:

A1 = 12-bit AD converter value - 0x800
B1 = 12-bit AD converter value - 0x800

ISBN 978-3-95450-139-7

The full read head 1 Vpp signal strength can then be produced using

SQRT (A1*A1+B1*B1) / ADC Counts

The driver must be configured from the IOC shell before it can be used. This includes providing its port name, the hostname of the EIB, the trigger to use and the system information on tape length, nominal increments etc.

Fig. 2 shows an associated test tool we developed that allows for detailed setup and testing of the read heads. This tool does not use the EPICS driver but shares much of the code base. Sinusoidal, position and reference data is being captured at 10 kHz and displayed at a 10 Hz update rate.

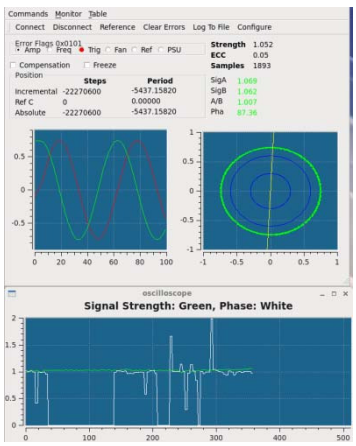

Figure 2: QT based EIB oscilloscope.

NATIONAL INSTRUMENTS RIO

This section describes the driver-level interface between EPICS and a National Instruments MXI/Ethernet Reconfigurable I/O (RIO) chassis and the associated IO. In TCSU the Axes Control and Rotator subsystems will utilize RIO and C Series IO for general purpose IO.

Figure 3: MXI RIO chassis with C Series I/O.

The RIO systems consist of a reconfigurable chassis housing the user-programmable FPGA, hot-swappable I/O modules, and graphical LabVIEW software for FPGA programming. The RIO core has an individual connection to each I/O module and is programmed with easy-to-use elemental I/O functions in the NI LabVIEW FPGA Module to read or write signal information from each module.

Communication from a Linux box to a MXI/Ethernet RIO is accomplished through the NI-RIO driver and the associated FPGA to C API. This low level driver is capable of communicating with the FPGA through DMA FIFOs or named variable access (indicators or controls). FPGA development is performed on a Windows machine using LVFPGA. The FPGA bitfile is compiled on the windows machine and a deployment-specific header file is created from it for use with the FPGA to C API.

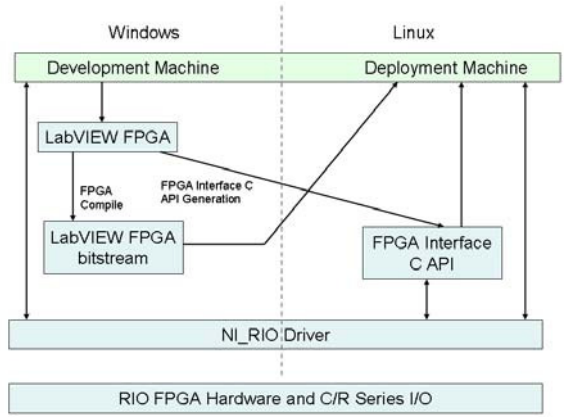

Figure 4: FPGA development and deployment.

In using the driver the basic steps to interface with the FPGA system are as follows:

1. On Windows, develop LabVIEW FPGA VI, compile bitfile, and generate C API.
2. Deploy bitfile and associated header file to Linux subsystem host
3. Preprocessed FPGA header file
4. Configure the driver to use the bitfile and mapping
5. Start IOC

In order to make the driver generic, the header generated from the bitfile cannot be used directly. The relevant information to address FPGA indicators and controls are present in the header file but are only available as enumerations. In order to allow the end user of the driver to express the parameter as part of the EPICS record definition the driver needs to support some type of simple text to address mapping. This is generated through a preprocessing step.

TCSU provides a script to take the FPGA output and create a simple mapping file that can be used by the driver. The generated file has the format

- FPGA Bitfile name
- Bitfile Signature
- Indicator/Control , DateType , Name, Address

```
e.g.
Indicator I16   A10C1        0x811E
Control   Bool  AOStop       0x8112
```

The asynFloat64 interface is used for AI and AO records. From the EPICS side read and write parameters

ISBN 978-3-95450-139-7

are expressed in floating point and the driver will convert these to fixed point numbers as needed by the RIO. The driver does not support INTR I/O as there is not a specific need for TCSU. In addition ondata change updates are not directly supported by the FPGA API making this more difficult to implement.

Natively all data transfers within the FPGA and between the FPGA and controller are 32 bit transfers. At the API level data transfers can occur as signed/unsigned 8, 16, 32 and 64 bit transfers. The FPGA does not support floating point but does utilize fixed point numbers.

All read or write processing has the same basic data flow:

- Use the ASYN base class to get the parameter name
- Use the mapping to get the correct FPGA address
- Use the FPGA session id, the FPGA item address and provided value to perform a read or write

Analogue data is more complicated than Boolean/integer data because of the use of Fixed Point Data Types and the fact that the word sizes can change depending on the IO module type. The EPICS driver takes care of all this providing automatic and correct fixed to floating point conversions as needed in both direction.

The entry point for the driver, which needs to be called from the st.cmd file before IocInit is *mxiRioAsynPortDriverConfigure (const char *portName, const char* filename).* It takes two parameters, the asyn port name to use and the fully qualified path to the mapping file.

SYMMETRICOM BC635

The current observatory timing solution consists of a pair of Symmetricom Network Servers that feed an IRIG-B signal to K1 and K2 through a set of IRIG distribution amplifiers. TCSU will continue to use this architecture and will utilize the Symmetricom bc635PCIe timing cards to connect the TCSU subsystem nodes to the timing network. The timing module will provide precise time and frequency functions to the subsystem controllers and associated peripheral data acquisition systems. The Symmetricom Linux SDK is used to create the EPICS driver.

TCSU has a logical subsystem called TIM that is comprised of:

1. EPICS driver and device support for the Symmetricom bc635PCIe board
2. EPICS general time support
3. Local NTP server based on BC635
4. Time conversion utilities
5. An IERS interface

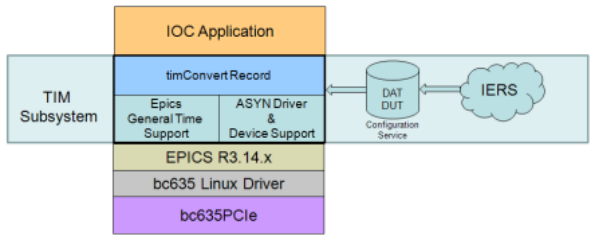

Figure 5: TIM subsystem.

The Linux and EPICS driver with general time support will support the following features:

- Providing an accurate and synchronized time source
- Accurate EPICS processing timestamps.
- Reading the current time on request.
- Triggering I/O interrupt and time report on an external event
- Supporting I/O interrupt at a specified time of day.
- Allowing accurate periodic interrupts to drive the system clock and thus initiate database processing at precise intervals.
- Generating an external periodic signal

The time conversion utilities will support the following:

- Accept time in UTC or GPS
- Convert time to TAI, GPS, UTC, UT (UT1), TT, TDB, GMST and LAST

The IERS interface will provide access to the various correction factors needed to convert between different timescales.

The bc635 driver supports the Symmetricom 635 PCIe timing board. The driver provides the core functionality to read and configure the timing and event related properties of the board. Clients can access the current global time and event time on demand, as well as utilize I/O interrupts to trigger software event processing. The driver supports many but not all of the bc635 capabilities but fully covers TCSU needs. I/O INTR and standard demand reads are supported. Items that can be read include: the current interrupts mask settings, the current operational mode, the current year, the current global time, event time, board/signal status. When reading the time (global or event) the driver will convert the UT timestamp returned by the board into the local UTC time in seconds.

When a command is written to the driver the associated command methods typically just invoke the Symmetricom library functions to set or configure the timing board. Event interrupts are executed asynchronously as part of a callback by the timing board library. The processing routine uses the interrupt source to determine where to fetch the current event time. This can be any one of the three event sources (*EVENT* 1, 2, or 3). Event 1 is the only source capable of being triggered off the DDS periodic pulse heartbeat. Event 2 and 3 are hardware driven interrupts. The event time is converted to the compatible UTC time and the driver member is

ISBN 978-3-95450-139-7

updated. I/O INTR records will automatically process if the corresponding event member is updated. The driver is configured as part of the IOC start-up. The following C function defines the entry point to the driver library:

```
bc635AsynPortDriverConfigure(const
char portName)
```

General Time, Reference Clocks and NTP

A typical computer has two clocks; a battery-backed clock that is always running (the ``hardware", “Real Time Clock (RTC)”, ``BIOS", or ``CMOS" clock), and another that is maintained by the operating system currently running on the computer (the ``system" clock). The hardware clock is generally only used to set the system clock when the operating system boots, and then from that point until a reboot or the system is powered off the system clock is the one used to keep track of time.

Optionally a system may utilize an additional high precision reference clock, typically this is an accurate clock utilizing IEEE 1588, IRIG or GPS for synchronization and available over the network or locally. Standard behavior of most operating systems is as follows (and this will be used by TCSU):

- Set the system clock from the hardware clock on boot
- Keep accurate time of the system clock with an NTP daemon
- Allow NTP to keep the hardware clock updated

For TCSU the system and hardware clocks will be set from the bc635PCIe card via the NTP protocol. TCSU provides an updated NTP daemon that uses the bc635 as a reference clock. Within NTP the use of a bc635 card has been standardized as Type 16 Bancomm GPS/IRIG Receiver (`GPS_BANCOMM`).

EPICS General Time Support

Record timestamps in EPICS are obtained through the generalTime framework which provides a mechanism for several time providers to be present within the system. There are two types: one provider for the current time and the other is for providing Time Event times. The bc635 driver is capable of registering as both a general time and event provider. The initialization occurs as a result of the iocsh *BC635Time_Init* call, which takes the provider priority as a parameter.

Time Conversion Utilities

Time conversion is provided via a custom record and allows conversion from UTC and GPS to TAI, GPS, UTC, UT (UT1), TT, TDB, GMST and LAST.

IERS Interface

The International Earth Rotation and Reference Systems Service (IERS) serves the astronomical, geodetic and geophysical communities by providing data and standards related to Earth rotation and reference frames. The IERS Rapid Service/Prediction Center is the product center of the IERS and is responsible for providing Earth orientation parameters (EOP) on a rapid turnaround basis.

TCSU provides an interface to the IERS Rapid Service/Prediction Center. It extracts and locally updates the IERS parameters that correct for polar motion, UT1 and occasionally leap seconds. As all of these affect the pointing of the telescope, the design here automatically handles the accessing of these parameters. To prevent the TCSU subsystems from having to access the IERS sites during operations, the IERS interface writes the data to the configuration service database (see TUPPC032). Once downloaded, the parameters can then be accessed when required without the need to make a connection to the IERS service. The latest downloaded values will then be available to the controller as soon as it is started.

OTHER DRIVERS

Additional drivers were developed using ASYN to support Accusort 20 barcode readers, the Daytronic System 10 data acquisition and Keithley M1000 series I/O. The latter two were migrated from drvAscii to use Streams. PMAC IP and the associated Motor record and the EtherNet/IP drivers are been reused directly from the EPICS community.

ACKNOWLEDGEMENTS

The W. M. Keck Observatory is operated as a scientific partnership among the California Institute of Technology, the University of California, and the National Aeronautics and Space Administration. The Observatory was made possible by the generous financial support of the W. M. Keck Foundation.

REFERENCES

[1] ASYN EPICS driver framework website: http://www.aps.anl.gov/epics/modules/soft/asyn
[2] Heidenhain website: http://www.heidenhain.com
[3] National Instruments (RIO subsection) website: http://www.ni.com/compactrio/
[4] Symmetricom (BC635 subsection) website: http://www.symmetricom.com/products/bus-level-timing/pci-express/bc635PCIe-IRIG
[5] SYNAPPS TPMAC (PMAC IP) website: http://www.gmca.anl.gov/TPMAC2/index.html
[6] K.U. Kasemir, L.R. Dalesio, “INTERFACING THE CONTROLLOGIX PLC OVER ETHERNET/IP”, THAP020, ICALEPS 2001

ISBN 978-3-95450-139-7