A Cost-Effective Way to Operate Instrumentation Using the Motorola MVME162 Industry Pack Bus and HiDEOS

J.B.Kowalkowski
Advanced Photon Source
Argonne National Laboratory

ABSTRACT

The HiDEOS software package has been implemented at the Advanced Photon Source to run the Motorola MVME162 embedded controller together with a variety of industry pack modules. The HiDEOS-based MVME162 controller acts as an intelligent second slave I/O processor in a VME crate. The implementation consists of vxWorks running the EPICS control system package with HiDEOS running the MVME162 in stand-alone mode with no additional operating system. Use of Industry Pack modules provides an inexpensive way to control a large number of serial ports, add a GPIB controller, or add a large number of analog and digital I/O modules at a reasonable cost. HiDEOS only utilizes software available in the public domain, allowing users to operate the MVME162 and supported Industry Pack hardware at no additional software cost. The HiDEOS package has been interfaced and integrated into the EPICS control system toolkit and handles the backplane communication with tasks under vxWorks and adds an object-oriented model to EPICS device support. Being object oriented, HiDEOS contains Industry Pack control classes that make it easy to add support for new Industry Pack modules.

INTRODUCTION

The software package HiDEOS [1] is currently being used at the Advanced Photon Source for GPIB, digital/analog input/output and serial communication to instruments. A library was added to create a standard interface to the EPICS control system toolkit [2]. The primary target is the Industry Pack (IP) Bus available on the Motorola MVME162 because of the low-cost IP modules available to the user. HiDEOS was chosen as a development platform because of its ability to hide complex issues, such as the IP Bus Controller and interprocess communications from the device driver creator, and to distribute processing among several processors.

One of the objectives of this development effort was to create generic support routines, accessible through the higher-level EPICS control system, that perform transactions with an instrument on a serial or GPIB link. With this set of tools, users can add new instruments and control them through EPICS without writing any embedded application programs. Existing scripting language interfaces to EPICS can interact with the existing embedded system.

A VME-embedded system main processor can be under substantial load in a large accelerator application. Adding a number of serial-linked instruments to the system can create a large additional load on the main CPU, especially if the instrument requires a complex protocol. A second goal of this development effort is to allow additional general-purpose, low-cost processors to perform the actual transactions to the GPIB and serial links. By allowing this type of system to be assembled, the main processor load, which is implementing most of the actual control algorithms, can be reduced.

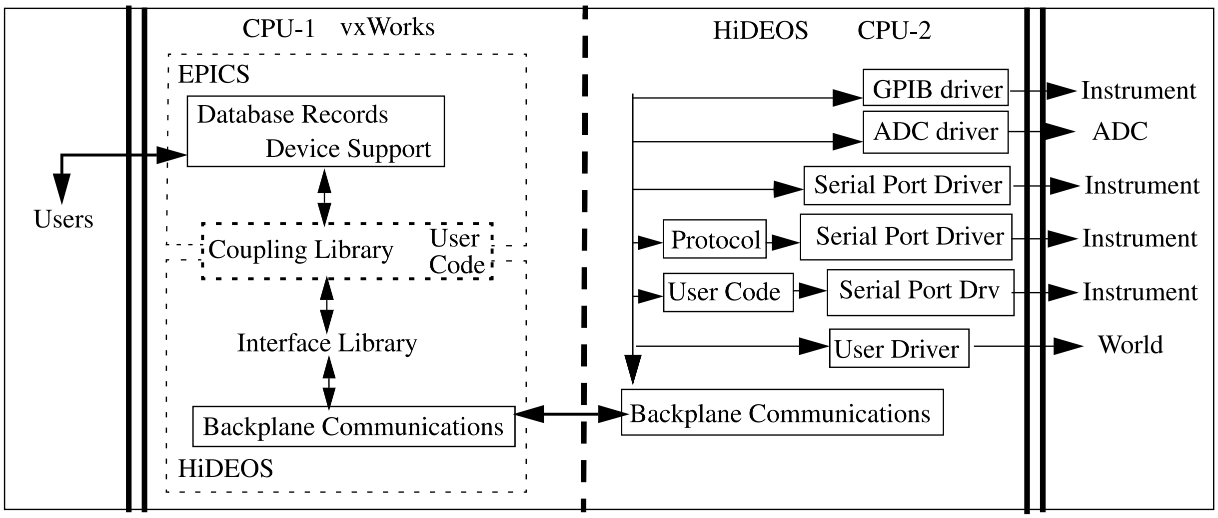
A third goal was to keep the cost of purchasing and running serial ports and GPIB connections low. The Industry Pack modules have helped fulfill this goal. The cost of serial ports on IP modules can be less than $50 per serial port. A GPIB interface module can be added for $200. By using HiDEOS, drivers can be provided to users with no licensing fees. A site running EPICS and owning a vxWorks license can easily add the HiDEOS subsystem for basically the cost of hardware plus setup time.

ARCHITECTURE OVERVIEW

The control system software includes three major pieces: EPICS, vxWorks, and HiDEOS. With the implementation discussed here, EPICS and vxWorks reside and operate on the main embedded system processor and the main portion of HiDEOS resides and runs on a second slave processor in a VME crate. The main control procedures are written in

EPICS which runs under vxWorks. EPICS handles the interaction with remote users and the running of high-level control algorithms. The job of HiDEOS is to operate the IP modules, run protocols or user algorithms and operate instruments connected to the IP modules. It also has the responsibility of running the backplane communications between processors. A portion of HiDEOS runs on the main processor under vxWorks and works together with EPICS to communicate with the slave HiDEOS board.

FIGURE 1. General Architecture



Users interact with EPICS components without any knowledge of the HiDEOS portion and they usually sit on an ethernet LAN away from the VME control crate. As shown in Figure 1, the crate contains two processors: the main controller and the HiDEOS slave MVME162 board. All communications between the two CPUs is made through the HiDEOS package. EPICS allows device support routines to be attached to control system primitives called records. A record can contain an arbitrary number of device support routines of which one is active at a time. The EPICS database designer generally chooses the device from which the primitive is to get data. The user is generally free to implement device support in any fashion; there are really no hard rules on implementing device drivers. HiDEOS has a fairly rigid method defined for writing drivers or software modules which communicate using messages. The coupling library sits between the two systems and serves as a simple place for users to add code to the system. A simple generic EPICS device support routine communicates with the coupling library. The coupling library uses the HiDEOS interface library to push messages into HiDEOS and get results in the form of messages. HiDEOS backplane communication tasks reside on both CPUs and exchange messages. CPU-2 receives commands in the form of messages and distributes them to the running HiDEOS device drivers. This is a very flexible architecture, allowing users to choose the location from which an instrument will be operated. User code can exist as part of the coupling library, where the main processor executes it under EPICS, or as HiDEOS software modules on the slave processor. In addition, users can attach to existing HiDEOS drivers and utilize their services. Figure 1 shows a series of drivers that could be running in a system.

INDUSTRY PACK MODULE SUPPORT

The primary supplier of IP modules for the APS has been Green Spring Computers. A few have come from Systran Corporation. An MVME162 can accommodate up to four IP modules on the IP bus. The HiDEOS package comes with drivers for the following modules:

- IP-Serial - 2 serial ports (RS232/RS422/RS485)
- IP-488 - GPIB
- IP-OctalSerial - 8 serial ports, (RS232/RS422/RS485)
- IP-QuadSerial - 4 serial ports
- IP-PrecisionADC - 12 bit 16 channel ADC

- Systran-DAC128V - 12-bit DAC
- IP-16DAC - 16 channel, 16-bit DAC (under construction)
- IP-16ADC - 16 channel, 16-bit ADC (under construction)
- IP-ADIO - 48-bit digital I/O, 13-bit ADC, counter/timers (under construction)
- Systran-DIO - 48-bit digital I/O (under construction)

The IP bus standard defines an ID memory space which contains manufacturer and model codes. HiDEOS contains a database of all supported IP module make and model codes and provides a way to create a driver for each of them. During the boot process, HiDEOS probes the ID space of each IP bus slot on the MVME162. For each module that is found, the make and model codes are compared to those in the database. If a match is made with a database entry, then a HiDEOS driver is created and given a name that includes information about the slot where it exists. This discovery process allows HiDEOS to be self-configuring for IP modules; a driver will be started for each of the services that needs to be available on the IP bus.

All IP drivers are written under HiDEOS using C++ and the HiDEOS methodology of task writing. Application developers required to create IP device drivers are given several tools through HiDEOS to simplify their effort. The driver creator must conform to the HiDEOS IP module manager database rules in order to have the driver automatically started during the boot process. HiDEOS provides a tool for slot management which allows the user to easily manipulate the IP bus. Some common features are the ability to enable and disable interrupts from an IP slot and find the memory addresses of the different IP memory spaces.

EPICS INTERFACE

All HiDEOS drivers are named with a character string. The character string in the case of an IP module driver is prefixed with a single character indicating the slot in which the module exists. A typical example is the GPIB module. If the user installs a GPIB IP module in slot B of the MVME162, then the string “b-GPIB” will refer to the driver that operates the module. Many of the IP modules have several channels or ports, such as the quad-serial. In this case the string must include the port or channel using the C language style array index operator. If the user wants to communicate with port 3 of the quad-serial IP module installed in slot C, then the name string will be “c-Serial[2]” (indexing starts at zero). The current implementation of HiDEOS requires the user to know the card in which the HiDEOS driver resides. The main processor is zero, the first slave processor is board one. The reason for this is that a user must know exactly where an instrument is attached. To summarize, the HiDEOS IP naming mechanism requires the user to know up to four simple pieces of information: the card in which the driver exists, the slot in which the module exists, the name of the module and the port or channel of the module (if needed).

The naming of a user-supplied HiDEOS process or protocol driver differs slightly from the above scheme. This type is not associated with an IP module, so no slot character is required. Generally these processes do not have multiple ports or channels, so that portion is also not required. Usually only two pieces of information are required: the card where the HiDEOS process is running and the name of the process.

When EPICS database records are created for use with a HiDEOS driver, the database designer fills in the card number where the HiDEOS driver exists and the above name information in the optional parameter field. The coupling library will parse the parameter field and verify the existence of the HiDEOS driver during EPICS boot time.

EPICS is written in C and HiDEOS requires C++. A problem with vxWorks 5.2 is that is does not supply a loader capable of loading C++ object modules correctly. HiDEOS provides a wrapper on top of the vxWorks ld() function called ldpp() which correctly loads C++ modules. The functionality added to ld() is the ability to run global constructors and supply the default new and delete operators.

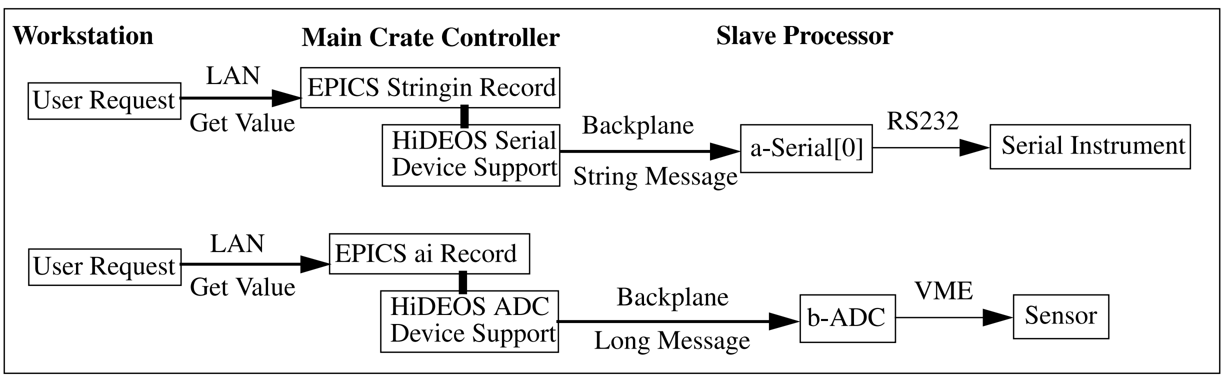
The coupling library discussed earlier hides many of the ugly details of EPICS device support and HiDEOS driver support from a user in a base class. The user is required to create three functions: a start I/O, a complete I/O and an initialization function. The start I/O is invoked by EPICS when a record requires an I/O transaction to be started on a HiDEOS driver. The complete I/O function is invoked by EPICS when the HiDEOS driver completes the transaction started earlier. The initialization function is invoked when the EPICS record is being initialized. The HiDEOS/EPICS interface package includes the coupling library, along with code which does string transactions, ADC/DAC I/O, and GPIB transactions using various EPICS records.

APPLICATIONS

User applications developed so far fit into three categories. The first category includes applications which use existing EPICS and HiDEOS support software. All that is required is to create EPICS database records and specify the parameters correctly. The second category includes applications requiring code to be added to the EPICS/HiDEOS coupling library. This type of application will usually communicate with existing lower-level HiDEOS drivers. The third category requires a HiDEOS driver to be written and run on the second processor. For the purpose of this paper, all applications will require EPICS database records. Some of the more complex applications fit into categories two and three.

Many instruments and sensors can be operated using the first application category. These are important because the user does not need to write any code. An example application of this type is reading from a supported ADC using an EPICS analog input record or writing to a DAC using the analog out record. Another example is writing and reading strings directly to a serial link through EPICS string out/in records. Generally only the simplest of serial-controlled instruments will be able to make use of this method. GPIB devices can be operated in this fashion using the EPICS GPIB library. Figure 2 illustrates two simple example applications which fall into the first category.

FIGURE 2. Simple Applications



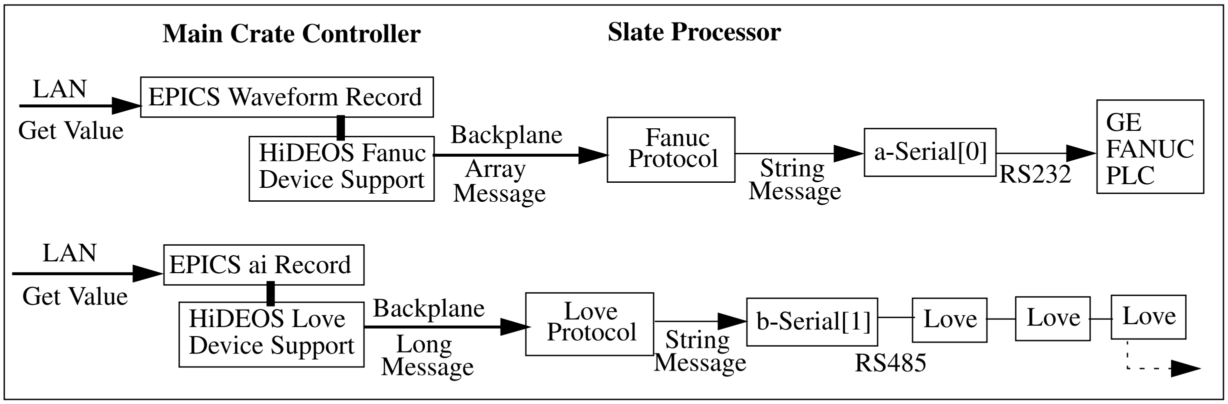
Instruments or IP modules which require simple transactions to operate and return status information fit into the second category. Here a simple piece of code must be added using the coupling library which will in turn be added to EPICS device support. Simple serial-controlled instruments will usually require delimiters to be added to outgoing transactions and require a time-out; incoming transactions will require delimiters to be stripped and data interpreted. This category usually requires a small, simple program to be developed and added to the EPICS/HiDEOS support library which exists under the coupling library of Figure 1.

Multi-function or multi-state instruments such as digital voltmeters, motor controllers, temperature controllers, and PLC usually require a complex protocol to operate. These instruments can be attached to IP modules with existing drivers such as the serial highway or GPIB. A new HiDEOS program will sit between the EPICS/HiDEOS coupling interface and the IP driver to actually operate the instrument. This type of configuration typifies the third application category and requires the largest code development effort. Examples of such a setup are the GE FANUC [3] interface and the Love temperature controller developed for the Experimental Facilities Division of the APS.

In Figure 3, the GE FANUC sends a string of characters representing the current state of the PLC. The existing HiDEOS IP serial driver accepts the string and passes it to a GE-FANUC protocol driver which deciphers it and decides whether or not to inform the EPICS control system of any state changes. The EPICS/HiDEOS device support procedure converts the data to EPICS-acceptable format and notifies other EPICS database records of change-of-state information. The GE FANUC provides data to the HiDEOS processor continuously at 19,200Kbps which would put a large load on the main processor if it were not configured in the two processor fashion.

The Love controllers operate on a RS485 multi-drop network. The HiDEOS Love controller protocol driver polls the controllers for temperature and status reading using the Love protocol. The data is converted to a simple transaction that can be sent to the EPICS/HiDEOS support module where EPICS database records are updated.

FIGURE 3. Bigger Applications



CONCLUSION

This development effort has demonstrated that HiDEOS can be integrated into an existing system. Interacting with HiDEOS using its interface library is straightforward providing the user has an understanding of the HiDEOS system. Addition of the coupling library was an important part of the development, as it allows easy access to commonly used pieces of both systems. The current complete system has a considerable number of important IP modules and sample applications developed, with more to be added. HiDEOS is explained in more depth in [4].

The learning curve for understanding the HiDEOS system and using its facilities can be high for a user not familiar with object-oriented programming and design in C++. Understanding each component which must be modified or added to EPICS and vxWorks start-up scripts is difficult. Like EPICS, the HiDEOS package cannot just be picked up and used without first understanding a set of fundamental principles.

ACKNOWEDGMENTS

Thanks to Dave Reid, Josh Stein, Tim Mooney, and Greg Nawrocki of the APS for providing applications, hardware, and time for testing the initial development of the HiDEOS system and the EPICS interface library. This work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

REFERENCES

[1] J.Kowalkowski; “Home Of HiDEOS,” World Wide Web URL: https://www.aps.anl.gov/asd/controls/hideos/intro.html.

[2] W.McDowell; “Experimental Physics Industrial Control System (EPICS),” World Wide Web URL: https://www.aps.anl.gov/asd/controls/epics_home.html.

[3] J.Stein; “Personnel Safety System Status Reporting Via EPICS at the Advanced Photon Source,” these proceedings.

[4] J.Kowalkowski; “An Object-Oriented Approach to Low-Level Instrumentation Control and Support,” these proceedings.