A SHARED MEMORY INTERFACE BETWEEN LABVIEW AND EPICS*

D. Thompson and W. Blokland, ORNL, Oak Ridge, TN, USA

Abstract

The diagnostics systems of the Spallation Neutron Source project are based on rack-mounted PCs with off-the-shelf and custom PCI hardware and LabVIEW. About twenty out of the total of three hundred systems are already installed and have been integrated into the EPICS-based control system. LabVIEW communicates with the EPICS IOC using a simple shared memory interface implemented in a dynamic linked library (DLL) based on previous work by Los Alamos National Laboratory. The DLL supports events and buffered communication of scalar data types such as integers, floats, and booleans, as well as single dimension arrays and strings. The LabVIEW programmer needs only minimal EPICS knowledge to use the interface. At a rep rate of 6 Hz, the writing and transmitting of 4 arrays of 1025 double precision variables plus twenty individual doubles give a 3% CPU load on a 2 GHz PC. This is fast enough for the planned applications. At higher rep rates, the interface comes close to utilize the full bandwidth of a 100 Mbit/sec ethernet connection. In this paper we describe the implementation and performance of the shared memory interface on both the EPICS and LabVIEW side.

INTRODUCTION

The diagnostics systems are designed as Network Attached Devices (NAD), see [1], and implemented using rack-mounted PCs. The PCs currently run Windows as their operating system and LabVIEW as their programming environment. The NADs communicate using EPICS, see [2] with the rest of the control system. The initially deployed NADs used the ActiveX/EPICS interface, see [3]. Starting with version 3.14 of the EPICS IOC, other platforms than VxWorks, including Windows, are now supported. Switching to IOC combined with a Shared Memory Interface now gives full compatibility with the Controls Group's IOC nodes, increased performance, and support for platforms other than Windows. The Shared Memory Interface described in this paper has a compatibility layer to work with the shared memory interface as implemented by LANL. LANL used a shared memory layer to pass data to the ActiveX/EPICS interface just in case one wanted to switch to a different EPICS interface.

* The Spallation Neutron Source (SNS) project is a partnership of six U.S. Department of Energy Laboratories: Argonne National Laboratory, Brookhaven National Laboratory, Thomas Jefferson National Accelerator Facility, Los Alamos National Laboratory, Lawrence Berkeley National Laboratory, and Oak Ridge National Laboratory. SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy.

THE SHARED MEMORY INTERFACE

The Shared Memory Interface (SMI) has three components: a dynamic link library (DLL), modified IOC device support, and a LabVIEW library.

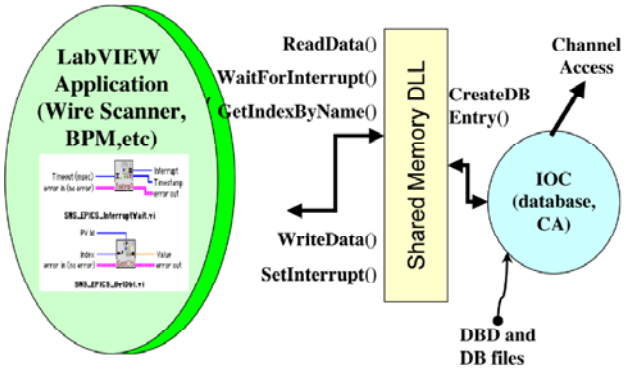

Figure 1: The Shared Memory Interface.

The DLL itself is not dependent on EPICS or LabVIEW and can be used without either. Data is shared equally between applications attached to the DLL. Figure 1 shows how the Shared Memory DLL connects LabVIEW to EPICS using a common interface.

In addition to providing the reading and writing of data, the DLL also supports the event handling. One application can signal another application waiting on the data to synchronously process the data.

The DLL supports two modes of operation. Data can be shared asynchronously supporting a blackboard data model, or synchronously supporting a message-passing model. All data can be read from the DLL asynchronously but only data inserted into the DLL as synchronous data can be read synchronously.

The data structure inside the DLL uses multiple buffers even for asynchronous data. In asynchronous mode, data is always read from the last buffer available. If data is simultaneously being written to the same named variable, the operations will be using different buffers. Data locks are sparingly used in conjunction with sequence tags to prevent and detect buffer errors. In the case of a buffer overrun, where data is written to a buffer being used by a read operation, the read operation will detect that the sequence number changed and will re-copy the buffer into the reader's data area. This rarely happens on a single CPU system.

In synchronous mode, data items are written to buffers as before but the buffer pointers are advanced only when a processing event is sent with a time stamp. Synchronous readers also have a buffer pointer and can only read data from the head of a named buffer and only after receiving the processing event for that data. The DLL implements this behavior in a ring buffer that has a user adjustable size. The ring buffer holding the data in conjunction with

the processing event message acts as a first-in first-out buffer holding data for more than one trigger event. Each time a reader receives an event, a new time stamp is made available along with the head of each process variable FIFO with waiting data. The reader will be blocked waiting for data if no new data is waiting in the FIFOs. The DLL supports the scalar and array data types of char, uchar, short, ushort, long, ulong, float, and double, and the data type string. The matching EPICS record types are ai, a0, longin, longout, bi, bo, waveform, and string.

INTEGRATING THE SHARED MEMORY WITH EPICS IOC

Our main focus was to integrate LabVIEW with EPICS and we have therefore implemented several functions in the DLL to support EPICS. Special attention was given to maintain time correlation of data through the use of time stamps and time stamped events.

The IOC uses the Channel Access protocol for communication with other nodes and provides the infrastructure to manage the creation and processing of data structures known as records. Records are the data types of EPICS and support both scalar data and arrays bundled with attributes such as the process variable's name, processing rates, units, and alarm limits. The DLL supports these attributes.

Two device support routines, one for initialization and one for processing are required for each record type that is supported by the IOC. These routines are modified to make calls to the DLL.

Initialization

An EPICS IOC starts by loading the binary software image and then a 'dbd' file containing a description of all the data records and enumerated types used in the in-memory database. The instances of variables are defined in 'db' files. During processing of a 'db' file, record and device specific routines are called to initialize the record. The DLL is called during each record instantiation to create the shared memory variable and link the record data field to the shared memory variable. In the diagnostics applications at SNS the IOC is responsible for completely initializing the DLL using data in the 'db' file. The LabVIEW program or any other program could initialize the shared memory if desired.

Operation

Data arriving from Channel Access or from a data base processing causes an output record to process triggering device support. A call is made by device support to the DLL writing the data to the shared memory. The EPICS timestamp or, optionally, the system timestamp is passed along to the shared memory variable and made available to LabVIEW.

Data from the LabVIEW application can be processed asynchronously by specifying a scan rate in the "SCAN" attribute of the EPICS record. If synchronous operation is desired then the "SCAN" field is set to "I/O interrupt". The IOC core treats processing events from other DLL clients as if an interrupt had arrived in a hardware IOC.

INTEGRATING THE SHARED MEMORY WITH LABVIEW

Many of the functions of the DLL are called from LabVIEW using the Call Library Node function. These functions include

1) read and write to variables,
2) find and get information about variables, and
3) set and receive events (blocking and non-blocking).

A software suite has been created to provide the programmer with documentation, tools, and templates to write LabVIEW code with the Shared Memory Interface, see [1].

To access the shared memory functions, the LabVIEW program needs an index identifying the particular variable. This is done by the function *Name2Id* that resolves the variable name to an index. On the IOC side of the shared memory, the variables names are the Process Variable (PV) names and these must be defined before starting the IOC by means of above mentioned 'db' file.

The LabVIEW routine that performs the name resolving operation is called *GetPVs*. *GetPVs* in effect also declares which variables the programmer is interested in. With a few EPICS specific details about the variable, such as scan type and scan rate, added to *GetPVs*, the utility *GenerateDB* has enough information to generate the 'db' file, the IOC startup command file, and a spreadsheet documenting the generated variables. This mechanism provides a single point from where the shared memory variables and EPICS PVs are defined; by the LabVIEW programmer and in the LabVIEW code. An example of a *GetPVs* code snippet that finds the index of a variable but also declares the variable is shown in Figure 2.

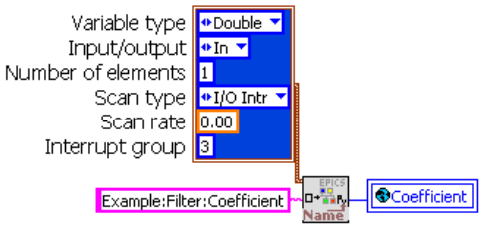

Figure 2: The Resolving and declaring of the PV names.

An example of the 'Initialize' state in a LabVIEW program is given by Figure 3. LabVIEW starts the IOC using a command file and registers the events (EPICS Interrupt Groups) it needs to receive. The IOC will now start and instantiate the shared memory variables based on the 'db' file generated from the GetPVs routine. Next, the GetPVs routine will find and store the variable's indices into a LabVIEW global. The shared memory interface is now established and the LabVIEW program can continue.

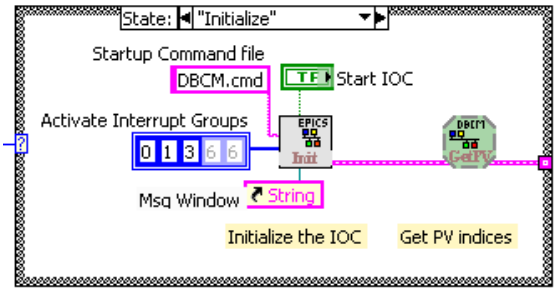

Figure 3: The startup state of a LabVIEW program.

PERFORMANCE

The performance of the Shared Memory Interface is listed in Table 1. The times to write or read the array of double precision variables are measured using the built-in LabVIEW profiler on a 2 GHz Pentium 4. The profiler gives times excluding other processes. When including other processes and program looping overhead, the read and write times increase by about a factor of two. The throughput is measured over a 100Mbit/s ethernet connection from the above-mentioned PC to a PC with a 2.4 GHz Pentium 4 running Windows XP. The client PC uses the LabVIEW Channel Access tools, also written at SNS, see [4]. The throughput for the larger array sizes is close to the practical maximum of the ethernet connection. The client PC has a CPU load of less than 5% when receiving 1024 doubles at 500Hz (about 4 Mbytes/sec) while the server PC has about a 10% CPU load. The performance numbers are excellent and suffice more than enough for the typical SNS application running at 6Hz. A high throughput system like a Beam Current Monitor or Beam Position Monitor generates data with about 10 double precision variables and up to four arrays of up to 1024 doubles. Communicating this data to one client at 6 Hz results in a 3% load on the CPU. Enough CPU time is left for data-acquisition and processing, and extra clients.

Table 1: The performance of the SMI

Array Size of doubles	Write (µsec) +- 25%	Read (µsec) +- 15%	Throughput (Mbits/sec) +- 5%
4096	23	90	84
1024	15	19	65
128	14	12	27
1	11	8	.4

The reliability of the Shared Memory Interface, the IOC, and LabVIEW has been very good. There have been no problems with the installed IOC-based systems running for up to 4 weeks. A longer term study has not been possible as of yet because of power outages or program code modifications. The currently installed systems are listed in Table 2. We expect to have installed around 300 systems based on the IOC, Shared memory, and LabVIEW at the completion of the SNS project.

Table 2: Installed Diagnostic LabVIEW systems

Installed Systems	MEBT	DTL	Test	Comments
Beam position	6	2		IOC/ActiveX
Wire scanner	5	2		ActiveX
Faraday cup	1	0		IOC
Beamstop	1	0		IOC
Aperture	1	0		IOC
Haloscraper with Beamstop	0	1		IOC
Faraday Cup with Energy Degrader	0	1		IOC
Fast Beam Current Monitor	0	1		IOC
LEBT Beam Current Monitor	1	0		IOC
Fast Beam Loss Monitor	0	1		IOC
Physics Test			1	IOC to test applications
Total	**15**	**8**	**1**	**24**

CONCLUSIONS

The Shared Memory Interface provides a connection between the LabVIEW application and the EPICS based control system. The LabVIEW programmer has a software suite available to him or her to assist in writing LabVIEW/EPICS applications and is thus not required to have intimate knowledge about the EPICS IOC to get a system up and running. The Shared Memory Interface is more than fast enough for the planned SNS applications.

Using the Shared Memory Interface, any PC application can be made available to channel access with the full benefits of EPICS record support.

REFERENCES

[1] W. Blokland T. Shea M. Stettler, "Network Attached Devices," pp 151-153, DIPAC2003, Mainz, Germany, May 5-7, 2003.

[2] http://www.apa.anl.gov/epics

[3] K.U. Kasimir. M. Pieck, L.R. Dalesio, "Integrating LabVIEW into a Distributed Computing Environment," pp 548-550, ICALEPCS'01, San Jose, CA, USA, Nov 27-30, 2001.

[4] A. Liyu, "EPICS C and LabVIEW Clients," Manual, SNS, Oak Ridge, TN, USA, 2003.