CU-NEES-07-16

NEES at CU Boulder

01000110 01001000 01010100

The George E Brown, Jr. Network for Earthquake Engineering Simulation

SCRAMNet LabVIEW

DAQ

By

Robert Wallen

Center for Fast Hybrid Testing
Department of Civil Environmental and Architectural Engineering
University of Colorado
UCB 428
Boulder, Colorado 80309-0428

Sept. 2007

1 Introduction

Real-time Fast-Hybrid Testing (FHT) is a unique data acquisition application. During a typical Fast-Hybrid test a physical substructure model or element is coupled with a numerical model and subject to earthquake motions in real-time. Displacement conditions are forced on upon the physical element by the numerical model via high-speed actuators and the resulting force is fed back into the numerical model. Data must be acquired from physical test specimen using conventional transducers and from the numerical model using real-time data recorders or "virtual transducers". These two data acquisition tasks must collect time-synchronized data in a fashion that does not affect the determinism of the numerical model.

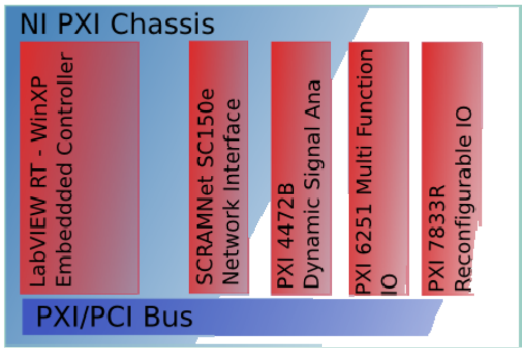
2 SCRAMNet

A Shared Common Random Access Memory Network (SCRAMNet) is utilized to perform the essential low latency, low overhead communication between each component of the FHT system. The current network consists of two real time computational targets, an MTS digital servo-hydraulic controller, a streaming data server, and the Hybrid DAQ system. SCRAMNet operates by mapping 2 MB of reflective memory as host system RAM. When any host on the network writes to its mapped portion of RAM, the SCRAMNet updates the corresponding memory on each host in the network. Certain memory locations can be configured to generate network interrupts when they are written. Network interrupts provide a means to synchronize each host on the network.

3 DAQ Architecture

3.1 Hardware

The Hybrid SCRAMNet DAQ was developed using LABView Real-Time (RT). LABView RT follows the host-target model where software tools are utilized in a host operating system environment (typically Windows) to develop a real-time application that runs on a target hardware platform. The CU NEES site utilizes a National Instruments PXI target platform with a PXI-8187 embedded controller configured to boot into LABView RT or WindowsXP. The PXI target is equipped with conventional multifunction IO hardware and dynamic signal analyzers in addition to a Systran SC150e network interface card. Refer to figure 1 for the hardware layout.


NI PXI Chassis
LabVIEW RT - WinXP Embeddded Controller
SCRAMNet SC150e Network Interface
PXI 4472B Dynamic Signal Ana
PXI 6251 Multi Function IO
PXI 7833R Reconfigurable IO
PXI/PCI Bus
Figure 1 Hybrid DAQ Hardware Layout

3.2 Software

The LabView SCRAMNet DAQ consists of a top-level application, which interacts with the SCRAMNet card through a collection of lower level functions. The low level functions utilize the NI Virtual Instrument Standard Architecture (VISA) library to communicate with the SCRAMNet card. The VISA library provides a platform independent interface for reading and writing the SCRAMNet Control Status Registers (CSRs), reading and writing SCRAMNet shared memory, and handling network interrupts through the VISA event structure. Refer to figure 2 for the layout of the software components.

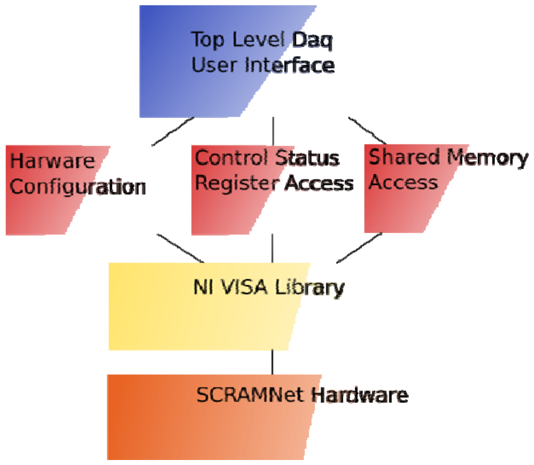
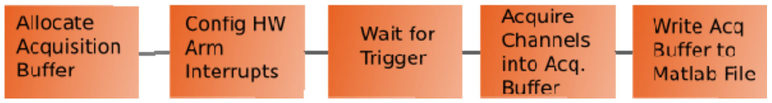


Figure 2 Software Component Layout

3.2.1 Top Level Daq Application

The user interface to the SCRAMNet DAQ system occurs through a top level LabView virtual instrument. This application allows the user to specify the channel list, sampling time and data file name. Currently channel names and their corresponding SCRAMNet memory adresses are hard coded into a control type definition. Future develop efforts will have the MTS controller data channels hard coded and the OPENSEES numerical model channels dynamically configured using configuration information written to the upper portion of SCRAMNet memory. For each new series of FHT tests a customized version of the top-level DAQ program is typically developed from a basic template. A custom VI allows triggering, sample time and channel lists to be customized for each test while minimizing the user interface complexity.


Allocate Acquisition Buffer
Config HW Arm Interrupts
Wait for Trigger
Acquire Channels into Acq. Buffer
Write Acq Buffer to Matlab File
Figure 3 Top Level DAQ Program Flow

The top level VI serves as the main loop of the DAQ application. The basic program flow is summarized in figure 3. First, the DAQ application gets the desired channels and sample time from the user interface and uses this information to allocate a memory buffer to store the acquired data before it is written to disk. Then VISA_SCRAMNetInteruptConfigure is called to configure the SCRAMNet card to generate interrupts on a network writes to a specified address. Once the interrupt configuration is performed, the program enters a loop that checks for a specified trigger condition each time a network interrupt occurs. The program currently uses either a change on a specified channel or a manual signal from the user interface as the trigger event. Once the trigger event occurs, the program enters the main acquisition loop. The acquisition loop is timed by the network interrupt by using the sub VI VISA_SCRAMNetWaitForInterupt. After an interrupt has occurred it is processed using VISA_SCRAMNetReadInterruptingAddress and VISA_SCRAMNetClearRearm. During each interrupt iteration the preprogrammed channel lists is copied from SCRAMNet to the buffer array using the VISA 32-bit peek function inside of a nested for loop. After the desired acquisition time has elapsed the program exits the acquisition loop and sends the data array to the WriteMatlabFile VI.

3.2.2 Sub VIs

3.2.2.1 VISA_SCRAMNetInterruptConfigure

The interrupt configuration routine performs basic configuration operations on the SCRAMNet hardware and arms the device the process network interrupts though reading and writing hardware registers through VISA library calls. In order to access the device, the sub routine opens four VISA sessions. Three VISA sessions are configured to access the SCRAMNet shared memory, SCRAMNet control status registers (CSRs), and the SCRAMNet PCI configuration space. The fourth VISA session configures the VISA event structure to handle to PXI interrupts. The main configuration steps are as follows. The PCI configuration space is written to disable byte swapping on shared memory reads. Then CSR 0 is set to allow access to the auxiliary control registers (ACRs). Using a for loop and the shared memory VISA reference, all ACRs are set to zero. This step disables network interrupts on all memory locations. Then the desired network interrupt address is programmed. CSR 0 is rewritten in the next step to disable ACR access. Then CSR 2 and CSR 3 are configured for burst mode with no fiber-optic loop back and the VISA event session is configured for PXI interrupt events. CSR 1 and CSR 0 are written to configure the network interrupts and the interrupt FIFOs are reset. The final configuration step enables interrupts on the V3 PCI bridge chip.

3.2.2.2 VISA_SCRAMNETWaitForInterrupt

The wait for interrupt vi simply calls the VISA Wait for Event vi which suspends the execution of the program until a SCRAMNet interrupt has occurred or a time out period of 2 ms has elapsed.

3.2.2.3 VISA_SCRAMNETReadInterruptingAddress

The read interrupting address vi obtains the list of interrupting addresses out of the interrupt FIFO by reading and decoding values stored in CSR 4 and CSR 5. For any given call of VISA_SCRAMNETReadInterruptingAddress there should be only one address in the FIFO, the address programmed to produce interrupts. If the DAQ program fails to finish acquiring all of the data before the next interrupt occurs, there will be duplicate values of the interrupting address in the interrupt FIFO. This is detected as an error case by the top-level DAQ application.

3.2.2.4 VISAClearRearm

This vi clears the current interrupt and rearms the SCRAMNet card to generate future interrupts by writing to CSR 1.

3.2.2.5 WriteMatlabFile

The WriteMatLabFile vi writes the acquired data to a Matlab format file. The Matlab format was chosen for its compact packed binary format. Post processing of data is always performed with Matlab due to its ability to handle the large files (in excess of 400 MB) often generated during FHT Testing. The Matlab file writer also produces a time column and channel name header in the data file.

3.2.3 Current and Future Development

Current and future development for the Hybrid DAQ system is focused on two main areas. First, there is a strong need to automatically configure the channels to be acquired from the numerical model. In this scenario SCRAMNet node recorder commands in the OPENSEES input script will be translated into channel metadata written to the upper 1 MB of SCRAMNet memory. The Hybrid DAQ system will parse the metadata and use it to configure the acquisition. Using this scheme, the channel metadata is easily propagated from the numerical model to the output data file without affecting the determinacy of the real-time test.   The second area of future enhancement will deal with a more robust method of synchronizing the data acquisition. Currently the acquisition is synchronized with the MTS generated network interrupt. This interrupt signals the availability of data from the MTS controller and does not necessarily mean that data is available from the numerical model. Additional signaling is need to guarantee that data is aligned correctly.