ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA137

TIMING SYNCHRONIZATION AND CONTROLS INTEGRATION FOR ESS DETECTOR READOUT

W. Smith[†], STFC, Sci-Tech Daresbury, Warrington, UK
J. Nilsson, S. Alcock, ESS, Lund, Sweden

Abstract

The European Spallation Source (ESS) is a new facility being built in Lund, Sweden, which when finished will be the world's most powerful neutron source. STFC has an in-kind project with the Detector group at ESS to provide timing and control systems integration for the detector data readout system. This paper describes how time is synchronised and distributed to the readout system from the ESS timing system, and how EPICS is used to implement a controls interface exposing the functionality of detector front ends.

INTRODUCTION

ESS will be the leading facility in Europe in neutron science [1]. Neutrons are produced at the facility through the process of spallation when protons collide with a tungsten target. These neutrons are guided along beamlines to scientific instruments. The detectors that form part of these instruments generate neutron events which are then timestamped.

Data produced by detectors is acquired by detector readout front end nodes. These are connected in a ring or star topology to a detector readout master. The readout master runs firmware on an FPGA that controls the front ends, ingesting the data they produce and sending it to the DMSC (Data Management and Software Centre. The readout master is monitored and controlled by an EPICS IOC (Input Output Controller) [2]. (See Fig. 1)

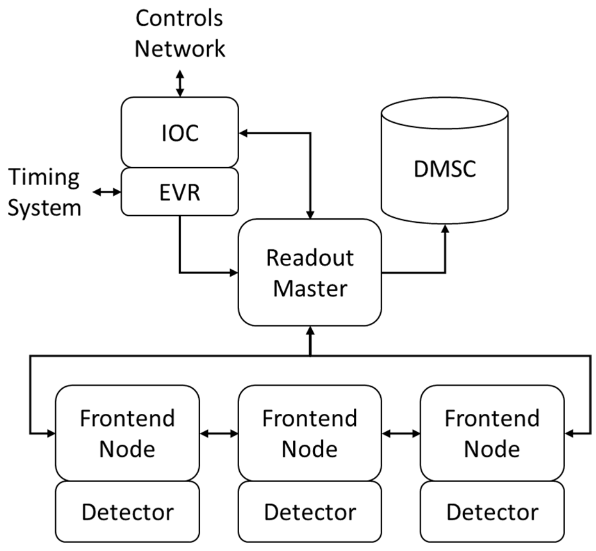

Figure 1: Detector readout system ring topology.

† william.smith@stfc.ac.uk

The readout system does not specify anything about the front ends. Instead the readout master exposes a register space to the control system and forms packets of data to be sent to the DMSC. The EPICS support module defines the communications between the control system and the detector. This definition is used to along with a register map for a specific detector implementation to create an EPICS IOC. Integration with the control system allows parameters on the detectors and the readout master to be monitored and controlled, alarms generated, logged and responded to, and data archived in the EPICS Archiver Appliance [3][4].

Timing System

ESS uses an event timing system provided by Micro Research Finland (MRF) to distribute an 88.0525 MHz clock, triggers, timestamps, and other data about the accelerator [5]. This clock and time must be transferred to the detector readout master and forwarded to the front end. They are then used to timestamp responses from detectors for time of flight measurements and more generally correlating data across many different systems. The readout system aims to provide timestamps with a precision of at least 100ns.

TIMING SYNCHRONISATION

The timing system briefly comprises of event generators (EVM) and receivers (EVR) connected by a fibre network. A GPS receiver provides a 1Hz signal to an EVM which transmits this along with an 88.0525MHz event clock. EVMs transmit event codes and EVRs recover the clock and then decode and respond to event messages. Timestamps are transmitted by sending the current NTP time (Network Timing Protocol) [6] from the EVM to each EVR and then sending the 1Hz signal to validate the time.

Time and a synchronous clock could be provided to the detector readout system by embedding an event receiver in the detector readout master. This would require either using transceivers on the readout master FPGA [7]which would reduce the bandwidth of the readout system, or developing an FMC (FPGA Mezzanine Card) based EVR. It was decided early on in the project not to do this because it required too much effort to develop. Instead the time and clock on a separate PCIe EVR would be synchronised with the time and clock on the readout master.

Due to limitations of the PCIe EVR, the event clock can only be supplied through a prescaler. As such the highest clock that can be transferred is 44.02625MHz. This is sent from one of the outputs of the EVR to the detector readout master. Having provided this clock the synchronisation process involves sending a time in the future and then sending a pulse from another EVR output to validate that time. When the detector readout master receives the time valid signal it starts counting with the clock provided. This timestamp counter is then distributed around the ring to front end detector nodes. Data acquired at those nodes can be associated with those timestamps.

mrfioc2 Implementation

The ESS branch of the EPICS support module mrfioc2 [8] is used to implement the synchronisation process (See Fig. 2). Sequencers in the EVR define event codes that can be generated with known delays. A sequencer is configured to run once if enabled and if the 1Hz event is received from the EVM. The sequencer contains events that cause the following operations:

1. Reset the EVR prescaler. This allows the phase of the time valid signal relative to the clock sent to the detector readout master to be consistent every time the synchronisation process is performed.
2. Send “Reset” and “Enable” commands to the detector readout hardware.
3. Read the timestamp latched by the EVR when the sequencer was triggered and send that plus some additional time to the detector readout hardware.
4. Send the time valid pulse at the appropriate time to validate that timestamp.

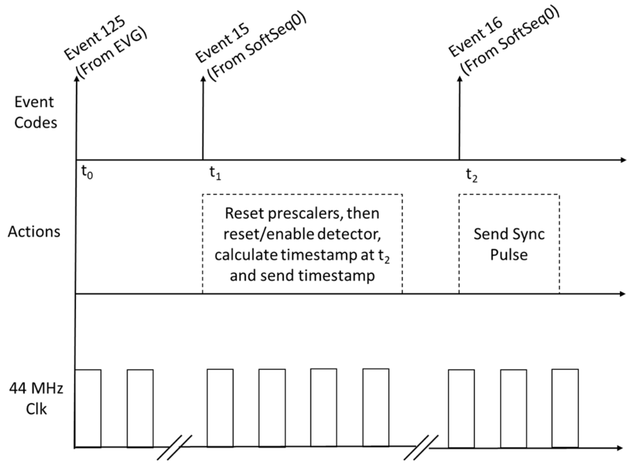

Figure 2: Synchronisation process showing events in EVR sequencer and the actions that they cause.

Accelerator Metadata

The accelerator repetition rate is 14Hz. Every pulse of the machine is associated with timing system event number 14. The timestamp of this event is used to correlate data about the accelerator with other systems. Data collected from the detectors must have this timestamp associated with it.

To achieve this the sequencers previously used for timing synchronisation are reconfigured to assert the time valid signal every time the event 14 is received. The firmware responds to this by latching the current value of the timestamp counter. Since the two systems are operating synchronously the detector readout system now has the accelerator pulse timestamp. This is included in the waveform packets that are sent to the DMSC.

CONTROL SYSTEM INTEGRATION

It was decided that the EPICS IOC, the program that handles communications between the detector system and the rest of the ICS (Integrated Control System), would not run on the readout master FPGA. This would have meant using either a Zynq SoC [9] or instantiating a processor like a MicroBlaze [10] on the FPGA, but neither were supported by the ICS at the time. Instead, the IOC runs on an industrial PC and communicates with the readout master using a serial communications protocol defined by the ESS Detector Group. A USB connection between the IOC host machine and a UART interface on the readout master is used for the communication. The support module uses StreamDevice to implement the communications protocol [11]. This was chosen for ease of implementation.

Register Access

The support module allows all registers in the register space to be read and written individually or as blocks. This is useful when setting up a detector or to pull the current state of the register space.

Other registers which hold information that relates to the setup of the readout system or the status of the detector are additionally exposed as EPICS records. While the need for this is clear the specification of what these records should be and which registers they should relate to is not known until the instruments themselves are complete. As a result the support module only defines the methods of communication and timing synchronisation.

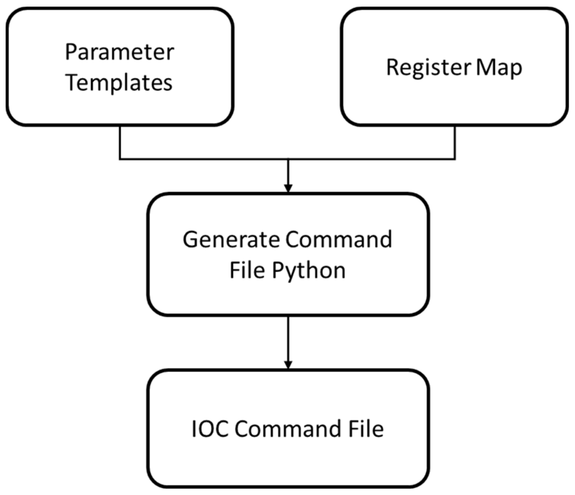

Figure 3: Generation of IOC from register map using parameter templates.

Since the register map can change throughout development and because many of the parameters are of similar types, templates are used to define the records needed by different classes of parameters. These can be as general as read write 32 bit integers or as specific as MAC addresses which span multiple registers. Parameter templates define the records required to expose that parameter in EPICS and are used with a register map that specifies offsets and names to generate an IOC. The templates make use of the Autosave support module for read write parameters [12]. This is used to restore the values entered previously if the system is restarted.

A Python script is used to parse a register map file and create an EPICS command script that loads the appropriate templates (See Fig. 3). Adding a new register is simple; provided a template for a similar parameter exists all that is required is a one line change to the register map. This is especially useful for designs employing lots of groups of similar parameters, such as the data acquisition channels of a detector. No knowledge of EPICS is required to add new registers that are exposed in the control system.

DEMONSTRATOR

A demonstrator was built to prove the integration of the detector readout master with the controls and timing system. It also provided a system to test the acquisition and transfer of data from the detector to the DMSC. This comprised of an FPGA development board with a 4 channel ADC mezzanine card. The demonstrator was connected to an industrial PC that ran the IOC and controlled the PCIe EVR that provided a clock and time (See Fig. 4). All of the parameters of the demonstrator are exposed as EPICS records. Parameters that give information about the status of the system are archived using the EPICS Archiver Appliance. A CS Studio user interface was developed to allow users to configure the acquisition, network and synchronisation settings [13].

Testing and Deployments

The demonstrator was initially deployed at a lab at ESS and used to develop the timing synchronisation process. Tests at the V20 beamline at HZB Berlin followed. The demonstrator provided timestamped synchronous data acquisition with integration with the controls system and DMSC and this made it attractive to other projects at ESS. It was decided that the system could be used for readout of beam monitors. This added additional requirements, most notably running, and synchronising, multiple demonstrators from one IOC and timing system. It was this that pushed the development of the IOC generation script.

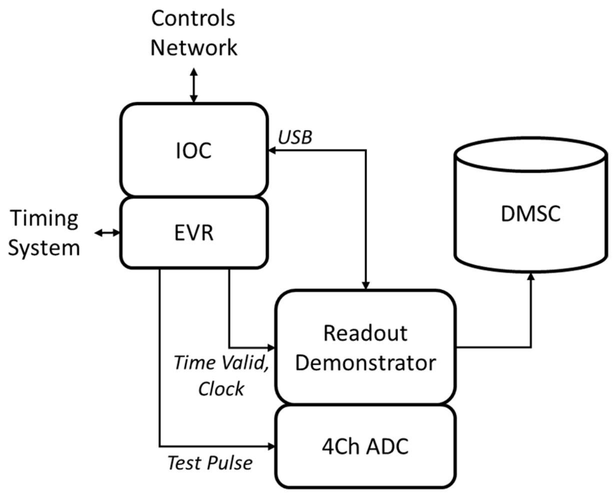

Figure 4. The readout demonstrator system showing connection from the EVR to the ADC used to perform end to end test.

An end to end test at the lab at ESS involved sending a pulse at a known time from the EVR to one of the channels of the synchronised readout demonstrator. The timestamp of the rising edge of the acquired pulse was then monitored. The timestamp was shown to have a constant offset relative to the time it was sent from the EVR. This showed that the demonstrator was able to maintain synchronisation with the rest of the timing system. Provided the bias is accounted for the timing resolution of the system is limited by the stability of the clock, exceeding the specification of 100ns precision.

FURTHER WORK

While the demonstrator successfully demonstrated synchronisation and single register communications more work is needed. StreamDevice allowed for simple implementation but it doesn't easily lend itself to block read and write. Instead the Asyn EPICS support module will be used to define the communications [14]. This allows much more flexibility.

The USB based communications will be replaced with a 1 Gigabit Ethernet connection. This will improve the reliability of the connection, increase the data rate and make setup of multiple devices easier.

The register map that is used to generate the IOC is currently not in the same format as a similar file used to generate the firmware that runs on the readout system. These two processes will be merged so that changes only need to be made to one file.

Finally the demonstrator will be extended to include the ring of front end nodes. More tests to prove the synchronisation of the system will be required.

CONCLUSION

An EPICS support module which controls the ESS detector readout hardware has been demonstrated. A script to parse a register map makes modifications to the system easier and less prone to error, requiring minimal knowledge of EPICS. These processes will be unified with a similar script for firmware generation from a single register map file. Further work is needed to implement block read and write of registers and use ethernet for communications

Synchronisation with the timing system has been proven in tests at ESS and during deployments at the V20 beamline at HZB. The system has been shown to exceed the timing accuracy specification required by ESS.

REFERENCES

[1] ESS, `https://europeanspallationsource.se/`

[2] EPICS, `https://epics.anl.gov/`

[3] K.-U. Kasemir, X. H. Chen, and E. Danilova, “The Best Ever Alarm System Toolkit”, in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper TUA001, pp. 46-48.

[4] M. V. Shankar, L. F. Li, M. A. Davidsaver, and M. G. Konrad, “The EPICS Archiver Appliance”, in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 761-764. `doi:10.18429/JACoW-ICALEPCS2015-WEPGF030`

[5] J. Cereijo GarcÃ, T. Korhonen, and J. H. Lee, “Timing System at ESS”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 618-621. `doi:10.18429/JACoW-ICALEPCS2017-TUPHA088`

[6] NTP, `http://www.ntp.org/`

[7] openEVR, `https://github.com/jpietari/mrf-openevr`

[8] mrfioc2, `https://github.com/icshwi/e3-mrfioc2`

[9] Zynq, `https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html`

[10] MicroBlaze, `https://www.xilinx.com/products/design-tools/microblaze.html`

[11] StreamDevice, `http://epics.web.psi.ch/software/streamdevice/`

[12] Autosave, `https://epics.anl.gov/bcda/synApps/autosave/autosave.html`

[13] CS Studio, `http://controlsystemstudio.org/`

[14] Asyn, `https://epics.anl.gov/modules/soft/asyn/`