March 16

A FELIX Based Interface to DAQ and DCS for Neutrino Experiments

Office 2004 Test Drive User

This documents describes a FELIX based architecture for a Data AcQuisition (DAQ) and Detector Control System (DCS) that may be used a future neutrino experiment, such as DUNE. It proposes a plan in order to be able to proof the validity of the approach at protoDUNE. An important part of the document is dedicated to establishing the list of pre-requisites that need to be clarified in order to be able to design a FELIX based DAQ/DCS system, in particular for the protoDUNE detectors.

G. Lehmann Miotto, CERN/EP-DT-DI

1 Introduction

The FELIX [1][2][3] ("Front End LInk eXchange") is a PC based system that translates data coming from point-to-point detector front-end links into packets that are routed towards a series of destinations over a switched network and that, reversely, routes data coming from a switched network and destined to the detector front-end over point-to-point links. The typical use-case of it is the data transfer to and from the Data AcQuitision (DAQ) and the Detector Control System (DCS). One or more PCIe cards handle the I/O from the point-to-point links to the host memory. A software program organizes the I/O from the host memory to a switched network.

The FELIX has been introduced for the ATLAS experiment upgrade with the aim of interfacing detectors as early as possible to commercial networks and computers, thus maximizing the modularity and upgradability of the different components in the data acquisition and detector control chains. The protocol currently supported by the FELIX on the point-to-point links is GBT [6], though other more lightweight protocols will be considered in the future for detectors that are not subject to high radiation environments.[1]

In this document we describe a possible system architecture and outline a work plan in order to develop a prototype system to be used for the readout and control of a slice of the protoDUNE detector that will be installed at CERN in 2017.

In the first section we describe the pre-requisites that need to be clarified with the experiment's collaboration in order to be able to start the design phase and then smoothly integrate the desired functionality into a prototype implementation.

Thereafter, the system architecture is presented. It is meant to guide the discussion about the desired solution within the collaboration.

In the last section a tasks breakdown proposal is outlined, as a basis for organizing the work among all interested institutes, according to their skills and available manpower.

2 Prerequisites and requirements

2.1 Functionality definition

The FELIX can support detectors with five distinct functionalities:

- Readout of physics/calibration data
- Readout of detector monitoring quantities (temperatures, voltages, status, ...)
- Distribution of commands to control/configure the detector
- Distribution of a common clock

[1] The advantage of using GBT "out of the box" at this stage is the availability of HDL code for several FPGA families that will ease the initial development phase. For protoDUNE the so-called GBT wide-mode seems the best suited.

- Distribution of trigger and reception of BUSY signals

It is important to agree on which functionality may be demonstrated at protoDUNE[2] and to prioritise the implementation of different aspects according to their importance and the available manpower. It is essential that the interfaces to external systems be specified, both in terms of protocols and data formats.

2.2 Data formats

For each stream the data format shall be specified. This is a necessary pre-requisite in order to develop the FELIX based prototype and also make it compatible with any other protoDUNE DAQ or DCS solution.

In particular:

- The event format and header information shall be implemented and filled by the data sources (front-end electronics or DCS)
- FELIX is not modifying the content of the data, but needs to access data headers in order to establish the correct routing. This means that the header must contain enough information to infer the destination of the data.
- The clock, trigger and busy signals are directly treated by the FPGA on the FELIX PCIe cards, thus protocol and format must be specified, if this functionality is required. It shall be noted that presently only the TTC system is supported by FELIX: thus, integration with other clock distribution systems such as WhiteRabbit or NOVA will need dedicated manpower, as well as interaction with the ATLAS firmware experts in order to find the best way of accommodating those variants. An option also is to simply allow for the on-board FMC SMA connectors to allow to take various input TTL signals that represent (1) different trigger pulses or inhibits (vetos), along with some kind of a timing sync pulse. This may take very little development. It doesn't really represent a full trigger and timing solution, but may be a nice development stage, and is perhaps all protoDUNE requires.

2.3 Data flow

The FELIX system is inherently stateless (to support DCS irrespective of the state of the experiment) and data driven. It is necessary to establish how it will interface and integrate into the DAQ and DCS architectures.

In particular

- The DAQ software interface to the FELIX shall be data driven (destination assignment done at the FELIX) and capable of receiving and assembling

[2] The on-detector Warm Interface Board (WIB) will also have separate hardware connections for the purpose of detector control, clock reception, etc. It is important to reach an agreement with the developers of this system on whether, how and when it may be possible for them to include into the WIB firmware the capability of receiving all those data via the bi-directional GBT links and route them appropriately within the board.

individual e-link fragments. A Data Handler component will be introduced in order to allow the event builder to transparently receive data from the FELIX or RCE and adapt the FELIX output to the event builder protocol.

- The data flow between the DCS and the front-end electronics must be specified. All communications shall be asynchronous.

2.4 Hardware setup

1 Anode Plane Assembly (APA) of the protoDUNE detector will be connected to a FELIX system. An estimated data throughput of ~100 Gbps shall be sustained. . We hope that this may be arranged in a sensible manner, as in one APA or one flange per FELIX node. In fact, we calculate that one card can take one APA of continuous data. The grouping of channels and number of FELIX PCIe cards/computers needs to be established and strongly depends on the link speeds supported by the switched network that will be installed (10/40/100 Gbps). If compression is achieved in the FELIX FPGA or if only triggered data causes readout from the electronics and thus DMA'ing into the FELIX host memory the network requirements may be relaxed.

If the FELIX is also meant to be used for readout and/or control of the photo detectors (both for single and dual phase prototypes), and for possible triggering of the rest of the detector, it is important that this be specified as early as possible, in order to correctly dimension the system, and to ensure that the points listed in 2.1, 2.2 and 2.3 will be compatible also for those.

3 System architecture

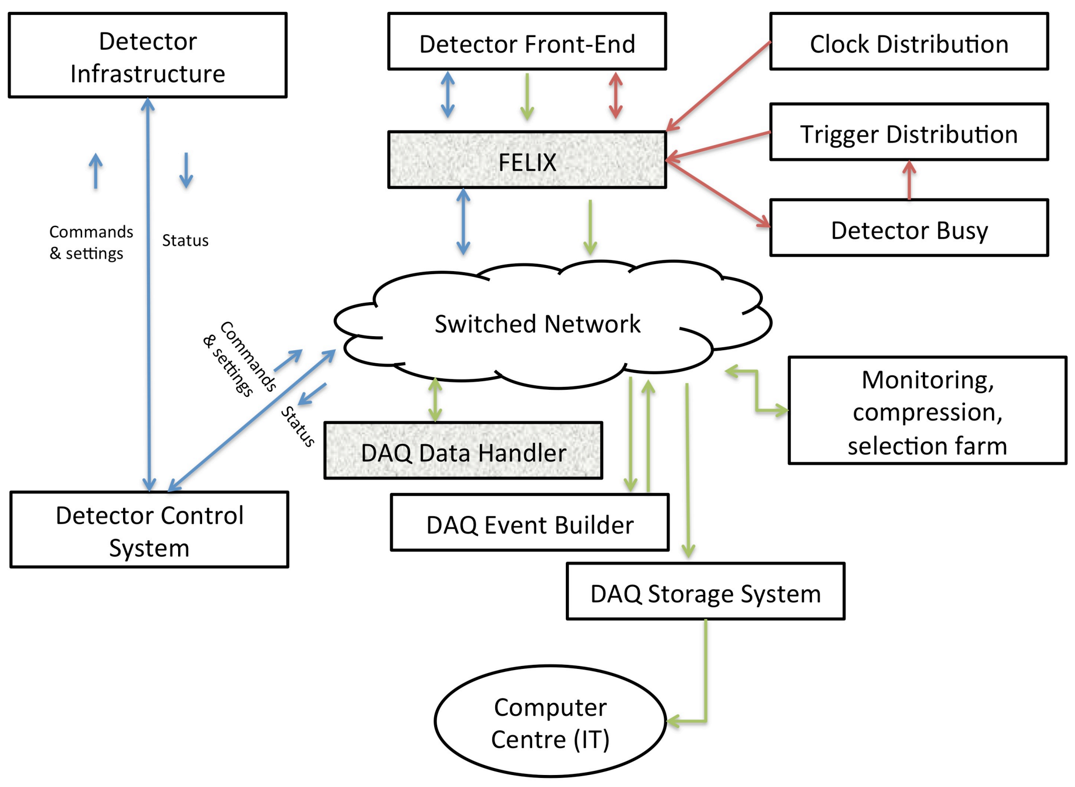

Figure 1 A high level architecture of a DAQ, DCS, timing and trigger system using FELIX.

Figure 1 shows a functional architecture of the system. In order for it to be compatible with the other solutions used for protoDUNE, it is mandatory to agree on a common system (and format) for clock distribution, trigger and busy. Also, the data format of physics data should be such that it allows FELIX to route data to the correct data handler. The data handler acts as an interface between the event builder and FELIX, and performs a stage of aggregation, and data formatting. It may also carry out data processing, such as selection and compression. Last but not least the detector control commands, configuration parameters and monitoring data that will transit through FELIX must be compatible with a solution without FELIX.

The FELIX acts as a pure router between actors (Detector Front-End, Clock/Trigger generator, Detector Control System, Data Handler) and has no notion of the content of the data. The data headers are used for the correct packets routing. Besides the FELIX and the Data Handler, all other elements in the system should be the ones developed or used anyway for protoDUNE. Physics data flow from the FELIX, through the event builder, to the PC farm and storage system. The temporary storage system eventually forwards data to the CERN computer centre.

It should be noted that, despite the fact that FELIX and the Data Handler are depicted in this sketch as separate blocks across the network, they may be tightly integrated from an implementation point of view. The separation is only a logical and functional one.

Detector control commands and configuration data flow from the DCS system through the FELIX to the Detector Front-End. Here they are processed; responses follow the reverse path as well as asynchronous detector monitoring data.

Clock and Trigger signals are fed into the FELIX card using a dedicated hardware connection and are forwarded to the Detector Front-End. Similarly a busy signal can be generated by the Detector Front-End and forwarded by the FELIX to the trigger system.

4 Tasks breakdown

4.1 A FELIX based readout system

The FELIX based DAQ shall smoothly integrate into the overall protoDUNE DAQ system. This is valid both for the data flow as well as for the control, configuration and monitoring aspects. Since the overall DAQ system for protoDUNE is still evolving, the proof of concept of the FELIX based system may use libraries and tools from other existing DAQ systems, in order to speed up the development.

The component that mostly requires development effort is the **Data Handler**: while the FELIX system can be taken from ATLAS “as is”, at least initially, the Data Handler is the element that adapts data coming out of the FELIX to the remaining DAQ system. As already stated before, it may consist of software/firmware running on independent hardware, or on the FELIX and/or event builder nodes. Several implementation choices may be explored. Initially, a standalone element could be used, for ease of implementation.

Its tasks are:

- Data aggregation (mandatory): fragments coming from the FELIX should be aggregated into super-fragments.
- Data formatting (mandatory): the super-fragments shall be framed with a format compatible with the (artDAQ) event builder expectations.
- Data forwarding (mandatory): the super-fragments shall be sent to the event builder using a common networking library (provided by artDAQ or TCPIP/UDP using the socket or boost API).
- Data compression (optional): if deemed necessary, fragments or super-fragments may be compressed.
- Data processing (optional): if desired, the Data Handler may provide a stage of event selection.

An example of high degree of integration between the Data Handler and FELIX may look like this: the server holding the FELIX cards is used for some computation and data handling. In this case we imagine a few extra ingredients beyond ATLAS mode running, which involve some R&D. Lossless, e.g. Huffman data encoding, would potentially run on the FELIX FPGA and gain a factor of roughly five compression, in this case. Further, data must be pulled from memory from which it has been DMA'd and packaged into a fragment compatible with the event builder. Hence the DAQ Data Handler is collapsed onto the FELIX server. The impact of such a solution with respect to the physically decoupled solution will need to be measured, as well as the compatibility of this approach with the other functional requirements on the FELIX (e.g. its liveness irrespective of the DAQ status).

4.2 FELIX control, configuration and monitoring

The FELIX must be stateless and not depend on the DAQ state or configuration, in order for it to be used also for detector control. Nodes that require data to receive data from the FELIX will thus dynamically subscribe to specific streams.

Monitoring data (counters and histograms) will be continuously published by FELIX in order to assess its functioning and performance. This monitoring stream should be available already in the early stages of development, as it will be an important tool for understanding the system. This is one aspect of the FELIX software that may need to be worked on within the neutrino community, if ATLAS timescales are not appropriate for the protoDUNE prototype.

Due to the uptime requirements of FELIX, its monitoring data should be made available via the DCS, and possibly streamed also to the DAQ when it is active. The exact mechanisms will be discussed in the framework of the overall DCS design.

4.3 A FELIX based DCS

The requirements for the DCS system for (proto)DUNE still need to be defined. The aim is to use the toolkit provided by the JCOP framework that has delivered and maintains detector control solutions for the LHC experiments any beyond at CERN. It's a natural choice, since this implies the reuse of software as well as a smooth integration with other infrastructure control systems (cryogenics, etc…) needed by the experiment that are handled via the same tools.

The requirements on DCS are independent of the use of FELIX. Nevertheless, if DCS data will be routed via FELIX, then this will imply the development of a OPC UA server [5] translating the DCS data structures into network packets suitable for FELIX and vice versa.

4.4 FELIX based clock, trigger, back-pressure distribution

Depending on the technologies chosen for the experiment, hardware, firmware or software development may be needed to distribute clock, trigger and backpressure signals through the FELIX system. It is premature at this stage to make any planning, but it would be nice, if the solution chosen was in line with widely used systems (e.g. WhiteRabbit), in order to profit from a wider developers community.

4.5 Initial work plan

This initial work plan only indicates an estimated amount of time required by different activities, assuming that enough skilled man power is available. It will need to be modified and adapted to reality once institutions and people will commit to the work.

Task	Activity	Priority	ID	Time estimate	T0	Details
Readout	Data from WIB to FELIX	1	1	1 month	WIB prototype available	This activity entails implementation and debugging of the GBT communication between WIB and FELIX; low level FLX software tool are sufficient to perform the complete integration.
	Data from FELIX to Data Handler	1	2	1 month	Now	A data sample should be loaded into the FELIX FPGA, go through the FELIX server and be routed according to header information to a (set of) Data Handler(s). Consistency checking in Data Handler. Data Handler implementing only mandatory functionality.
	Data from Data Handler to Event Builder	1	3	1 month	Now	Simulated FELIX fragments should be assembled and sent to an event builder node.
	Data from FELIX to Event Builder	1	4	1 month	After 2, 3	This activity should make use of multiple FELIX/Data Handler/Event Builder nodes to be fully validated ; performance measurements to be included
	Data from WIB to Event Builder	1	5	1 month	After 1, 4	This activity completes the baseline prototyping of the DAQ data flow aspects using FELIX.
	Merging Data Handler into FELIX and/or EventBuilder	4	6	2 months	Best after 2, 3	Evaluation of modularity, simplicity, performance.
	Data compression in Data Handler	3	7	1 month	After 4	Evaluation of required computing resources, compression effectiveness, …
	Data compression integrated into FELIX FPGA	3	8	2 months	Best after 5	Prototyping of algorithm implementation may start now, though integration should be pursued after having achieved a baseline system.
Run Control, Configuration, Monitoring	Integration of FELIX, Data Handler and Event Builder into control infrastructure	2	9	4 months	After 2, 3	This task requires the selection of the DAQ control framework that will be used, as well as the specification in terms of states, monitoring, streams, configuration database choice, etc... Options such as artDAQ or JCOP should be evaluated and prototyped. Finer grained activities will be established once the overall framework will be clearer.
	Graphical User interface	2	10	3 months	After 9	Design and implementation of a graphical user interface giving a clear view of the system state and functioning and allowing to operate the experiment.
Detector Control	Integration of DCS with FELIX	3	11	2 months	DCS requirements available	This task requires the selection of the DCS control framework that will be used, as well as the specification in terms of detecotr states, monitoring, streams, configuration needs, etc... In order to be in-line with the control of the experiment infrastructure we suggest to base this system on the JCOP toolkit.
	Integration of DCS with WIB (via FELIX)	3	12	1 month	Availability of WIB supporting this data path, after 11	
Clock & Trigger	Interface of clock with FELIX	4	13	6 months	Technical choice of clock distribution done	Depending on the technical choice this activity may be very manpower intense and time consuming.
	FELIX forwarding clock to WIB	4	14	2 months	WIB supporting this function, after 13	Complete debugging and assessment of clock quality coming through FELIX.
	FELIX forwarding trigger information	5	15	1 month	Encoding of trigger info established; after 14	
	FELIX forwarding backpressure signal to trigger	6	16	2 months	WIB and trigger supporting this functionality	Not clear whether this functionality will exist in protoDUNE.

5 References

[1] J.T. Anderson et al, A new approach to front-end electronics interfacing in the ATLAS experiment, http://cds.cern.ch/record/2065122

[2] J.T. Anderson et al, FELIX: a High-Throughput Network Approach for Interfacing to Front End Electronics for ATLAS Upgrades, http://cds.cern.ch/record/2016626

[3] J. Schumacher, Improving Packet Processing Performance in the ATLAS FELIX Project, http://cds.cern.ch/record/2014753

[4] JCOP, https://wikis.web.cern.ch/wikis/display/EN/JCOP+Framework

[5] OPC Unified Architecture, https://en.wikipedia.org/wiki/OPC_Unified_Architecture

[6] GBT, https://espace.cern.ch/GBT-Project/default.aspx