ATLAS NOTE

ATLAS-INT-2017-XXX

March 19, 2018

Draft version 0.84

FELIX User Manual

ATLAS FELIX Group

Contents

Revision History

Revision	Date	Author(s)	Description
0.84	19-03-2018	L. Levinson	Clearer word-oriented format for L1AInfo, Figure 30.
0.83	06-03-2018	W. Panduro Vazquez	Fix TTC endianness (to little endian).
0.82	20-02-2018	L. Levinson	Review of SCA documentation.
0.81	15-02-2018	W. Panduro Vazquez	Updates to TTC and BNL-711 appendices, fix broken link to LAr user experience doc for TTC.
0.8	09-02-2018	W. Panduro Vazquez	Integrate flx-mon docs, TTC B-channel docs, updates to software for new release.
0.71	22-11-2017	L. Levinson	Correct GBT 8b/10b mode to "not implemented".
0.7	11-11-2017	W. Panduro Vazquez	Add SCA and TTC appendices, general review
0.63	26-10-2017	J. Schumacher	Add debugging instructions for FelixCore.
0.62	08-09-2017	L. Levinson	Update TTC appendix.
0.61	08-09-2017	W. Panduro Vazquez	Implement review comments and update software descriptions.
0.60	08-07-2017	W. Panduro Vazquez	Major update - material on BNL-711 and FULL mode, rewrite of TTC and clock setup, update to ftools.
0.54	08-05-2017	J. Schumacher	Add FelixBus chapter content.
0.53	26-07-2017	L. Levinson	Added TTC item to section: Guide for Front end designers.
0.52	26-07-2017	L. Levinson	Added section: Guide for Front end designers.
0.51	06-04-2017	W. Panduro Vazquez	Update firmware distribution details, other minor fixes.
0.50	06-04-2017	J. Schumacher	Add Felixcore chapter content.
0.40	28-02-2017	W. Panduro Vazquez	Overhaul release distribution information and contact information.
0.34	20-01-2017	W. Panduro Vazquez	Minor tweaks and bug fixes.
0.33	19-01-2017	W. Panduro Vazquez	Modify VC-709 USB programming instructions.
0.32	30-11-2016	L. Levinson	Added details on the SMA cables for the TTCfx and on the TTCvx connection.
0.31	28-11-2016	J. Vermeulen	Small corrections, capitals in file names of figures now consistent with actual filenames, typesetting under Linux now OK
0.3	10-11-2016	W. Panduro Vazquez	Integrate TTC guide as an appendix, continue filling in basic tools guide, expand and organise content of user testing guide
0.2	09-11-2016	W. Panduro Vazquez	More detail to hardware guide, added firmware programming guide
0.1	08-11-2016	W. Panduro Vazquez	Added driver and software guide
0.0	26-10-2016	W. Panduro Vazquez	created

1 Overview

This document is intended to support all users of the Phase-I FELIX readout infrastructure with installation, maintenance and operation of their system. The document covers all aspects of the FELIX system from recommended hardware to firmware and driver installation and maintenance. Finally the full suite of FELIX software will be presented, including useful test tools leading up to the primary 'FelixCore' dataflow application which is intended to form the backbone of all data taking sessions. For more information users should consult the following locations for updates:

The FELIX users mailing list:

atlas-tdaq-felix-users@cern.ch

The FELIX Project Website:

https://atlas-project-felix.web.cern.ch/atlas-project-felix

The FELIX release distribution site:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/dist/

User support requests from users to the FELIX team should be made via the dedicated JIRA project:

https://its.cern.ch/jira/projects/FLXUSERS

Note: User support via SharePoint has been discontinued. Please report any broken links of obsolete material to help improve the overall quality of our documentation

1.1 Document Compatibility

This document is continuously evolving alongside FELIX firmware and software. The table below will keep track of the versions of each which should be considered covered by a given version of this manual.

Manual Version	GBT F/W Revision	FULL Mode F/W Revision	Software Release
0.8+	5317+	5327/5168	3.9.1
0.7	5317+	5327/5168	3.8.1
0.61	5317	5327	3.8
0.6	5214	N/A	3.7
< 0.6	4400, 4500	N/A	3.4.2

2 Introduction to FELIX

FELIX is a new detector readout component being developed as part of the ATLAS upgrade effort. FELIX is designed to act as a data router, receiving packets from detector front-end electronics and send it to programmable peers on a commodity high bandwidth network. Whereas previous detector readout implementations relied on diverse custom hardware platforms, the idea behind FELIX is to unify all readout across one well supported and flexible platform. Rather than the previous hardware implementations, detector data processing will instead be implemented in software hosted by commodity server systems subscribed to FELIX data. From a network perspective FELIX is designed to be flexible enough to support multiple technologies, including Ethernet and Infiniband. Given the general purpose nature of the FELIX effort, the system has also been adopted by several non-ATLAS projects. This document is therefore targetted at users both within and outside of the ATLAS upgrade effort.

2.1 FELIX Variants and Functionality

FELIX supports two different link protocols for the transfer of data to and from front-end peers. Each is supported by the same hardware platform, with separate firmware revisions both based on the same core modules.

2.1.1 Gigabit Transceiver (GBT) and the Versatile Link

The Gigabit Transceiver (GBT) chipset and associated technologies were developed as part of CERN's Radiation Hard Optical Link Project [1]. The goal was to develop a radiation hard bi-directional link for use in LHC upgrade projects. GBT provides an interface an optical connectivity technology known as the Versatile link [2], which provides high bandwidth and radiation hard transport of data between GBT end points.

The GBT transmission protocol is designed to aggregate multiple lower bandwidth links from front-end electronics components into one radiation hard high bandwidth data link (running at up to 5 Gb/s). The logical lower bandwidth links which make up a GBT link are known as E-links. The details of how E-links are supported within the FELIX project are discussed in Section 6.1.5 of this document.

The GBT protocol has been implemented both in dedicated hardware (e.g. the GBTx chip [3]) as well as directly on FPGA platforms, the latter of which has been built on for use by the FELIX project [4].

2.1.2 FULL Mode

Within the context of the ATLAS upgrade (and subsequently externally) a requirement arose for a higher bandwidth data link from detector to FELIX than was possible with GBT, which has to support radiation hardness. These newer clients did not require radiation hardness, and were able to support a protocol which could be implemented in FPGAs on both sides of the link. The resulting development is known as 'FULL mode' [5], referring to full bandwidth.

The FULL mode protocol is a implemented as a single wide data stream with no handshaking or logical substructure (i.e. no E-links). The reduced constraints mean that FULL mode links can operate at a line

transmission rate of 9.6 Gb/s, which accounting for 8b10b encoding means a maximum user payload of 7.68 Gb/s.

Note that FULL mode in FELIX is currently only implemented in the from detector to FELIX direction, as there are currently no requirements for the to detector direction. FELIX FULL mode variants therefore implement to detector links with the GBT protocol, as this is sufficient for the required payloads.

2.1.3 Propagation of ATLAS TTC Information

As well as transferring data to and from front-ends, FELIX is also required to interface with the ATLAS Timing, Trigger and Control (TTC) system. FELIX must provide TTC information both to the front-ends at full granularity, and to network peers in a reduced form. The propagation of TTC information to the front-end is performed via dedicated E-links.

3 Hardware Requirements and Setup

3.1 Recommended Hardware Platforms

3.1.1 FPGA I/O Hardware: VC-709

The hardware platform recommended for FELIX operation in detector test stands is based on the Xilinx® VC-709 Connectivity Kit [6]. This platform provides 4 optical transceivers compatible with both GBT and 'full mode' operation as well as a Xilinx® Virtex®-7 series FPGA and 8-lane PCIe Gen 3.0 interface. The TTC interface for the system is provided by the TTCfx v3 FMC mezzanine card. An image of the VC-709 board and guide to features is presented in Figure 1.

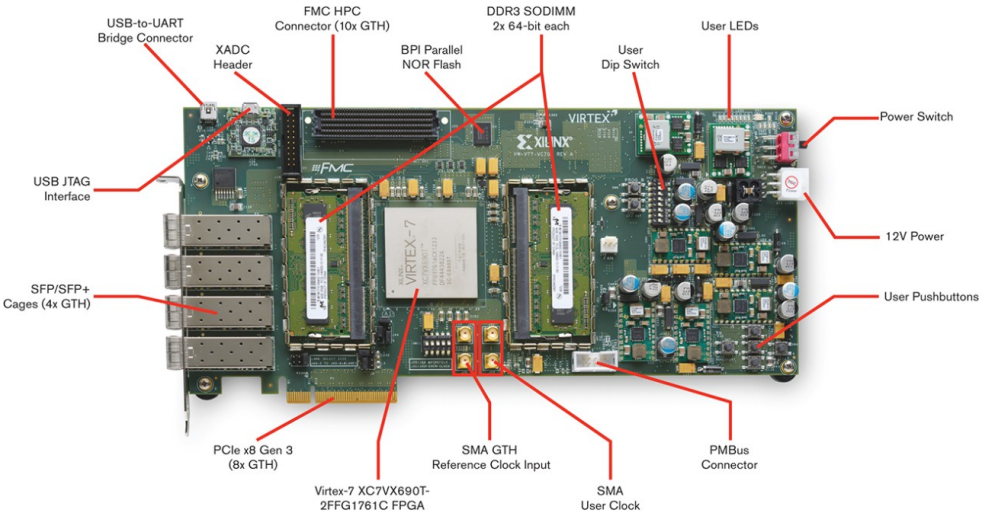


Figure 1: The VC-709 development board.

3.1.2 FPGA I/O Hardware: BNL-711

The hardware platform currently under development for the final FELIX implementation in Phase-I is a custom interface board designed by BNL, known as the BNL-711. The BNL-711 hosts a Xilinx® Kintex® UltraScale FPGA on a board capable of supporting 48 high speed optical links via MiniPOD transceivers, with a 16-lane PCIe Gen 3.0 interface. On-board clock jitter cleaning and TTC circuitry mean that no mezzanine attachment is required to connect with ATLAS clock and control systems. An image of the BNL-711 and its key features are presented in Figure 2. While the board is still under active development, prototype versions are available to detector test stands for commissioning and integration. Please contact the FELIX group for more information.

3.1.3 FELIX Host Systems

The current recommended hardware platform for a VC-709 FELIX system is based on the Supermicro® X10SRA-F motherboard [7]. The system should be populated with at least 32 GB of DDR4 RAM and

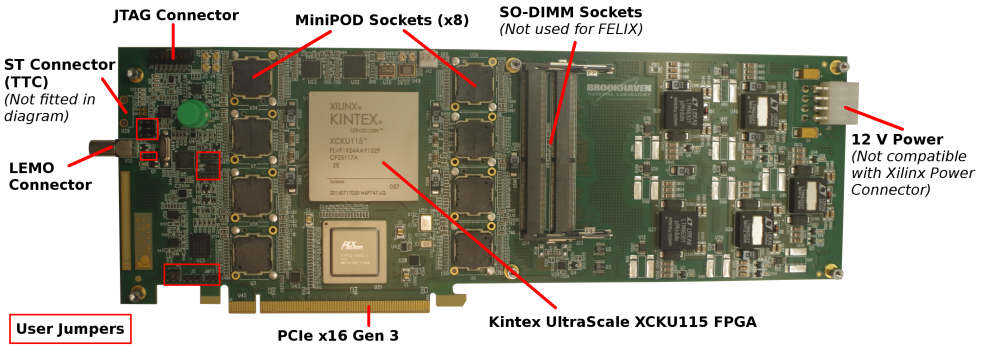


Figure 2: The BNL-711 V1.5 board.

an Intel® Xeon™ E5 family CPU (v3 or v4) with at least six real cores. Please see the motherboard manufacturer specification for more details.

3.2 Installation of VC-709

For full details regarding the VC-709 please consult the manual provided with your equipment. In terms of installing the card into a FELIX system please follow the following guidelines. The VC-709 should be installed into an 8-lane or 16-lane Gen 3 PCIe slot on the host motherboard, taking into account the need for clearance on all sides. The board must be connected to power from the system's internal ATA power supply via a custom Molex adapter provided with the board. The power socket on the board is shown on the upper right hand corner of Figure 1, labelled '12V Power'. Ensure that the power switch, just above the socket, is switched to the on position.

The FPGA aboard the VC-709 is configured via an on-board JTAG programmer, which can be connected to a mini-USB cable with the 'USB JTAG Interface' on the top left of Figure 1. A right angled mini-USB connector is recommended to minimise obstruction of the hosts case lid, although a straight cable is provided for free with your kit. Note that this has currently only been tested for USB2, which is the recommended interface. In order to be able to program the card please connect it to a convenient USB port on your host machine, or to another machine which you wish to use as a programming server. Finally, ensure that the link transceivers are safely inserted into the on-board cages.

3.3 Installation of BNL-711

The BNL-711 should be installed in a 16-lane Gen 3 PCIe slot on the host motherboard. The board must be connected to power from the system's internal ATA power supply via an 8-pin Molex adapter (of the type commonly used for graphics cards). Note that the board does not support use of Xilinx power connectors.

The BNL-711 provides a JTAG connector to which programmers can be connected for FPGA configuration. The Digilent®HS2 programmer is recommended for this purpose. Aboard the BNL-711 are a series of jumpers to permit users to reconfigure various I/O properties of the board. For a full specification of these please consult Appendix B.

3.4 Connecting to an existing TTC system

This section is only relevant to users who wish to connect their FELIX system to a ATLAS TTC infrastructure. Other users should skip this section and proceed directly to clock configuration.

3.4.1 VC-709 Only: TTCfx V3 Overview and Installation

For VC-709 systems the TTCfx mezzanine card [8] is designed to connect your FELIX card to the ATLAS TTC system as used throughout Run 1-3 operations [9]. BNL-711 systems do not require this component as the same logic is implemented on the BNL-711 itself. The TTCfx is a small FMC mezzanine card, as shown in Figure 3, which can be attached to the VC-709 via the single FMC slot on-board (top left of Figure 1).

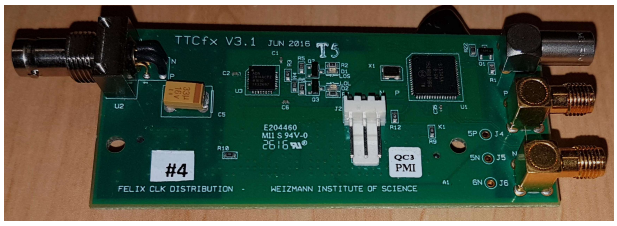


Figure 3: Image of a TTCfx V3 card.

To complete the installation, you must then connect the P and N SMA GTH Reference Clock inputs on the VC-709 (middle bottom of Figure 1) to the SMA connectors on the TTCfx V3 (P to P and N to N) via suitable SMA cables [1].

The TTCfx mezzanine card requires no specific firmware programming, and should work out of the box once connected to a TTC peer and a software configuration script is run. More detail is provided in the next section.

3.4.2 Connecting TTC and BUSY

This section assumes you are connecting a TTCvx-based system to FELIX. Notes on setting up such a system are available in Appendix A. Once set up, connect a TTC output from the TTCvx to the TTCfx V3 using a Multi-Mode fibre with ST connectors. The connector on the TTCfx V3 end is visible in Figure 3, on the upper left hand side. On the BNL-711 the ST and LEMO connectors are located on the upper left part of the board, as shown in Figure 2.

Finally, use a LEMO connector to connect the TTCfx V3 or BNL-711 to a destination for BUSY signals (as per your use case). The connector on the TTCfx V3 is visible in Figure 3 on the upper right hand side. The BUSY signal is the ATLAS standard open-collector BUSY signal, but with a weak 24 Kohm pull-up to 1.8 V to allow viewing on a 'scope.

[1] An example SMA cable is: http://eu.mouser.com/ProductDetail/Amphenol-RF/135103-01-0600/?qs=sGAEpiMZZMufBZYvsU/be%2bYZgfjb/mihYZ4wKp9N4jE=. The right angle side goes on the VC-709, to make the cable bending a bit more gentle. If you have space in your chassis, straight SMAs on both ends will do the job as well.

3.5 Configuring FELIX Clock

This section assumes you have set up the FELIX software infrastructure as in Section 5. If you have not, then please do so before proceeding.

3.5.1 Clock Source Selection

FELIX requires a clock source in order to synchronise propagation of signals both within the FPGA and to external peers. The FELIX firmware supports the use of both a received clock from an external TTC source as well as an internally generated clock for users who don't need or have access to a such a system.

Note: older firmware revisions did not support this feature, so please ensure your version is labelled with CLKSELECT to ensure compatibility.

To check your current clock selection, run the following command:

```
$ flx-config get MMCM_MAIN_OSC_SEL
```

A result of 0x1 indicates a system configured for TTC clock, while 0x0 indicates a local clock is in use. To change your clock selection run the following:

```
$ flx-config set MMCM_MAIN_LCLK_FORCE=0xN
```

For N = 0 or 1 as needed. It is also possible to select your clock source via the *elinkconfig* graphical tool. More information on this feature will be provided in Section 6.1.1.3.

Another way to view the overall clock status by running via the FELIX info tool, or `flx-info`. This can be run with no command line parameters to dump summary information for your board as follows:

```
$ flx-info
```

Clock settings can then be viewed in the 'Clock Resources' section, as shown in Figure 4.

```
Clock resources
---------------------------------
Local clock in use  : YES
Internal PLL Lock   : YES

ADN2814 TTC Status: ON
```

Figure 4: `flx-info` Clock Resources Output.

3.5.2 TTC Clock Recovery: ADN2814

Should you wish to use a TTC clock source, you must next check that your FELIX board's ADN2814 clock recovery chip [10] is functioning correctly. Non-TTC users can skip this section.

The overall status of your ADN2814 is reported in the Clock resources report from `flx-info` as show in Figure 4. For more detail on the chip's status, run with the following extra parameter:

```
$ flx-info ADN2814
```

If your chip is functioning correctly, and you have a TTC system connected, you should see output matching Figure 5.

```
TTC Status: ON
Loss of Signal Status: 0
Static Loss of Lock:   0
Loss of Lock Status:   0
```

Figure 5: `flx-info` ADN2814 status output.

If the output differs (e.g. if you see a loss of lock reported) please check your connections before resetting the ADN2814 using the following:

```
$ flx-reset ADN2814
```

3.5.3 Clock Jitter Cleaning

Whether you are using an internal or external clock, the signal must be cleaned to minimise jitter and ensure stable performance. FELIX uses one of two dedicated chips for jitter cleaning depending on your clock source and hardware.

TTC clocks should be cleaned by the *Si5345* chip [11], which is hosted by the TTCfx V3 for VC-709 systems as well as on-board the BNL-711. Non-TTC clocks can also be cleaned by the Si5345, but for those who don't have a TTCfx V3 the VC-709 also hosts a different cleaning chip, the *Si5324* [12], which offers sufficient jitter correction for the non-TTC case.

Note: the Si5324 is currently only supported with a dedicated firmware build for those wishing to connect optical links to FELIX using the FULL mode protocol. Users of GBT must have a Si5345-based system. Should you wish to use the Si5324 please make sure to check the filename of the firmware tarball provided on the FELIX firmware distribution site to ensure the name of the cleaner is present.

Whichever your use case, your FELIX card must be configured to the correct jitter cleaner in order to function correctly. This can be achieved using the `flx-init` command line application as follows

```
$ flx-init -T <N>
```

For Si5324 use N = 1, for Si5345 use N = 2.

Note: you will have to redo this jitter cleaner initialisation step each time you change FELIX clock source to maintain normal operation.

To check the status of your jitter cleaner, use the following command:

```
$ flx-info <cleaner name>
```

3.6 Connecting and Initialising Optical Links

Assuming you have set up your FELIX clocks specified above for your use case, set up the FELIX software environment as described in Section 5 and programmed the FPGA aboard your VC-709 or BNL-711 as described in Section 4 you are now nearly ready to attempt to connect the system to a peer via optical link using either GBT or FULL mode protocols.

The first step to bringing up your links is to connect your fibres to the transceivers aboard the VC-709 or BNL-711, ensuring not to place excessive strain on them. Once the connectors are properly seated, you can check the physical status of your links.

3.6.1 Physical Link Layer Status: VC-709

In order to check the status of your physical connections for a VC-709 (which are SFP based) run the following:

```
$ flx-info SFP
```

Look for the lines marked 'Link Status' in the output as per Figure 6

```
                1    2    3    4
---------------------------------
Link Status   | Ok   Ok   Ok   Ok
```

Figure 6: `flx-info` VC-709 SFP physical status output.

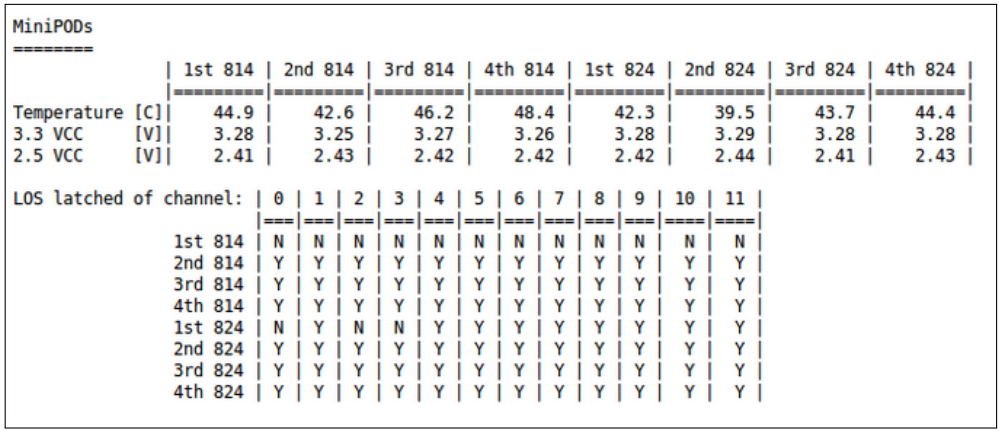
3.6.2 Physical Link Layer Status: BNL-711

In order to check the status of your physical connections for a BNL-711 (which are MiniPOD based) run the following:

```
$ flx-monitor POD
```

There will be many lines of output, but you should check the section labelled 'MiniPODs' as shown in Figure 7.

If your physical link is working correctly you should see loss of latch status 'N' for the relevant MiniPOD, where 814 corresponds to Tx and 824 to Rx PODs respectively. For a physical map of MiniPOD locations please consult Appendix B.



```
MiniPODs
========
                | 1st 814 | 2nd 814 | 3rd 814 | 4th 814 | 1st 824 | 2nd 824 | 3rd 824 | 4th 824 |
                |=========|=========|=========|=========|=========|=========|=========|=========|
Temperature [C]|    44.9 |    42.6 |    46.2 |    48.4 |    42.3 |    39.5 |    43.7 |    44.4 |
3.3 VCC     [V]|    3.28 |    3.25 |    3.27 |    3.26 |    3.28 |    3.29 |    3.28 |    3.28 |
2.5 VCC     [V]|    2.41 |    2.43 |    2.42 |    2.42 |    2.42 |    2.44 |    2.41 |    2.43 |

LOS latched of channel: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
                        |===|===|===|===|===|===|===|===|===|===|====|====|
                1st 814 | N | N | N | N | N | N | N | N | N | N |  N |  N |
                2nd 814 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
                3rd 814 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
                4th 814 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
                1st 824 | N | Y | N | N | Y | Y | Y | Y | Y | Y |  Y |  Y |
                2nd 824 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
                3rd 824 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
                4th 824 | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  Y |  Y |
```

Figure 7: `flx-monitor` BNL-711 MiniPOD physical status output.

3.6.3 Logical Link Layer Initialisation

Once you have established a successful physical connection, the next step depends on your choice of logical protocol. If you are connecting with the FULL mode protocol your logical links should then come up immediately. You should therefore be ready to attempt to transfer data to FELIX.

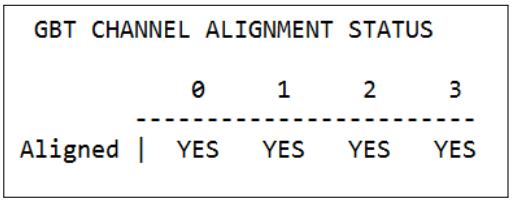
If you are connecting with GBT you will need to train the links to bring them up by running the following:

```
$ flx-init
```

This should run reporting no errors. You can then print the status of your GBT links with:

```
$ flx-info GBT
```

The results should match Figure 8.



```
 GBT CHANNEL ALIGNMENT STATUS

             0     1     2     3
          -------------------------
Aligned |   YES   YES   YES   YES
```

Figure 8: `flx-info` GBT status output (VC-709 version, a BNL-711 can have up to 24 channels displayed).

If this looks correct your GBT links should now be fully operational. Before attempting to transfer GBT data please ensure you have followed the guide in Section 6.1 for details on how to configure your E-links.

4 Firmware Releases and Programming

4.1 Firmware Distribution Protocol

4.1.1 Release Announcements

FELIX firmware (and software) releases will be announced on the following e-group:

atlas-tdaq-felix-users@cern.ch

Please subscribe to this group to stay up to date with the latest updates. All new releases will include a detailed change list and reference to the associated version of this user manual.

4.1.2 Firmware Distribution Site

Tarballs of firmware releases (containing both .bit and .mcs files) are made available via a dedicate web page:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/dist/firmware

Versioning information is available in the on-site 'bitfiles_change_log.md', please download the latest version as indicated.

Note: all recent firmware revisions are labelled 'CLKSELECT' to indicate that they support both dual TTC and local clock sources. Older revisions were dedicated to one clock or another, but these should now be considered deprecated. Please upgrade to a newer revision if you have such a version.

4.2 Firmware Programming

The FPGAs aboard both the VC-709 and BNL can be programmed directly via a JTAG interface using the Vivado™ software suite [13]. For the VC-709 this method also makes possible to program the on-board FLASH ROM. A configuration programmed into the FPGA directly will be lost if the machine is switched off, whereas a configuration programmed in the FLASH will persist. This will make it possible to retain the desired programming state of the card e.g. if transported. This section will describe how to program the card using all available methods.

4.2.1 JTAG Connectivity

The VC-709 comes with an on-board JTAG programmer, accessible via USB, as described in Section 3. The BNL-711 does not have an on-board programmer, and as such you will need to acquire a USB-accessible programmer. The FELIX developers recommend the Digilent®HS2 for this purpose.

4.2.2 Setting up the Vivado™ Suite

Specific installation instructions for the Vivado™ suite are provided with your development kit. Note that the instructions in this section are compatible with the 2014, 2015 and 2016 releases of the suite. We recommend you install the software locally on the PC you wish to use as your programming server. This should be connected to your VC-709 in your FELIX host via USB as described in Section 3.2. When you first connect your system via USB you will need to run a Xilinx® setup script to configure the bus properly (path may vary depending on product year):

```
$ source Xilinx/Vivado/2016.4/data/xicom/cable_drivers/lin64/digilent/install_digilent.sh
```

The Vivado™ environment can be started with the following commands:

```
$ source Xilinx/Vivado/2016.4/settings64.sh
$ vivado &
```

You will then be presented with the Vivado™ splash screen, where you should select 'Open Hardware Manager' as shown in the red box in Figure 9.

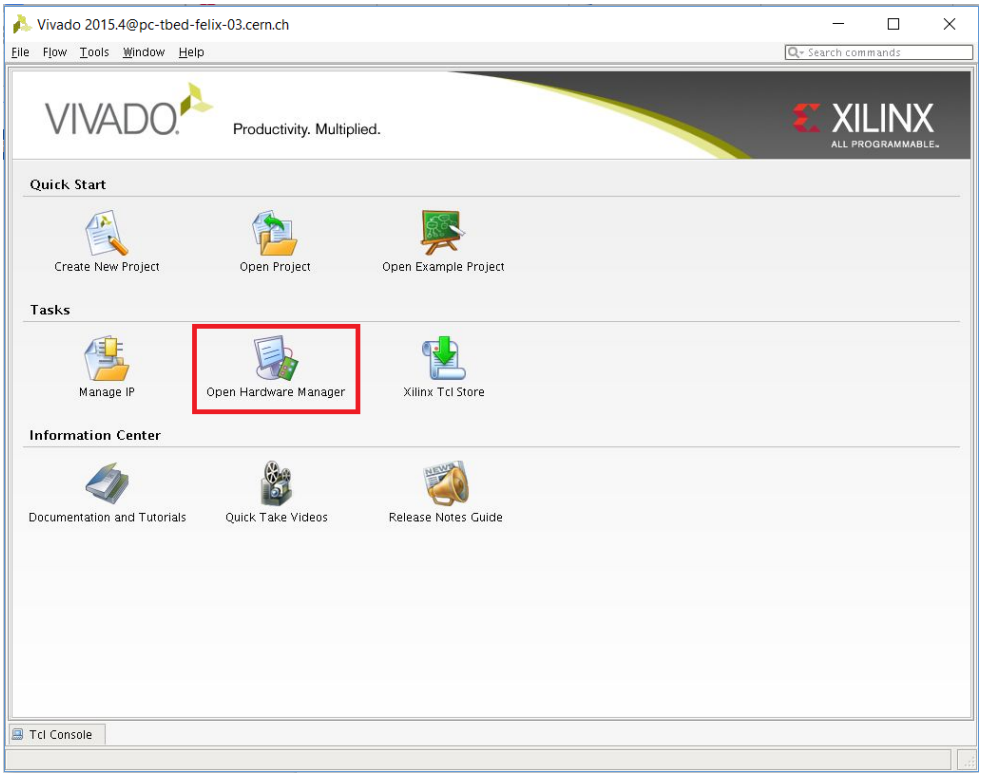


Figure 9: Vivado™ Splash Screen.

From the hardware manager select 'Open Target' on the top left as shown in Figure 10 and choose 'Open New Target'.

From this point, select 'Next' on the following screen and 'Connect to Local Server' after that, once again press 'Next'. This should bring you to the hardware list. On this screen select the FPGA on your VC-709 or BNL-711 from the uppermost list (if you have only one board there should be only one entry, if not,

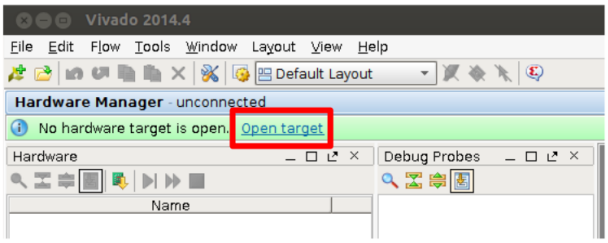


Figure 10: Vivado™ Hardware Manager.

find yours in the list by name). The screen you will see is shown in Figure 11. Once you have found your FPGA and selected it press 'Next' on the bottom right and 'Finish' on the following screen.

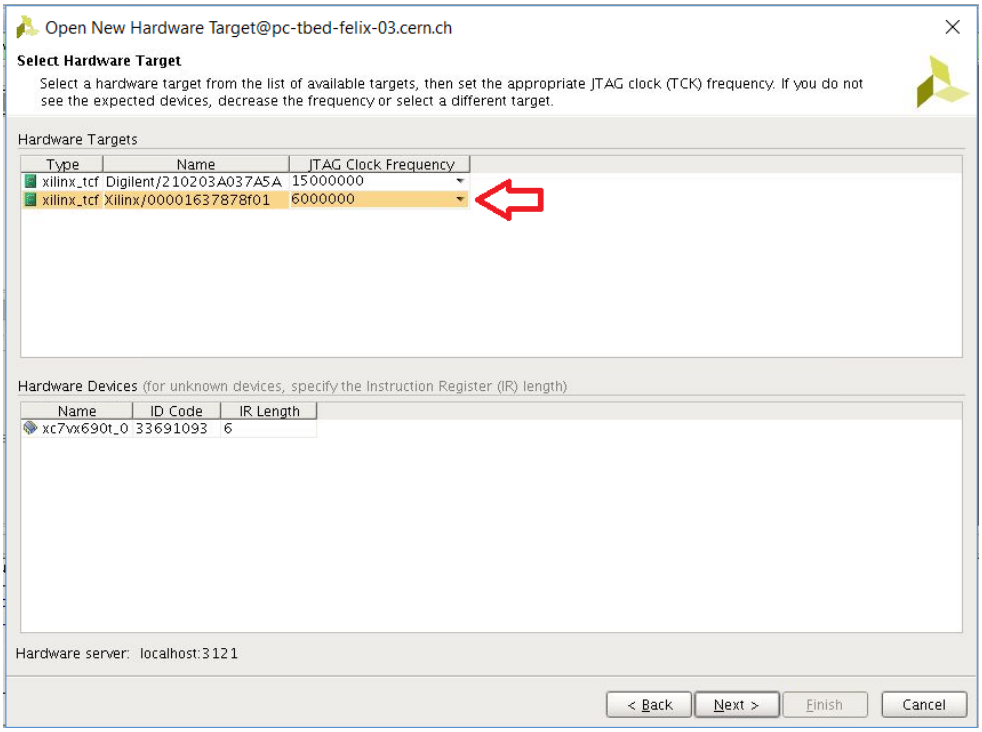


Figure 11: Vivado™ Target Selector with VC-709's Virtex7 FPGA selected, as indicated by red arrow (FPGA ID will vary from model to model). The BNL-711's Kintex Ultrascale FPGA will appear as $xkcu115_0$

From here, you will be taken to the main programming interface, as shown in Figure 12. You are now ready to program your FPGA or FLASH.

4.2.3 Programming the FPGA Directly

To program an FPGA directly, select it from the device list on the main programming window (as shown in Figure 13, right click and select 'Program Device'.

You will now be asked to select a .bit file as shown in Figure 14. This is available in the firmware release tarball as specified at the start of this chapter. You do not need to select a debug probes file. Once a file

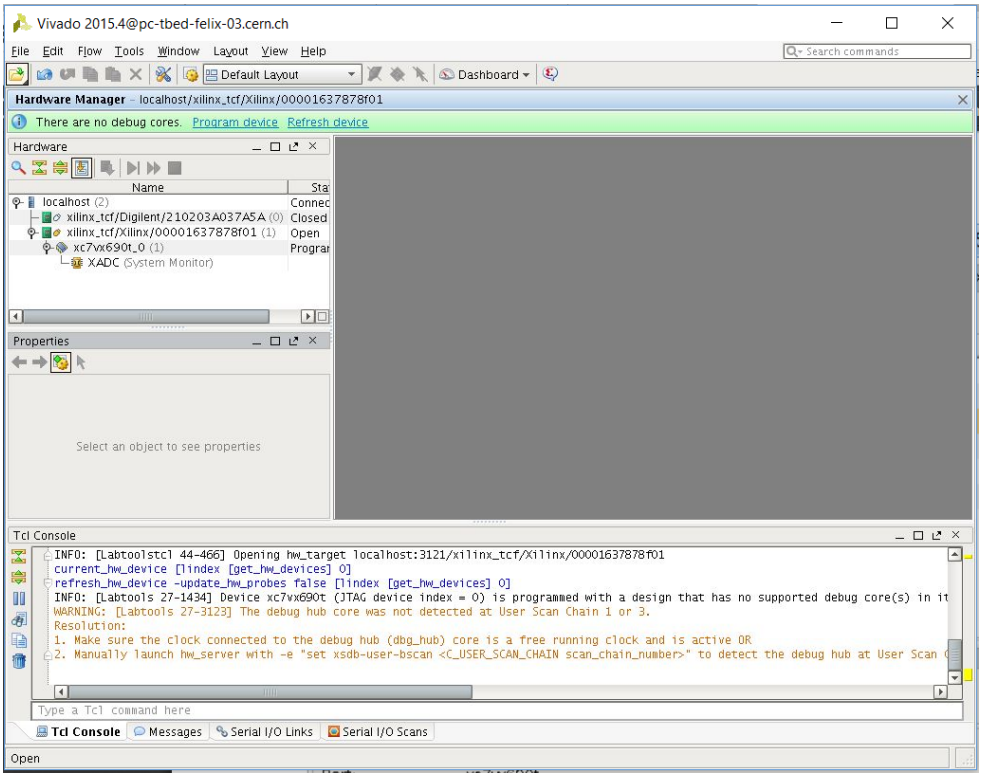


Figure 12: Vivado™ Programming Interface.

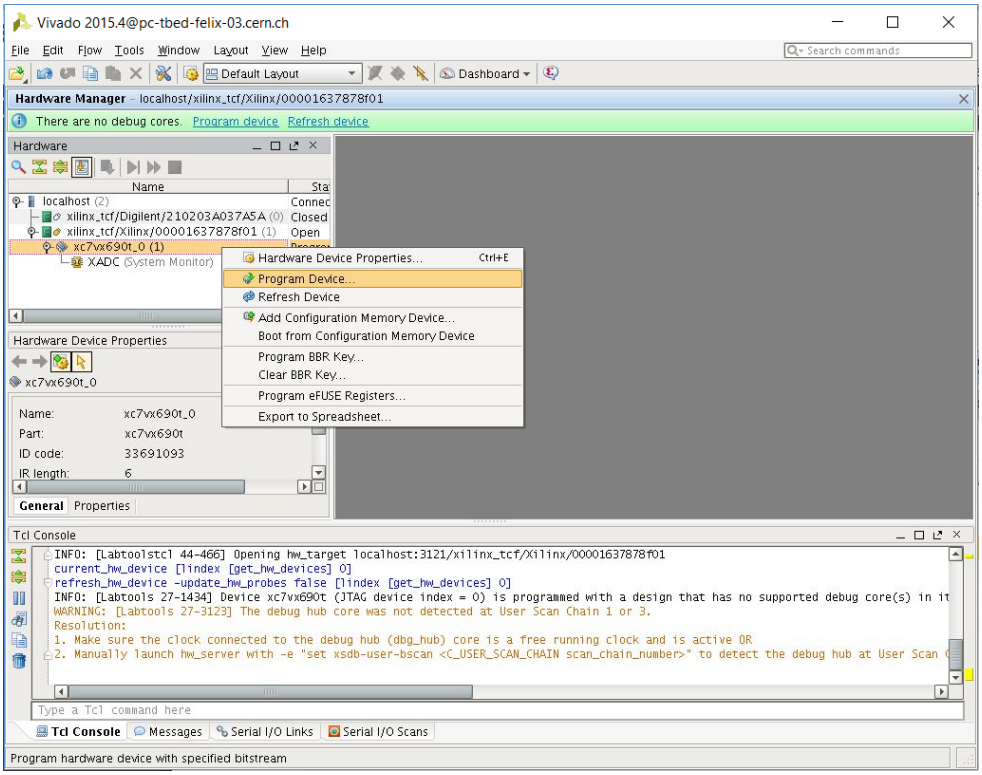


Figure 13: Selecting Device to Program.

has been chosen, select 'Program' on the bottom right to write the file to the FPGA. Once complete your FPGA should now be fully reprogrammed.

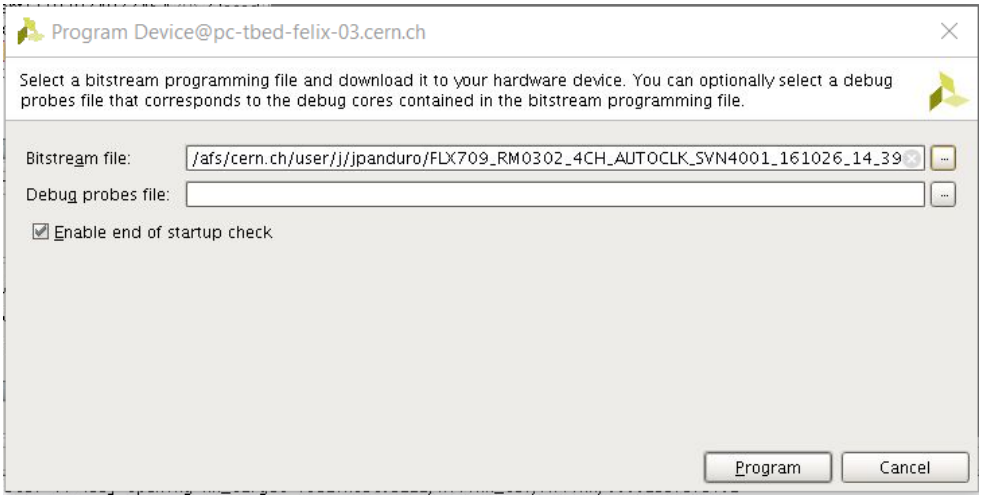


Figure 14: Selecting Bit file to Program.

4.2.4 Programming the FLASH ROM (VC-709)

To program the FLASH ROM start once again from the main programming window. Find and right click on your FPGA and select 'Add Configuration Memory Device' in the list, as shown in Figure 15

From here you will be taken to the a dialog requesting that you select the memory device you wish to program. On the VC-709 this will typically be a Micron memory device with given parameters. To find it quickly enter the criteria demonstrated in Figure 16 and select the device as shown. Look for the device with alias '28f00ag18f'.

Once selected, press 'Ok' on the bottom right and 'Ok' again on the following window asking 'Do you want to program the configuration memory device now?'. On the subsequent dialog, choose the .mcs file you wish to program (provided with your firmware release) as shown in Figure 17. Select 'Ok' at the bottom to program the FLASH. Once complete your card should be programmed with a non-volatile firmware installation that will survive loss of power to the host.

4.2.5 Programming the FLASH ROM (BNL-711)

FLASH programming for the BNL-711 is done via means of the `fflash` application, which is provided as part of the FELIX software suite. Please consult the instructions provided in Section 6.5.7. Note that the BNL-711 has 4 different FLASH sectors which can be programmed. The board will by default come up from powercycle loaded from the sector specified by the jumper configuration described in Appendix B. Please ensure you program the correct sector in order to see the expected image loaded.

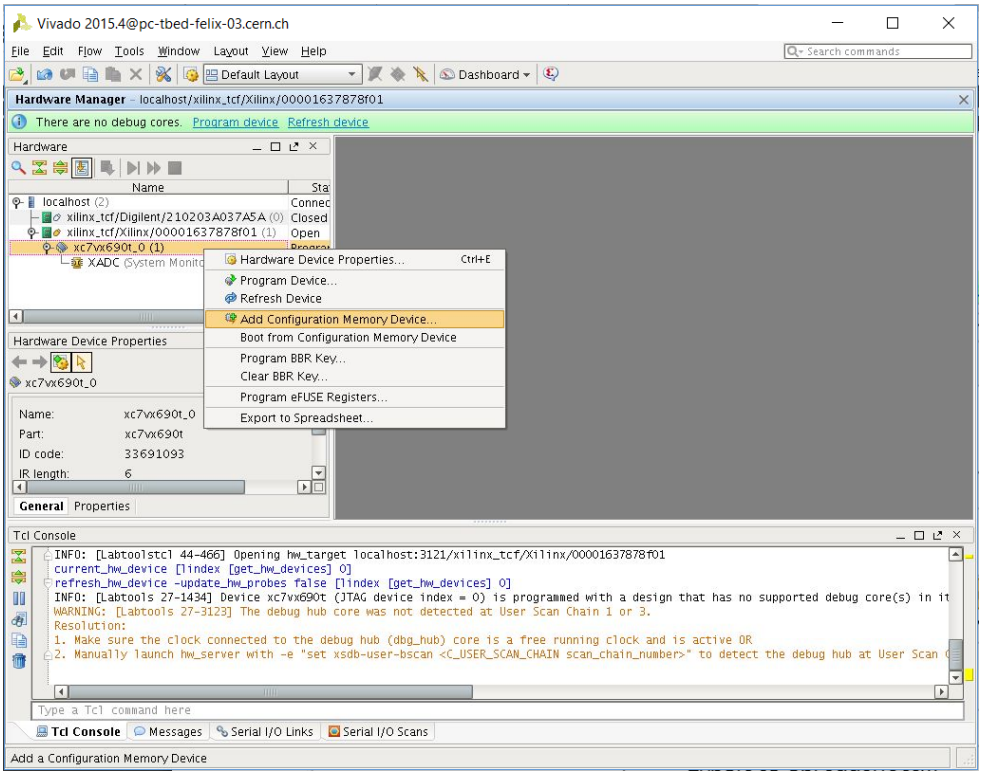


Figure 15: Select Vivado™ Flash Programming Dialog.

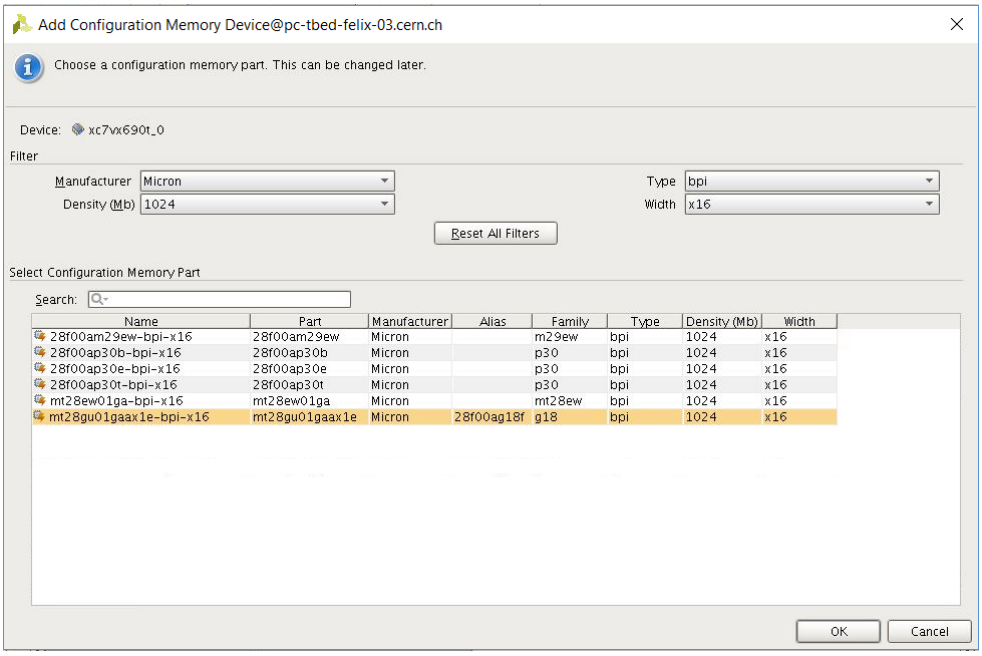


Figure 16: Memory Device Selection Interface.

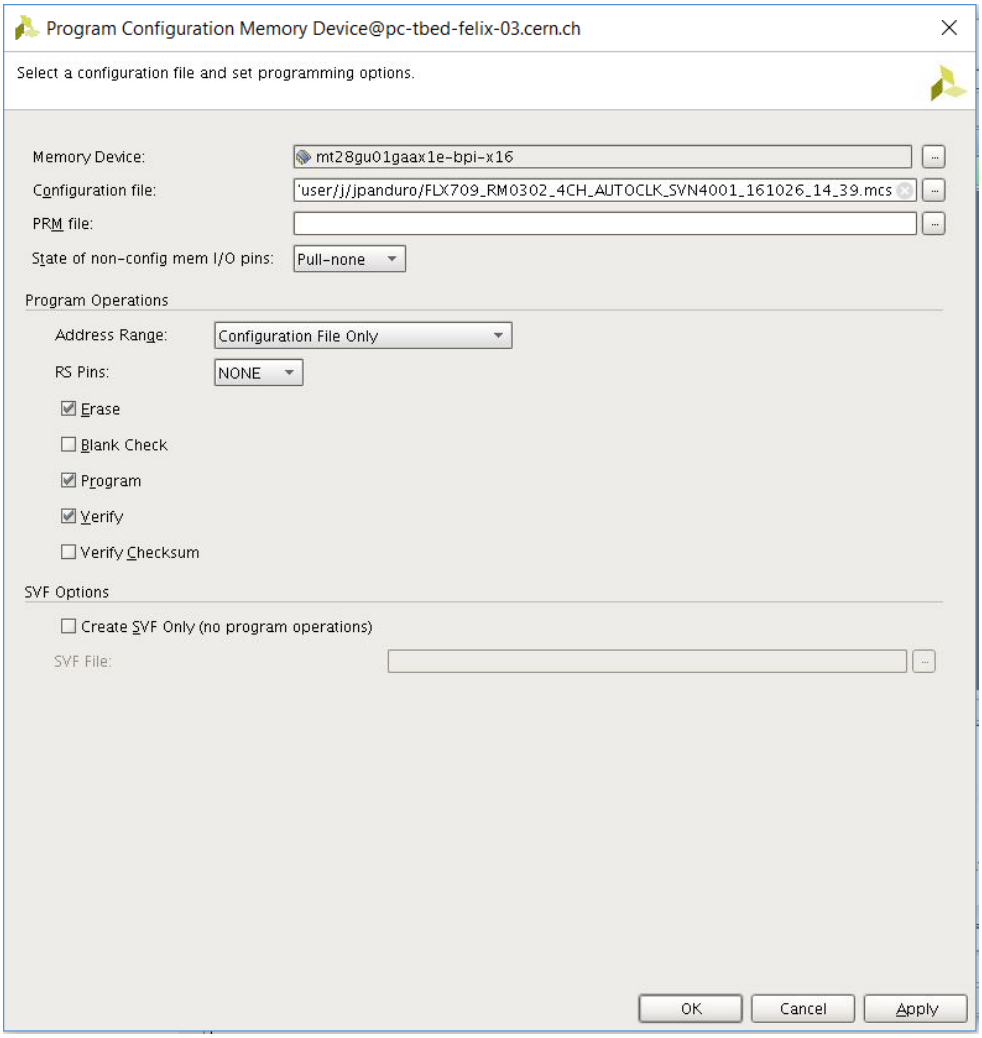


Figure 17: Selecting .mcs file to program.

4.2.6 Enabling new FPGA Configuration

If you have programmed our FPGA directly, please soft reboot your machine to pick up the new configuration. For changes to the FLASH ROM a full powercycle may be needed to pick new firmware, unless you have manually programmed the FPGA from FLASH as described in 6.5.7. In the latter case a soft reboot will be sufficient.

4.2.6.1 PCIe hotplug procedure

Should you wish to avoid rebooting your machine (assuming a powercycle isn't required) it is possible to rescan the PCIe bus re-synchronise with the new firmware image. Note that this procedure hasn't been fully validated, and may produce inconsistent results. It should only be attempted if a reboot is prohibited. First, remove the device from the bus list as follows (root privileges needed):

```
$ echo 1 > /sys/bus/pci/devices/0000:<bus ID>/remove
```

(where you get the bus ID of the device from lspci)

Then, rescan the bus to bring the device back with:

```
$ echo 1 > /sys/bus/pci/rescan
```

It is also recommended that you restart the FELIX driver at this point to pick up the new image:

```
$ ./etc/init.d/drivers_flx restart
```

.

5 Software Distribution and Installation

5.1 Software Distribution Protocol

5.1.1 Pre-requisites

FELIX software is currently only formally supported for systems using the SLC6 operating system. Stable functionality under CentOS7 is not guaranteed, but the intention is to move to more formal support in the near future.

5.1.2 Release Announcements

FELIX software (and firmware) releases will be announced on the following e-group:

atlas-tdaq-felix-users@cern.ch

Please subscribe to this group to stay up to date with the latest updates. All new releases will include a detailed change list and reference to the associated version of this user manual.

5.1.3 Release Distribution Site

The main distribution mechanism for new software releases is via a dedicated web page:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/dist/software

Here users will be able to find the latest firmware and software. The newest recommended version is marked in all cases. Installation instructions for the software suite and driver can be found below.

5.1.3.1 FELIX Driver

The latest version of the FELIX driver is available on the distribution site within 'software/drivers'.

5.1.3.2 FELIX Software Suite

The latest version of the FELIX software suite is available on the distribution site in within 'software/apps'.

5.2 Software Installation Instructions

5.2.1 Driver RPM Installation Instructions

5.2.1.1 DKMS

The FELIX driver makes use of 'Dynamic Kernel Module Support' (DKMS) to automatically track kernel changes once installed. Users should therefore only need to change their installation if a new version of the driver itself is released.

5.2.1.2 Removal of Existing Driver Installations

In order to update the FELIX driver it will first be necessary to remove any existing driver installations from your system. To do this please follow the procedure outlined below. You will require superuser privileges in order to perform the driver de-installation itself and subsequent cleanup.

To check if a driver is already installed issue the following command:

```
$ rpm -qa | grep tdaq
```

If a driver rpm is installed you'll see a response along the lines of:

```
$ tdaq_sw_for_Flx-1.0.6-2dkms.noarch
```

To remove the driver do the following (substituting 'filename' for the results of the search in the previous step):

```
$ rpm -e filename
```

Once this operation is complete you will be in a position to install the latest FELIX driver.

5.2.1.3 Installation of New Driver

To install the FELIX driver RPM, run the following command (superuser privileges required):

```
$ yum install tdaq_sw_for_Flx-1.0.6-2dkms.noarch.rpm
```

(this should take 1-2 minutes to complete, due to the need to compile the driver for your kernel as per the DKMS framework)

Once the driver is installed you should start it as follows (as superuser):

```
$ ./etc/init.d/drivers_flx start
```

Once started you can check the status of the card using:

```
$ cat /proc/flx
```

You should see output along the lines of Figure 18 (will vary depending on your firmware version).

```
FLX driver for release tdaq710_for_felix_1.0.6 and distributed with driver RPM 1.0.6

Debug                       = 0
Number of cards detected    = 1

Card 0:
Reg Map Version             : 3.7
Build revision (SVN version): 5214, Date:  2-7-2017, time     : 0h4
Number of descriptors       : 8, Number of interrupts        : 8
Interrupt count |      0 |      0 | 798595 |      0 |      0 |      0 |      0 |      0 |
Interrupt flag  |      1 |      1 |      0 |      1 |      1 |      1 |      1 |      1 |
Interrupt mask  |      0 |      0 |      0 |      0 |      0 |      0 |      0 |      0 |
MSIX PBA        00000000


The command 'echo <action> > /proc/flx', executed as root,
allows you to interact with the driver. Possible actions are:
debug   -> Enable debugging
nodebug -> Disable debugging
elog    -> Log errors to /var/log/message
noelog  -> Do not log errors to /var/log/message
```

Figure 18: Example output from /proc/flx

5.2.2 Installation of FELIX Software Suite

The FELIX software release is available pre-compiled as a tarball which can be installed anywhere and then set up for use by running a command line script. Each user can download their own version, or the release can be installed centrally and the location of the script shared with users.

To unpack the tarball, run the following command:

```
$ tar -xvzf <filename>
```

Once unpacked, a setup script must be run to enable access to all libraries and binary files. The script can be run as follows from the release base directory:

```
$ source felix-03-04-02/x86_64-slc6-gcc62-opt/setup.sh
```

This script will need to be run with every new session, or added to the environment setup procedure. Once complete you should have access to all FELIX software. In the next section we will describe how to test your installation to verify full functionality.

6 Basic Tools

The FELIX software suite comprises both high and low level tools. At the highest level, the FelixCore application is responsible for communication and bulk dataflow in a full slice system. At a lower level, the suite provides a number of tools, both command line and GUI based, to facilitate system configuration and testing. This chapter will describe these low level tools such that users will be able to effectively communicate with, configure and test their system. If you are looking to set up a full system slice with data output to a network please consult Section 7, which describes the FelixCore Application and NetIO library. This section assumes that you have set up your FELIX software environment as described in Section 5. None of the tools in this section should require superuser privileges to run. All tools presented below work in both GBT and FULL mode, and for VC-709 and BNL-711, unless otherwise stated. Where special parameters are needed to distinguish modes this will be indicated. A quick reference for all tools to be covered in this section is presented in Table 2.

Note: the FELIX software suite contains a number of tools which are considered for developer use only. All tools which are rated for use by front-end users are listed in this document. Use of any other software is not recommended unless asked to do so by a FELIX developer.

6.1 E-link Configuration with elinkconfig

Before FELIX can be used to transfer data its input and output links must be configured. The link configuration for a given FELIX card can be accessed and modified using the 'E-link configurator' application, or '`elinkconfig`'. This is a GUI based tool which displays the current configuration state for all links associated with a given card. The tool supports both GBT and FULL mode.

Note: the link configuration must be manually refreshed every time a FELIX FPGA is reprogrammed, including power-cycling of a host!.

To run `elinkconfig` application issue the following command:

```
$ elinkconfig
```

From here you will reach the main configuration panel as shown in Figure 19

The elinkconfig interface is split into three main areas. At the top there are two control bars to set FELIX card parameters, open/save configuration files as well as link selectors. The left main panel displays the from front-end to FELIX/host configuration for the selected link.

6.1.1 Global Panel

The elinkconfig global panel, shown in more detail in Figure 21 provides the top level interface for the tool. From there it is possible to select which FELIX card within your system you wish to configure, which link within that card, which link mode, as well as a number of other configuration properties. It is also possible to open previous configuration files, save new ones, and read the current configuration from the selected FELIX card.

Table 2: List of all recommended user tools. For more information on each please click the tool name to visit the dedicated section of this document.

Low Level Tools	
flx-info	View FELIX hardware and firmware status information
flx-config	View and modify low-level firmware parameters
flx-init	Initialise FELIX, as well set as low level GBT and clock/jitter cleaning parameters
flx-reset	Reset FELIX or specific component
flx-monitor	Status information for LTC2991 [14] devices about a BNL-711
Dataflow Tools	
fdaq	Receive data from FELIX and save to files or perform sanity checks
fupload	Upload data through FELIX to a front-end E-link
FELIX Configuration Tools	
elinkconfig	GUI for link and data generator configuration.
felink	Calculate link IDs given inputs with differing formats.
fereverse	Reverse the endianness of data passing through an E-link.
fgpolar	Switch 0/1 polarity of all data coming or going through a specific GBT link.
feconf	Upload link and/or data generator configuration to FELIX from the command line.
femu	Control FELIX data generators.
feto	Control FELIX timeouts (global, TTC and link data, a.k.a 'instant timeout').
ffash	Command line programming tool for FLASH ROM modules in BNL-711 card.
General Debugging Tools	
fcheck	Perform configurable sanity checks on data from a file or dump selected data blocks to screen.
fedump	Receive data from FELIX and dump it to the screen.
fec	Debug control and communication with GBT-SCA chip
fuptest	Upload data through FELIX to multiple front-end E-links.
fplayback	Load data through FELIX to a front-end E-link (or links) and expect data to return via loopback. Verify returned data.
Remote Communication and Configuration Tools	
fic	Read or write GBTx chip registers via the GBT-link IC-channel
fgconf	Read/Write GBTx registers via GBT-SCA i2c channel.

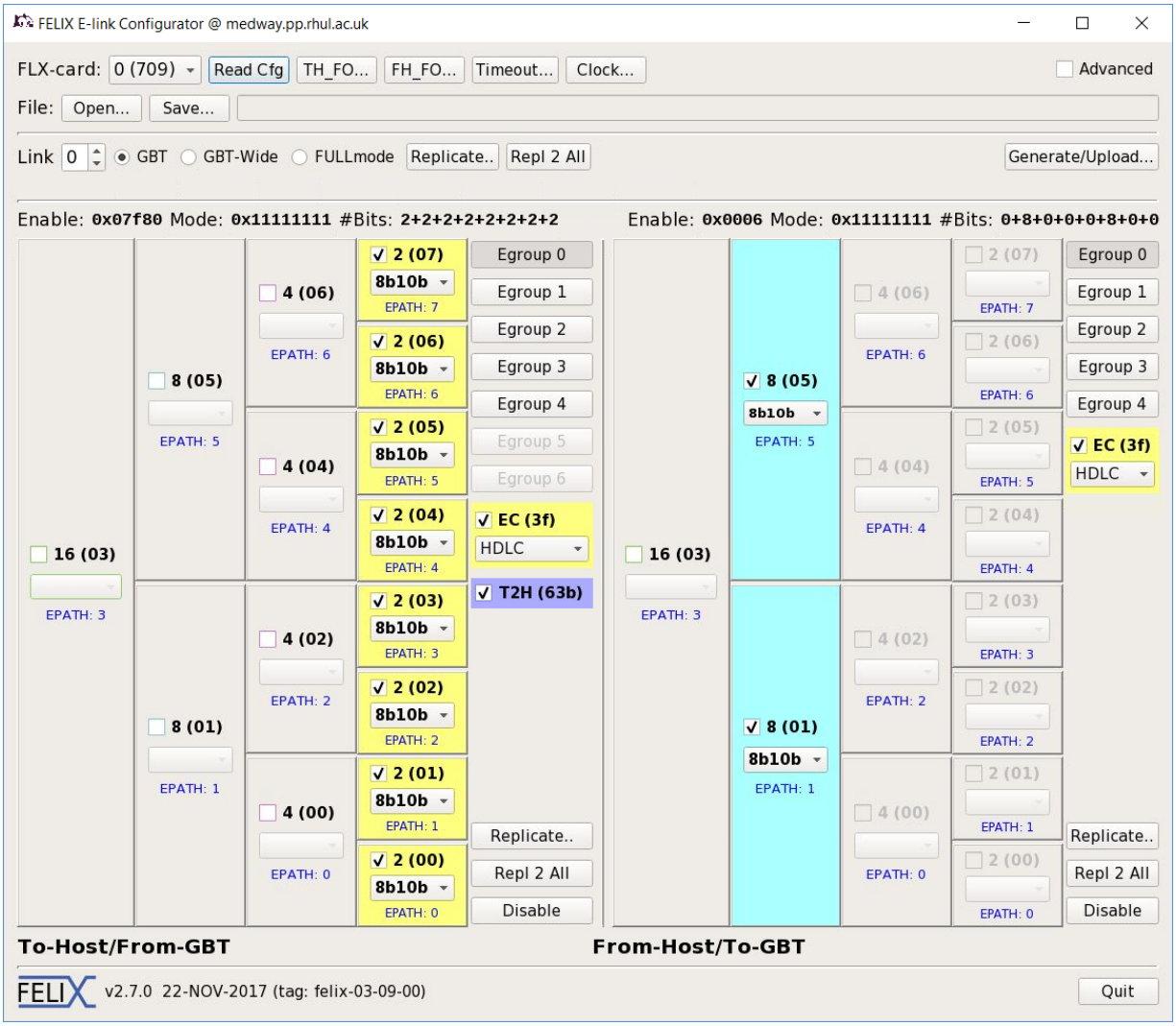


Figure 19: Main panel - elinkconfig

This panel also contains an advanced developer feature allowing you to select the maximum chunk size for a given E-link width - users are recommended to avoid changing these settings as they may cause unexpected behaviour.

From the global panel it is possible to access a number of sub-panels, as indicated in Figure 21. These give access to more advanced configuration options, details of which are presented below.

6.1.1.1 Data Path Fan Out Selectors: TH_FO and FH_FO

FELIX operates two separate data generators within its firmware, one attached directly to the data path going to the host, and one attached to the path going towards the front-end. While the generators are attached, they have mutually exclusive access to the data path with regular non-emulated data in both directions. To avoid the two data types colliding only one type may access the path at a time. The fan out selectors control this access by ensuring that only internally emulated data or external data can

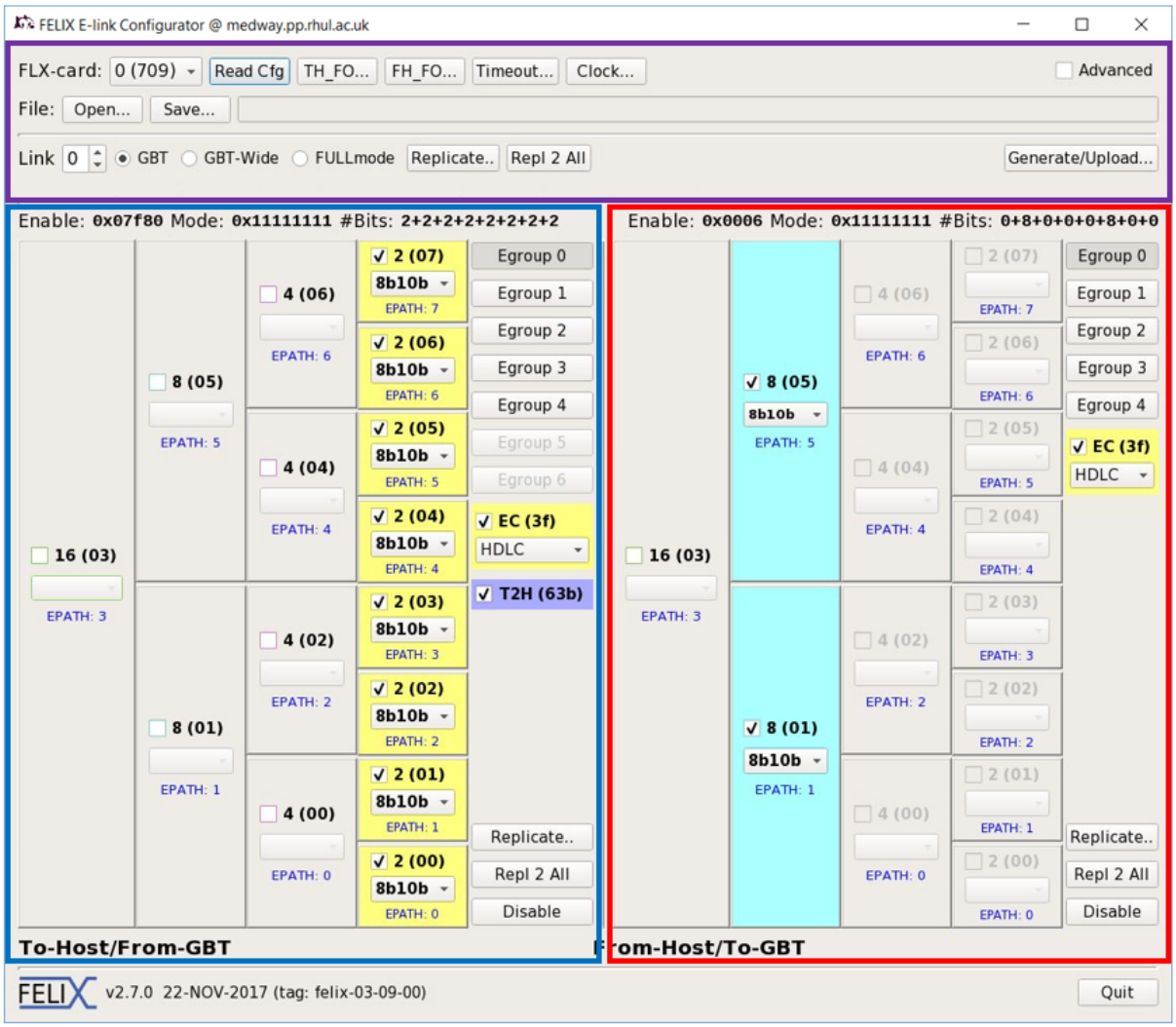


Figure 20: elinkconfig panel split. The uppermost panel (purple box) controls global settings and GBT selection. The left main panel contains the E-link configuration for the from front-end to host direction, the left main panel the from host to front-end direction

be configured to pass at any one time. Most FELIX applications are able to configure these selectors automatically, but for the purposes of user testing it may be necessary to set these values manually. The selectors are accessed via the TH_FO (to host) and FH_FO (from host) buttons in the global panel. The resulting dialogs are presented in Figure 22.

In order to switch the selector value simply open the required dialog and click on the link number you wish to toggle. A link displayed with its number alone is set to external data, if a link is displayed with its number plus 'E' it is in emulation mode. It is also possible to set/unset all values using the 'All' and 'None' buttons provided. Note that changes made in this dialog are immediately propagated to the FELIX card in question once you select 'OK'..

In some cases a user may wish to prevent other applications from automatically changing these settings. For example, if a specific link is nominated for TTC information transfer it may be convenient to fix this

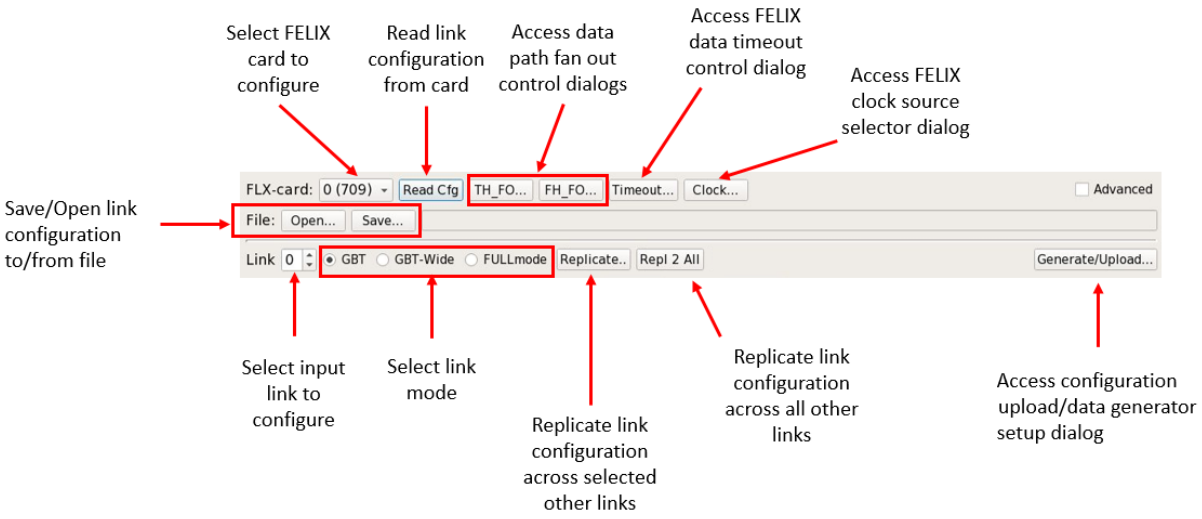


Figure 21: elinkconfig global panel.

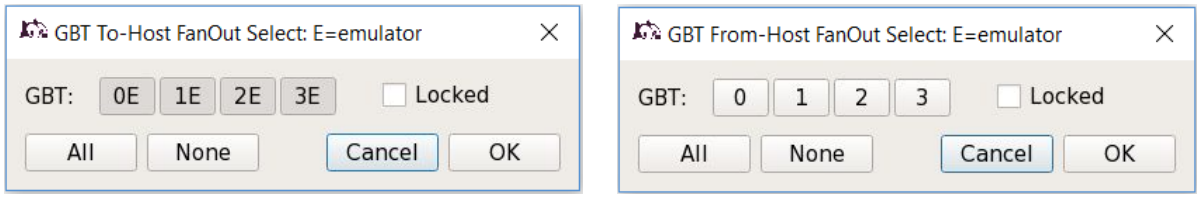


Figure 22: Fan out control for to-host (left) and from-host (right) directions. The setting for each link is displayed separately (in this case for a 4-link VC-709 system). It is also possible to lock the settings using the dedicated check box.

to external data for the duration of a test. In this case it is possible to lock the values by selecting the 'locked' check box. Applications will then be unable to change these settings until the card is reconfigured from this interface or the FPGA is reprogrammed. More information on configuring TTC transfer to the front-end are available in Section 6.1.3 below.

6.1.1.2 Data Timeout Control Dialog

FELIX offers the facility to time out pending incoming data after a configurable window from receipt of the first related packets. This is applicable for both regular and TTC data (in the to-host direction). Should data time out then all available blocks are transferred to the host with a dedicated trailer indicating that a timeout has occurred. The timeout feature is enabled by default, but can me modified or disabled/re-enabled via the control dialog accessible by selecting the 'Timeout' button in the global panel. This will open the dialog shown in Figure 23. From here it is possible to disable/enable both regular data and TTC timeouts using the check boxes, as well as modify the timeout window sizes. This should typically only be done under the guidance of a FELIX developer for debugging purposes. Note that changes made in this dialog are immediately propagated to the FELIX card in question once you select 'OK'.

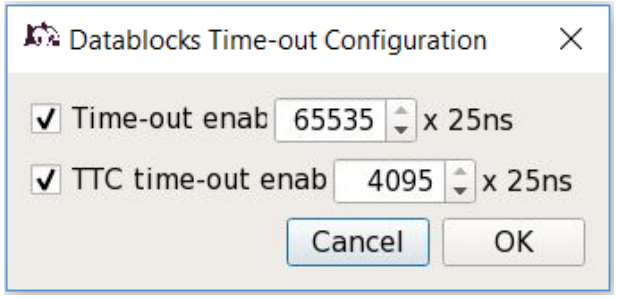


Figure 23: elinkconfig data timeout control dialog.

6.1.1.3 Clock Source Selector Dialog

As mentioned in Section 3.5.1, FELIX supports two different firmware clock sources. It is possible to switch between these sources from `elinkconfig` from the clock source selector dialog, accessible by clicking the 'clock' button in the global panel. The selector dialog is shown in Figure 24, and is a simple two button toggle between TTC and local clock. Note that changes made here will be immediately propagated to the FELIX card in question once you select 'OK'. Please also consult Section 3.5.3 before making any clock changes, to ensure you correctly configure your FELIX card's jitter cleaner post-clock change to ensure continued stable operation.

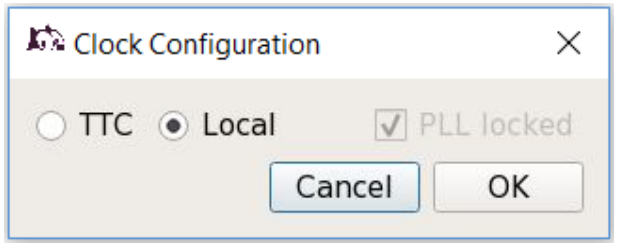


Figure 24: elinkconfig clock source selector dialog.

6.1.2 To-Host Panel

The to-host panel provides access to the configuration of the currently selected link (GBT or FULL mode) in the to-host direction. The type of panel to show can be selected in the global panel as described in Figure 21. In the GBT case it is possible to configure the complete set of E-links associated with this link, split up by E-group. It is also possible to configure the SCA and TTC links (see Section 6.1.6.4 for more info), the latter of which provides L1 accept information to the host. For each link it is also possible to select the type of encoding to be used, although 8b10b is recommended for all regular data links. A more detailed look at this panel is presented in Figure 25.

Note: In GBT mode it is possible to run in either 'Normal' or 'Wide' mode. Wide mode increases the width of the GBT link to accommodate two extra E-groups. This feature is currently not yet rigorously tested. If wide mode is selected the two greyed out E-groups in the to-host panel will become accessible.

In FULL mode this panel provides fewer options, as such this link mode does not contain logical E-link subdivisions. This version of the panel is presented in Figure 26.

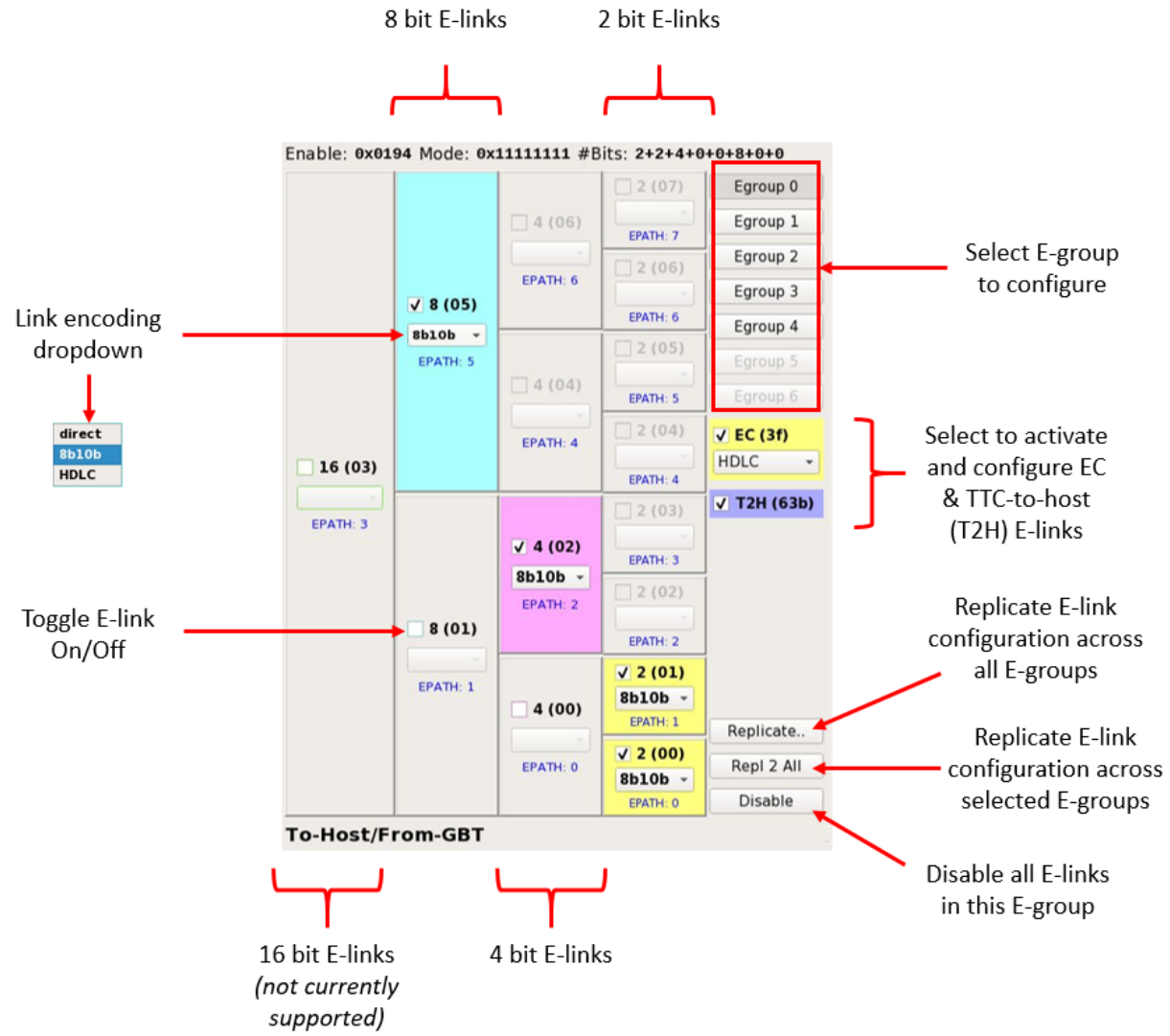


Figure 25: elinkconfig to-host panel (GBT mode). Various configuration options and tools are indicated as they appear in the panel.

6.1.3 From-Host Panel

The from-host panel makes it possible to configure the GBT links transporting data from FELIX towards connected front-end electronics. This panel only exists in GBT mode form as FULL mode is only a to-host protocol, and any FULL mode firmware will implement from-host links as GBT. A more detailed look at this panel is presented in Figure 27. A key difference between this panel and the to-host panel is that the link encoding available also includes several different TTC paths (in this case TTC-1 and TTC-2 are shown) which are for the propagation of TTC information from FELIX to the front-end. Depending on the E-link width used TTC paths from 0 to 4 are can be made available. Using this encoding selector it is therefore possible to nominate specific E-links to carry TTC data as needed.

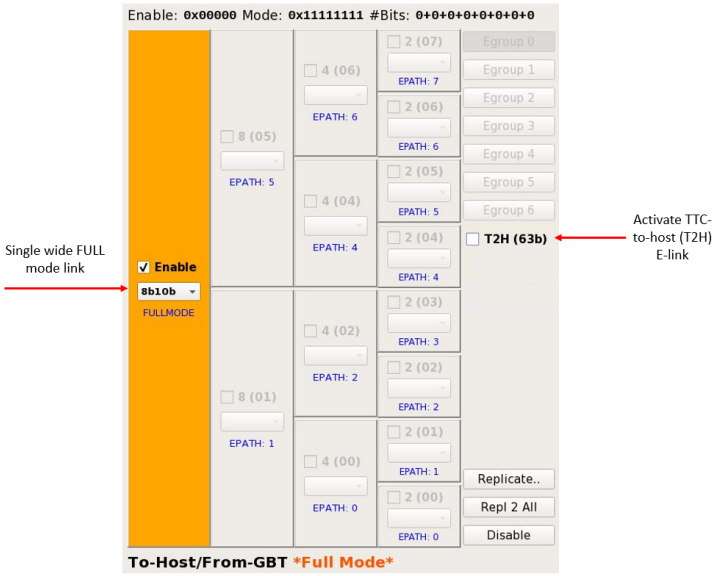


Figure 26: elinkconfig to-host panel (FULL mode).

6.1.4 Link and Data Generator Configuration Upload Dialog

The to-and-from host panels allow you to put together a complete configuration set for all links handled by a given FELIX card. Once you have prepared your desired configuration, you can upload it to the FELIX in question by selecting the 'Generate/Upload' button in the upper panel on the right. This will open the upload dialog, as shown in Figure 28. The GBT version is shown, but the FULL mode variant is essentially identical, beyond some disabled developer features.

Once the panel is prepared, select 'Upload' from the middle box labeled 'E-link Configuration' to write your configuration to the card. If you also wish to configure the FELIX on-board data generators for tests in emulation mode select the 'Upload' button in the lower 'Emulator Data' box. Note that if you are running in emulation mode and wish to change your E-link configuration you must remember to upload to the emulator every time you upload a change. Note that in FULL mode the data generators will only produce FULL mode data in the to-host direction. In the to-front-end direction GBT data will be produced. In GBT mode GBT data will be produced in both directions.

Once your configuration is uploaded you can then proceed to use the FELIX system as normal, the new settings will take effect immediately. To avoid unexpected behaviour please avoid reconfiguring the links while the FELIX card in question is in active use in your system.

6.1.5 Guide to Valid E-link Configurations

The E-link configuration uploaded to a FELIX card is actually a set of instructions to a component known as the *Central Router*. This is responsible for sending incoming data (in either direction) to the correct remote end point, as defined by E-link. For FULL mode there is no such thing as an E-link, and so the Central Router merely propagates a wide stream of bits across the link. In the GBT case, E-links are defined as separate logical links within a given physical GBT link. E-links can have (in the current implementation) three different bit widths, which given the link clock defines the maximum bandwidth

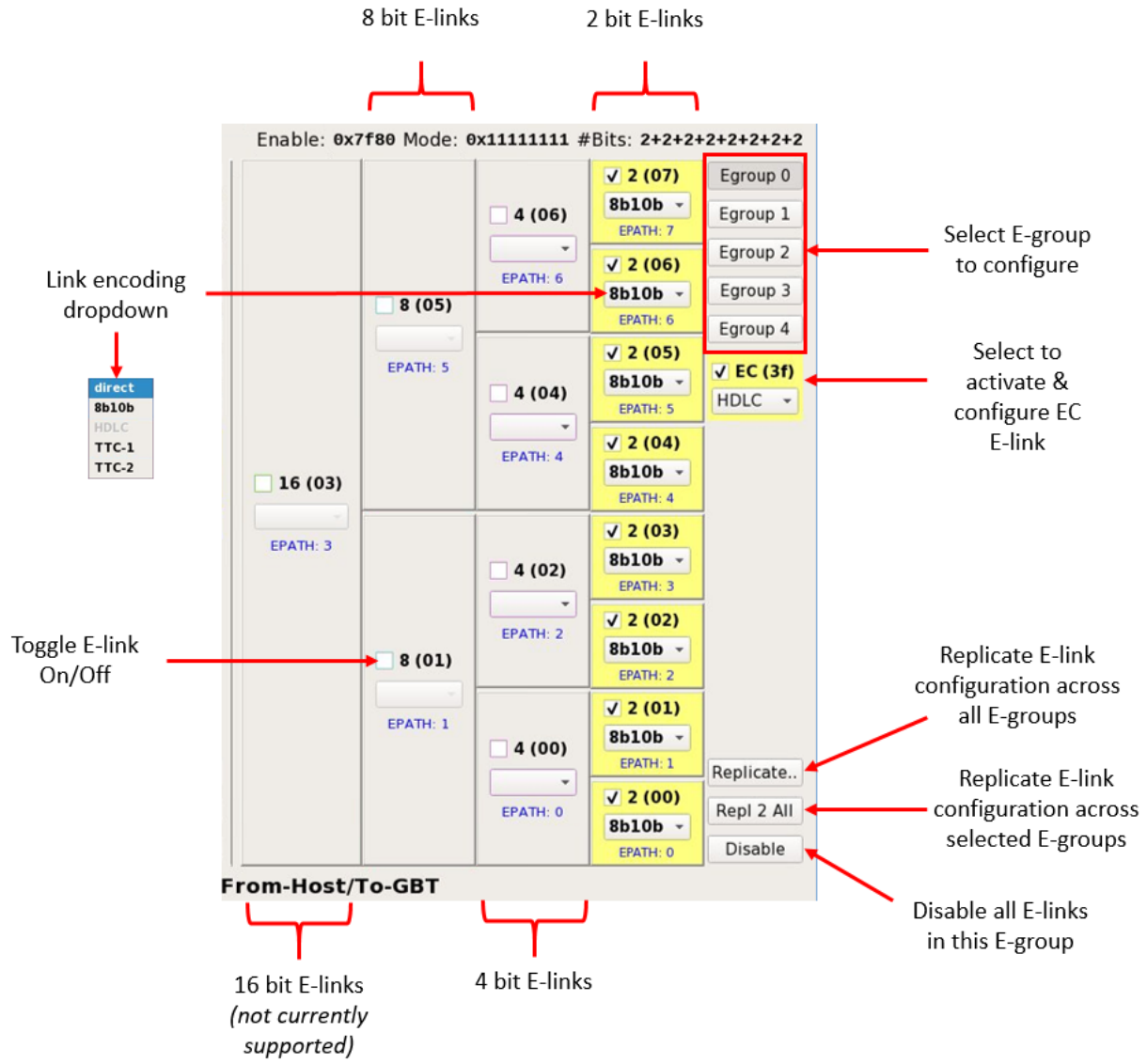


Figure 27: elinkconfig from-host panel (GBT mode). Various configuration options and tools are indicated as they appear in the panel.

they can sustain. The widths are 2, 4 and 8 bits, running at 40, 80 and 160 MHz respectively. There is currently no support for 16 bit E-links. A GBT link can therefore be considered as a logical aggregation of low bandwidth links into one high bandwidth transfer. For full details please consult the official documentation [3].

In 'Normal' mode, a GBT link is 80 bits wide, and this puts an upper limit on the number of E-links. It is therefore possible to have few wide 8 bit links, a larger number of narrower 2 or 4 bit links, or a mixture of the two. Should a GBT be operated in 'Wide' mode (not currently widely used or tested) then a further 32 bits are available within the GBT link (i.e. 112 in total), allowing for more E-links. The structure of a normal mode GBT frame is shown in Figure **??**. It is up to the user to decide how much of the GBT width to utilise as per their front-end needs. It is permitted to leave link bandwidth unused by not assigned E-links to that part of the GBT frame.

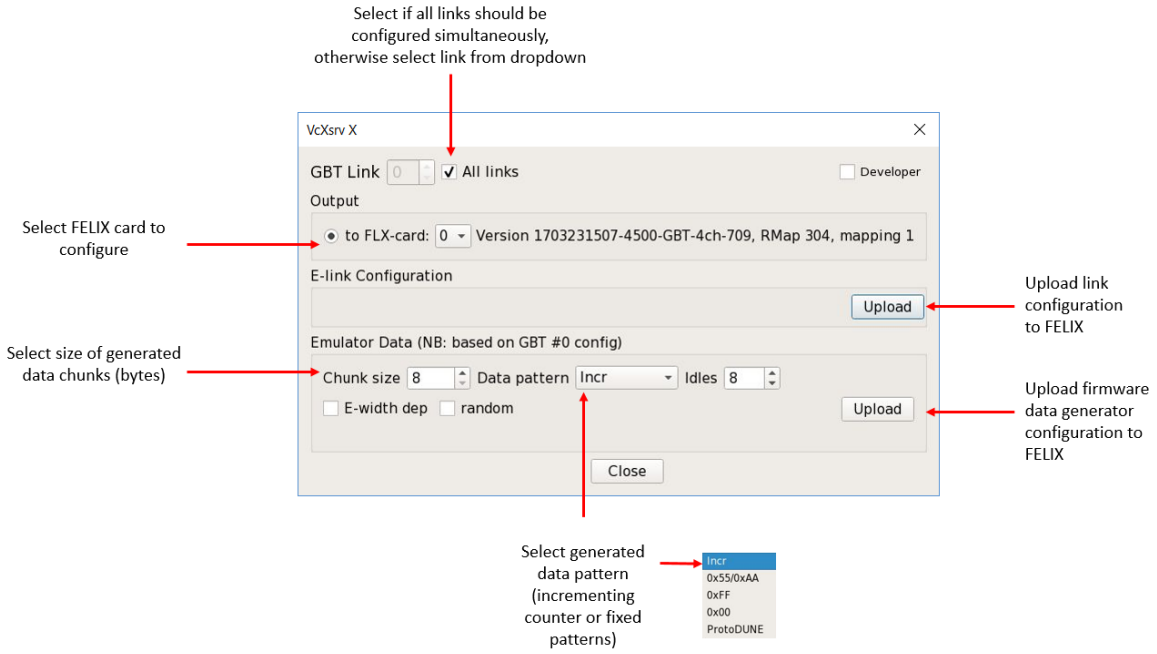


Figure 28: elinkconfig upload panel. Any features not indicated with arrows should be considered for experts only, and used only under consultation with a FELIX developer.

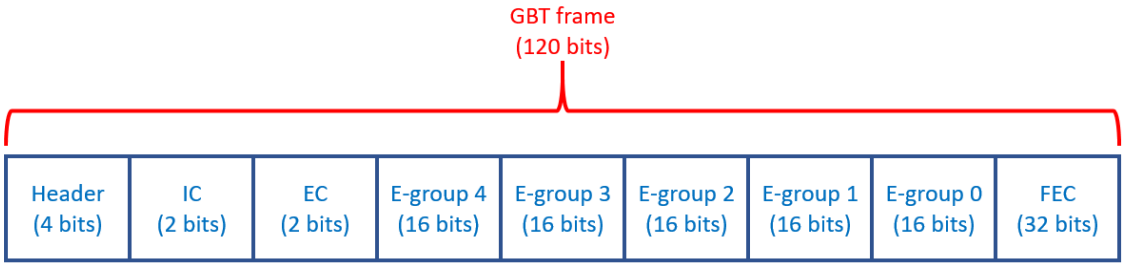


Figure 29: Bit structure of a GBT frame, showing E-groups, IC and EC links, as well as GBT header and Forward Error Correction (FEC).

Within a given GBT link, logical links are subdivided for management purposes into 16 bit wide 'E-groups'. Each E-group logically contains a combination of E-links up to an aggregate of 16 bits of width, looking at either extreme this means up to 8 of the narrowest 2 bit E-links at one end, or two of the widest 8 bit E-links at the other. The E-group is the unit of connectivity around which the `elinkconfig` interface is built, with the to and from-host panels designed around E-group granularity.

Looking within the E-group, there is one further layer of link identification to consider. Each group supports up to 8 logical 'E-paths'. These correspond to the logical connection end-points which the Central Router supports. For each E-group it is therefore only possible to send data to 8 destinations, which is designed to correspond to the maximum number of 2 bit E-links. However, the E-path structure imposes an additional restriction on E-link assignment. Because of the routing structure the E-path end points needed by E-links of different widths can overlap, meaning only a link of one width or the other is possible. Consider the routing diagram presented in the to-host panel in Figure 25. This display is

designed to mirror the structure of the Central Router to make the dependencies as transparent as possible to users.

The active 4 bit E-link in this panel is using E-path 2. This means not only that E-path 2 is unavailable for the 2 bit E-link which could be assigned to that path (see the column to the right) but also the 2 bit E-link for E-path 3, as the wider 4 bit link in E-path 2 overlaps with it. It is therefore possible to either have the 4 bit E-link active, or one or both of the 2 bit links, but 4 bit link cannot be active at the same time as either 2 bit link. Note that this doesn't affect the E-link which could be active in E-path 1, as this doesn't overlap. In this case this link is disabled through user choice, not through any logical limitation. However, if you wanted to enable the 8 bit wide E-link at E-path 1, this would overlap with all 2 and 4 bit E-links to its right, meaning only it could be active.

To summarise, in order to build a valid link configuration no two links using the same E-path can be active at the same time within an E-group. Depending on the width of the link in question this may also disqualify other links of smaller width if they overlap with it. For this reason, `elinkconfig` will not allow you to select overlapping links.

Within a given link map, each E-link can be configured to use different encoding formats as per front-end requirements. This area is still subject to active development, and it is strongly recommended that users work towards basing systems on 8b10b encoding. For FULL mode 8b10b is also the default.

6.1.6 Guide to common configuration tasks

6.1.6.1 Working with E-link configurations stored in files

`elinkconfig` can read and store configuration sets in .elc files. In order to load a previously existing configuration set into the tool simply select 'Open' from the global panel and choose the file to be loaded. The GUI will be automatically updated to reflect the new configuration. From here you can modify the configuration (if needed) by e.g. using the to and from-host panels to enable/disable E-links. Once your changes are complete you can upload the new configuration to the FELIX card of your choice using the 'Generate/Upload' button in the global panel. Make sure to upload both the link and data generator configurations if you wish to use the latter. Finally, you can save your modified configuration to a file by selecting 'Save' from the global panel.

6.1.6.2 Modifying the existing E-link configuration on a FELIX card without a file

If you are working without .elc files and wish you modify the existing configuration on a card you must first load it into the tool by selecting the card in question via the global panel and then pressing the 'Read Cfg' button. This will populate the GUI with the configuration currently active on the card. From here you can modify the configuration as required and upload a new version to the card as advised above. You can also save your configuration to a file.

6.1.6.3 Configure L1AInfo E-links to host

Users wishing to route Level-1 Accept information to network endpoints via the FELIX host may configure any number of E-links for this purpose. Each such E-link will provide a 20-byte 'L1AInfo' block containing information for each Level-1 Accept. The contents of the block are presented in Figure 30. Users can view and edit the E-links currently selected for L1AInfo transmission by selecting the 'EC/TTC' button in the to-host panel. More details on this and other FELIX data structures is available in Appendix C.

0	FMT(8)	Len(8) = 20	*reserved*	BCID(12)
1	XL1ID(8)	L1ID(24)		
2	orbit(32)			
3	Trigger Type (16)		*reserved(16)*	
4	L0ID(32)			

L1Ainfo_v01

Figure 30: The Level-1 Accept information message sent to the Back end software (20 bytes) as seen as five 32-bit words.

6.1.6.4 Configure TTC E-links from host

Users may configure any number of to-front-end links for the purpose of transferring TTC information to their electronics. The TTC data arriving at the FELIX card will be automatically decoded, and subsets made available to users for relay to front-ends in a configurable manner. The subsets which can be sent depend on the width of the E-link chosen for the transfer. By selecting the encoding box on the to-host panel for any given link it should be possible to see which options are available for that link.

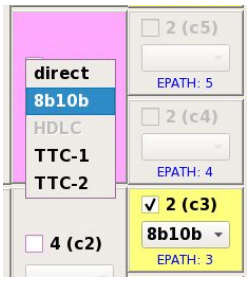


Figure 31: Example drop-down encoding menu for a 4-bit E-link. As this is a 4-bit link, the TTC-1 and TTC-2 options from Table 3 are available.

The current configuration sets are presented in Table 3, although this can evolve based on user requirements. For example, 2-bit E-links can only be configured in 'TTC-0' mode, meaning only the L1A and the full, non-decoded B-channel data stream can be sent. Alternatively, 4-bit E-links can be configured to send L1Accepts, Bunch Counter and Event Counter Resets, and a choice of either the non-decoded B-channel data stream or a user defined broadcast bit. Detector groups should communicate to the FELIX group which bits in which locations they need.

Table 3: Possible TTC options (Brcst[7:2] are the TTC user defined broadcast command bits (Brcst[1] is ECR, Brcst[0] is BCR). Bit 0 is the first bit transmitted out.

E-link option		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	2 bits							B-chan	L1A
1	4 bits					B-chan	ECR	BCR	L1A
2	4 bits					Brcst[2]	ECR	BCR	L1A
3	8 bits	B-chan	Brcst[5]	Brcst[4]	Brcst[3]	Brcst[2]	ECR	BCR	L1A
4	8 bits	Brcst[5]	Brcst[5]	Brcst[4]	Brcst[3]	Brcst[2]	ECR	BCR	L1A
5	4 bits					BCR	BCR	BCR	BCR
6	2 bits							BCR	BCR

6.1.6.5 Configure SCA E-links to/from host

SCA E-links are 2-bit wide HDLC-encoded links. This so-called EC mode is designed to carry slow control information to and from any desired front-end location. Users may set any number of 2-bit E-links (in either direction) to EC mode (not just the GBTx "EC" E-link) by selecting the 'EC/TTC' button (to host panel) or 'EC' button (from host panel), activating a link and selecting HDLC encoding. Communication with an SCA ASIC requires one TX and one RX link. FELIX performs the HDLC encoding and decoding. Note that an OPC-UA server and client for the SCA ASIC are being provided to allow high level communication with an SCA ASIC by user software.

6.2 Low Level Tools (System Status Monitoring & Control)

The following section will cover some general tools which allow you to monitor the state of your FELIX system, as well as make configuration changes and reset the system as necessary. All tools provide more detailed descriptions of functionality through their help output, accessible by running the tool with the '`-h`' option.

6.2.1 flx-info

The flx-info application is a command line tool which can print to the screen a range of monitoring and configuration information for you FELIX card(s). By default you will be presented with some system version and health status, as well as a basic link description. To access this default printout run the following command:

```
$ flx-info
```

This will provide output similar to that which is presented in Figure 32, including general information such as the firmware revision 'SVN version' and link mode 'GBT or FULL'. To produce more verbose output pass the tool the `-v` or `-vv` option.

Beyond this basic output, flx-info also makes it possible to read many FELIX configuration registers in detail. For a complete list of these options run flx-info with the '`-h`' flag. This will produce output like that shown in Figure 33.

```
General information

There are 1 FLX cards installed in this computer
-------------------
Reg Map Version  3.7
Card ID:         FLX-709
FW version date: 02/07/17 00:04
SVN version:     5214
FIRMWARE MODE:   GBT

Output of lspci | grep Xil:
03:00.0 Communication controller: Xilinx Corporation FPGA Card XC7VX690T


Interrupts, descriptors & channels
----------------------------------
Number of interrupts : 8
Number of descriptors: 8
Number of channels   : 4

Links and GBT settings
----------------------------------
Number of channels    : 4
GBT Wrapper generated : YES
Optical transceivers  : 4

Clock resources
----------------------------------
Local clock in use  : YES
Internal PLL Lock   : YES

ADN2814 TTC Status: OFF
```

Figure 32: Default output of flx-info (assuming a VC-709 running GBT-mode firmware)

```
Usage: flx-info [OPTIONS] [COMMAND] [CMD ARGUMENTS]
Displays information about a FLX device.

Options:
  -d NUMBER                     Use card indicated by NUMBER. Default: 0.
  -v                            Verbose mode.
  -D level                      Configure debug output at API level. 0=disabled, 5, 10, 20 progressively more verbose output. Default: 0.
  -h                            Display help.
  -V                            Display the version number
Commands:
  GBT                           Shows GBT channel alignment status.
  FMC_TEMP_SENSOR               Display FMC temperature from TC74 sensor.
  ADN2814                       Display ADN2814 register 0x4.
  CXP                           Display temperature and voltage from CXP1 and CXP2
  SFP                           Display information from Small Form Factor Pluggable transceivers
  DDR3                          Display values from DDR3 RAM memory
  ID_EEPROM                     Display the first 32 bytes of the eeprom memory
  SI5324                        Display SI5324 status
  SI5345                        Display SI5345 status
  LMK03200                      Display LMK03200 status
  ICS8N4Q                       Display ICS8N4Q status
  EGROUP       [channel]  [RAW] Display values from EGROUP registers:
                                    If no channel is specified, display all available.
                                    Using Hexadecimal notation if RAW is specified.
  ALL                           Display ALL information.
```

Figure 33: List of all flx-info features

6.2.2 flx-config

The flx-config tool allows users to modify specific FELIX control and configuration registers from the command line. This should normally only be done on advice from a member of the FELIX development team. Other features are also available, but these should be considered for experts only unless advised otherwise by the development team. The two primary features users will use will be the 'list' and 'set' features. List mode will dump the values of all FELIX registers to the screen. This will be a large amount of output, but can be searched with e.g. grep for the desired information. To run list mode execute the following command:

```
$ flx-config list
```

To change a given register value use the 'set' feature as follows:

```
$ flx-config set REGNAME=<new val>
```

In this case REGNAME corresponds to the register to be changed and <new val> to the new value be stored. Once set you can use list mode to confirm the change.

6.2.3 flx-init

This tool has the ability to reset the GBT wrapper and transceiver, and should be performed every time the FPGA is reprogrammed (including in case of loss of power). The tool should also be run if the GBT fibres are disconnected at any point before attempting to transfer data once again.

To run the basic initialisation issue the following command:

```
$ flx-init
```

If you wish to use more features (if instructed by a member of the development team), consult the help dialog as presented in Figure 34

```
Usage: flx-init [OPTIONS]
Initializes a FLX device.

General options:
  -d NUMBER                   Use card indicated by NUMBER. Default: 0.
  -h                          Display help.
  -D level                    Configure debug output at API level. 0=disabled, 5, 10, 20 progressively more verbose output. Default: 0.
  -v                          Display verbose output.
  -vv                         Super verbose mode.
  -V                          Display the version number
GBT calibration options:
  -s SOFT|FIRM                Use SOFTWARE (SOFT) or FIRMWARE (FSM) alignment. Default: SOFT.
  -a ONE|CONTIUNOUS           Select alignment type. Default: ONE.
  -t FEC|WideBus              Select transmission mode. Default: FEC.
  -e YES|NO                   Use the descrambler output value. Default: NO.
  -f FILENAME                 Specify which file will be used to calibrate the delay. Default: ../flx_propagation_delay.conf)
TTC calibration options:
  -G NUMBER                   Get and display the status of a SI53xx
                              Legal values are:
                              1 = SI5324
                              2 = SI5345
  -C                          Set the registers of the ICS 8N4Q001L
  -X filename                 Set a clock to a given frequency using the dangerous mode
  -T mode                     Set a clock to a given frequency
                              Legal values for mode are:
                              1 = (SI5324 only on FLX-709) 240 MHz
                              2 = TTCfx-v3 and BNL-711 (SI5345) 240 MHz usign legacy code
                              3 = TTCfx-v3 and BNL-711 (SI5345) 240 MHz
     -I INSEL                 To be used in combination with -T or -X. The value given will be written into register HK_CTRL_FMC_SI5345_INSEL
                              Legal values are:
                              FLX-709: 0-->FPGA (LA01), 1-->FMC OSC, 2-->FPGA (LA18)
```

Figure 34: List of all flx-init features

6.2.4 flx-reset

The flx-reset application makes it possible to selectively reset components of the FELIX firmware, or the complete board, as needed given the situation. This should only be done if advised by a FELIX development team member. To see the list of available parameters please consult the help output as presented in Figure 35.

To reset a given component simply pass the name to flx-reset on the command line:

```
$ flx-reset COMP_NAME
```

```
Usage: flx-reset [OPTIONS]
Tool to reset different aspects from the card.

Commands:
Options:
  FLUSH            Flushes (resets) the main output FIFO toward Wupper.
  RESET            Resets the whole Wupper_core.
  REGISTERS_RESET  Resets the registers to default values.
  SOFT_RESET       Global application soft reset.
  ADN2814          Reset the ADN2814.
  ALL              Do everything.
Options:
  -d NUMBER        Use card indicated by NUMBER. Default: 0.
  -D level         Configure debug output at API level. 0=disabled, 5, 10, 20 progressively more verbose output. Default: 0.
  -h               Display help.
  -V               Display the version number
```

Figure 35: List of all flx-reset features

6.2.5 flx-monitor

The flx-monitor application is a new application dedicated to presenting status information for LTC2991-monitored components [14] aboard BNL-711 cards (and thus will not work with a VC-709). To see the list of available parameters please consult the help output as presented in Figure 36.

Note - this tool is currently only available in the developer build, but is documented here for any users already working with it. It will be available in the user release from version 3.9.2 (if built) and 4.X onwards

```
Usage: flx-monitor [OPTIONS]
Read and decode data of the LTC2991 devices on a FLX-711

General options:
  -d NUMBER                 Use card indicated by NUMBER. Default: 0.
  -D level                  Configure debug output at API level. 0=disabled, 5, 10, 20 progressively more verbose output. Default: 0.
  -h                        Display help.
  -V                        Display the version number
Options:
  -r                        Raw dump of the registers of the LTC2991
```

Figure 36: List of all flx-monitor features

6.3 Dataflow from Front-end via FELIX to FELIX host PC

6.3.1 fdaq

Fdaq is the primary tool for testing the FELIX data acquisition path. The tool can run in multiple modes, from waiting for input for FELIX from a front-end source to using one of the two internal data generators on the card. In both modes fdaq will measure and report throughput for the duration of the test. Data can be dumped to a file or discarded upon receipt. If running in discard mode fdaq will check the integrity of the data blocks and chunks it receives (e.g. block headers and chunk sizes). If an error is found the test will, by default, stop and fdaq will report on the first detected error. However, if the '-D' option is used the run will continue with a report printed on all errors received.

Note that this section assumes that your E-links and data generators are configured properly as specified in Section 6.1. In this section we will cover various scenarios, but a list of all options can be found in the help output, which will be similar to that shown in Figure 37.

Note: in full mode it will likely only be possible to run fdaq or a couple of seconds as you will rapidly exceed the maximum rate at which you can write to disc, which will then cause the application to abort to avoid buffer overflow. For reference, for a typical SSD this limit is approximately 300MB/s.

```
fdaq version 17103100
Stream data from FLX-card to file(s). Whenever the set maximum file size
is exceeded a new file is created. Every second a status line
with data rates, data totals and memory buffer status is displayed.
(NB: if no filename is provided all data is consumed while checking the data blocks,
     i.e. blockheader and trailers; use -D to *not* terminate after an error is detected.)

Usage: fdaq [-h|V] [-D] [-d <cardnr>] [-b <size>] [-e|E] [-f <size>]
            [-i <dma>] [-I] [-r <runnr>] [-t <secs>] [-X] [<filename-base>]
  -h          : Show this help text.
  -V          : Show version.
  -D          : Debug mode on, i.e. output additional info.
  -d <cardnr>: FLX-card to use (default: 0).
  -b <size>   : DMA (cmem_rcc) memory buffer size to use, in MB (default 1024, max 4096).
  -e|E        : Enable FLX-card data generator, internal (e) or
                external (E) (default: false).
  -f <size>   : Maximum file size, in MB (default 1024, max 4096).
  -i <dma>    : FLX-card DMA controller to use (default: 0).
  -I          : use interrupt to receive data (default: polling)
  -r <runnr> : Run number to use in file names (default: none).
  -t <secs>   : Number of seconds to do acquisition (default: 1).
  -T          : Do not add datetime as part of file names.
  -X          : Stream data from individual e-links to separate files (default: false).
 <filename-base>: Name to be combined with datetime+runnumber+counter of files created
```

Figure 37: List of all fdaq features

6.3.1.1 Running DAQ Test with External Data Source

The most simple configuration for fdaq to run in is to listen for any data coming into FELIX over the GBT/FULL mode link and measure the bandwidth as this arrives at the host. In this mode the data is discarded. The only parameter a user must define is the time in seconds for which fdaq should perform the test. The default time is 1 second. The syntax is as follows:

```
$ fdaq -t <time>
```

For a three second test the output will resemble Figure 38.

```
Consume FLX-card data while checking the data (blockheader and trailers);
stops when an error is encountered
Opened FLX-card 0, firmw 1707020004-5214-GBT-4ch-709 channels=4 (cmem buffersize=1073741824)
**START** using DMA #0
-> 1 sec, Rates: recv   0.0 MB/s, file   0.0 MB/s; Total: recvd 0 B, file 0 B; Buffer: 0%, wraps 0
-> 2 sec, Rates: recv   0.0 MB/s, file   0.0 MB/s; Total: recvd 0 B, file 0 B; Buffer: 0%, wraps 0
-> 3 sec, Rates: recv   0.0 MB/s, file   0.0 MB/s; Total: recvd 0 B, file 0 B; Buffer: 0%, wraps 0
**STOP**
-> Data checked: Blocks 0, Errors: header=0 trailer=0
Exiting..
```

Figure 38: Output from fdaq test

If you would like to dump your data to a file for analysis simply specify a filename after the other command line parameters:

```
$ fdaq -t <time> testfile
```

This will run as above and produce a time-stamped .dat file in the directory you are running with a name of the format 'testfile-<timestamp>.dat'. You can specify the maximum size for the file with the '-f' option specifying a size in megabytes (default 1024, max 4096). If you would like to split the input from multiple E-links into different files use the '-X' option. The E-link number will be added to the filename.

6.3.1.2 Running DAQ Test with Internal Data Generation

A facility to use both data generators within the FELIX card for the purposes of testing is provided by fdaq. The 'internal' generator is connected directly to the data output path of the card (i.e. after input side of the GBT link interface). Data from this generator therefore passes through the full FELIX firmware data path with the exception of the link layer itself. The 'external' data generator is connected to the output path before the GBT layer. This means it can be configured to send data out of a GBT link. If some loopback fibres are connected it is therefore possible to send data out of one GBT transceiver and into another on the same FELIX and therefore test more of the data path. To access these options in fdaq one must use the '-e' option for internal data generation and '-E' for external data generation. The output from fdaq will be the same as shown for the external source tests. An example is shown in Figure 39.

```
Consume FLX-card data while checking the data (blockheader and trailers);
stops when an error is encountered
Opened FLX-card 0, firmw 1707020004-5214-GBT-4ch-709 channels=4 (cmem buffersize=1073741824)
**START(emulator)** using DMA #0
-> 1 sec, Rates: recv 1273.7 MB/s, file   0.0 MB/s; Total: recvd 1273.7 MB, file 0 B; Buffer: 2%, wraps 1
-> 2 sec, Rates: recv 1276.0 MB/s, file   0.0 MB/s; Total: recvd 2549.7 MB, file 0 B; Buffer: 2%, wraps 2
-> 3 sec, Rates: recv 1277.0 MB/s, file   0.0 MB/s; Total: recvd 3826.8 MB, file 0 B; Buffer: 1%, wraps 3
**STOP**
-> Data checked: Blocks 3727840, Errors: header=0 trailer=0
Exiting..
```

Figure 39: Output from fdaq test with internal data generation

6.4 Dataflow from FELIX Host PC to Front-end Systems via FELIX

6.4.1 fupload

The FELIX software suite makes it possible to transfer data from the FELIX host PC via the FELIX card to the front-end across a GBT link. This is done via the 'fupload' tool. With this tool it is possible to transfer data either from a user defined file, or from the FELIX data generators, to a specified E-link across a GBT connection. The full range of features of the tool can be seen in the help text, as presented in Figure 40.

Note: This tool works with FULL mode firmware versions, but uses a GBT link up to the front-end.

6.5 FELIX Configuration Tools

6.5.1 felink

The felink tool is a link descriptor interpreter which allows you to work out the E-link ID for a given link given GBT/E-group/E-path (or vice versa). This is intended to be used in conjunction with e.g. fupload to allow users to work out which link ID they should target with their data. Some examples of possible uses will be given below, but you can find all possible options in the help text, as shown in Figure 41.

A list of all valid E-links and coordinates can be seen with list mode, available with the following syntax:

```
$ felink -l
```

```
fupload version 17052300
Upload data (test data or from file) to the given FLX-card e-link.
The e-link number is provided as a (hex) number directly (-e option),
as a set of -G/g/p options, or as a set of -G/I/w options, unless option -R is given.
Checks whether the e-link is valid and configured on the selected FLX-card,
unless option -c is given.

In ASCII data files one line is one data packet (hexadecimal byte values separated by spaces),
while lines starting with certain characters may be used to:
  # insert a comment line
  + insert a packet of the given length containing bytes of the given byte value
  & insert a configurable delay in microseconds between two packets
  > change the e-link number to upload to

Usage: fupload [-h|V] [-D] [-d <cardnr>] [-b <size>] [-c] (-e <elink>
               | (-G <gbt> (-g <group> -p <path>) | (-I <index> -w <width>)) [-i <dma>]
               [-s <bytes>] [-P <patt>] [-f <speed>] [-R] [-t <secs>] [<filename>]
  -h          : Show this help text.
  -V          : Show version.
  -b <size>   : DMA (cmem_rcc) memory buffer size to use, in MB (default 16, max 4096).
  -B          : Contents of <filename> is read as binary data (default: ASCII).
  -c          : Do not check whether e-link is valid on FLX-card.
  -d <cardnr>: FLX-card to use (default: 0).
  -D          : Debug mode on, i.e. display blocks being uploaded.
  -f <speed> : Speed up default upload rate of about 8MB/s by factor <speed> (default:1)
  -i <dma>    : FLX-card DMA controller to use (default: 1).
  -P <patt>   : Test data pattern: 0=incr, 1=0x55/0xAA, 2=0xFF, 3=incr-per-chunk (default:0).
  -r <repeat>: Test data repeat count: upload <repeat>*<bytes> bytes of data (default: 30).
  -R          : Upload data raw, not as CR from-host datapackets with header.
  -s <bytes> : Number of bytes per chunk to upload (default:32).
  -t <secs>   : Number of seconds for DMA time-out or wait until DMA done when 0 (default: 0).

Options to define the e-link to use:
  -e <elink> : E-link number (hex) or use -G/g/p or -G/I/w options.
  -E <elink> : an optional 2nd E-link number to upload to
               (alternating with the first given E-link number).
  -G <gbt>    : GBT-link number.
  -g <group> : Group number.
  -p <path>   : E-path number.
  -I <index> : Index of first bit of e-link in GBT frame.
  -w <width> : e-link width in bits (2, 4, 8 or 16).

  <filename> : Name of file with data to upload (ASCII or binary),
               or test pattern data if no name is given.
```

Figure 40: List of all fupload features

```
felink version 16092800
Convert a given E-link number into GBT, egroup and epath numbers
as well as GBT and bit-index and width, or the other way around.
The E-link number is provided as a (hex) number directly (-e option),
as a set of -G/g/p options, or as a set of -G/I/w options.
Optionally checks if this E-link is valid and configured on a given FLX-card (option -d),
in either to/from-host direction.
Use option -l to display a list of valid E-link numbers,
optionally in combination with -G or -g options to restrict the list
to certain GBT-links and/or egroups

Usage: felink [-h|V] [-d <cardnr>] (-e <elink>
               | (-G <gbt> (-g <group> -p <path>) | (-I <index> -w <width>))
  -h          : this help text
  -V          : display version
  -d <cardnr>: FLX-card to use (default: 0)
  -e <elink> : E-link number (hex) or use -G/g/p or -G/I/w options
  -G <gbt>    : GBT-link number
  -g <group> : Group number
  -l          : Show a list of valid E-link numbers (use options -G, -g, -p to restrict the list)
  -p <path>   : E-path number
  -I <index> : Index of first bit of e-link in GBT frame
  -w <width> : E-link width in bits (2, 4, 8 or 16)
```

Figure 41: List of all felink features

6.5.1.1 Finding E-link ID from GBT/E-group/E-path of GBT/Bit address/width

Consider the example where a user wishes to know the E-link ID for a link connected to GBT link 2, within E-group 3 and E-path 4. This can be done as follows:

```
$ felink -G <GBT ID> -g <egroup ID> -p <epath ID>
```

Filling these in gives results as in Figure 42, from which we can see that the E-link ID is 0x9C. The results also show alternative coordinates for the E-link in terms of GBT bit address and width.

```
-bash-4.1$ felink -G 2 -g 3 -p 4
E-link 09C = GBT #2 group #3 path #4, bit#56 width=2|4
```

Figure 42: Result of E-link ID calculation by GBT/E-Group/E-path

It is also possible to search for link ID using the GBT ID, bit address of the start of the E-link in the GBT frame and E-link width. The syntax is as follows, noting that the index must correspond to a valid E-link start point.

```
$ felink -G <GBT ID> -I <bit address> -w <E-link width>
```

If a user then wants to search for GBT 1, bit 4 and width 2 the results will be as per Figure 43. This identifies the E-link in question as 0x42.

```
-bash-4.1$ felink -e 0x55
E-link 055 = GBT #1 group #2 path #5, bit#42 width=2 OR bit#40 width=8
```

Figure 43: Result of E-link ID calculation by GBT/bit address/width

These calculations can also be done in reverse, to yield the coordinates of a given known E-link ID. For this use the following syntax:

```
$ felink -e <E-link ID in hex>
```

If as user then wants to know the coordinates of e.g. E-link 0x55 the tool can be used to give the results in Figure 44. From this it can be seen that the GBT ID is 1, the E-group ID 2 and the E-path ID 5. An estimate for the bit address and width is also displayed.

```
-bash-4.1$ felink -G 1 -I 4 -w 2
E-link 042 = GBT #1 group #0 path #2, bit#4 width=2|4
```

Figure 44: Result of E-link coordinate search for a known E-link ID

6.5.2 fereverse

Note: this tool replaces 'feswap', which is now deprecated.

The fereverse tool makes it possible to swap the bit ordering of data propagating through a designated E-links (or set of links). For more details consult the help text, as shown in Figure 45.

In order to use the tool to toggle the bits for a given E-link use the following syntax:

```
$ fereverse -d <FELIX ID> -G <GBT ID> -g <E-group ID> -p 1 <set/reset>)
```

```
fereverse version 17060500
Enable, disable or display the bit-order reversal feature for e-links,
a setting per e-path (e-link).
Without keyword 'set/reset' the current setting is displayed.

Usage: fereverse [-h|V] [-d <cardnr>] [-G <gbt> [-g <group>] [-p <path>]] [-e <elink>] [-f] [-t] [set|reset]
  -h         : Show this help text.
  -V         : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -G <gbt>   : GBT-link number
  -g <group> : Group number (default: all groups).
  -p <path>  : E-path number (default: all paths).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -f         : Configure FromHost only.
  -t         : Configure ToHost only.
  set        : Enable e-link bit-reversal.
  reset      : Disable e-link bit-reversal.
```

Figure 45: List of all fereverse features.

In this case the 'set' option indicates the bits should be switched and 'reset' indicates deactivation of the switch. It is also possible to pass the E-link ID directly using the '`-e`' option. If neither set or reset are specified the tool will simply report back the current status. Examples of both cases can be found in Figures 46 and 47.

```
-bash-4.1$ fereverse -d 0 -G 1 -g 1 -p 1 set
GBT 1 egroup 1 epath 1 TH: ENABLED
GBT 1 egroup 1 epath 1 FH: ENABLED
```

Figure 46: Using fereverse to enable endianness swap by specifying GBT/E-group and E-path.

```
-bash-4.1$ fereverse -d 0 -e 49 reset
GBT 1 egroup 1 epath 1 TH: disabled
GBT 1 egroup 1 epath 1 FH: disabled
```

Figure 47: Using fereverse to disable endianness swap with E-link ID.

6.5.3 fgpolar

The fgpolar tool makes it possible for FELIX to adapt to the bit polarity of data produced by front-end systems and sent via a Versatile Link [15] transceiver. The transceiver, by design, swaps the polarity of incoming and outgoing bits (i.e. 0 becomes 1 and vice-versa). Some front-end systems may already account for the swap in their design, but in order to send and receive packets to and from those who haven't this tool configures FELIX to automatically swap the bits for any designated GBT links (and therefore all E-links within). For more details consult the help text, as shown in Figure 48.

In order to use the tool to toggle the polarity of a particular GBT link use the following syntax:

```
$ fgpolar -d <FELIX ID> -G <GBT ID> <set/reset>)
```

In this case the 'set' option indicates the activation of a polarity switch and 'reset' indicates deactivation of the switch. It is also possible to modify the Tx and Rx directions separately using the '`-r`' and '`-r`' flags accordingly. By default both will be changed. The expected output from the tool can be found in Figures 49 and 50.

```
fgpolar version 17060900
Configure or display the GBT transceivers RX and TX polarity.
Usage: fgpolar [-h|V] [-d <cardnr>] [-G <gbt>] [set|reset]
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -G <gbt>    : GBT-link number.
  -r          : Configure RX only.
  -t          : Configure TX only.
  set         : Set reverse polarity for given GBT transceiver(s).
  reset       : Set default polarity for given GBT transceiver(s).
 (without keyword set/reset the current setting is displayed)
```

Figure 48: List of all fgpolar features.

```
-bash-4.1$ fgpolar -d 0 -G 1 set
GBT 1 RX polarity: 1
GBT 1 TX polarity: 1
```

Figure 49: Using fgpolar to swap bit polarity.

```
-bash-4.1$ fgpolar -d 0 -G 1 reset
GBT 1 RX polarity: 0
GBT 1 TX polarity: 0
```

Figure 50: Using fgpolar to disable bit polarity swap.

6.5.4 feconf

feconf is a command line tool providing some of the functionality of elinkconfig. With this tool it is possible to upload a pre-defined E-link configuration file (.elc format) to a FELIX card. Alongside the basic configuration, feconf also makes it possible to configure the FELIX firmware date generators. For more information on the meaning of each parameter, consult Section 6.1 on elinkconfig. For a full list of commands consult the help text, as shown in Figure 51.

```
feconf version 17032900
Upload an e-link configuration from file (generated by elinkconfig) to the given FLX-card,
and generate and upload matching emulator data contents.
Usage: feconf [-h|V] [-d <cardnr>] [-s <chunksz>] [-w] [-R] [-I <idles>] <filename>
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr> : FLX-card to use (default: 0).
  -n          : Don't write the configuration, just read and display some info.
  -R          : Generate emulator data chunks with 'random' size.
  -s <chunksz>: Emulator data chunksize to generate (default: 32).
  -w          : Generate emulator data chunksize dependent on e-link width (default: false).
  -I <idles>  : The number of idles between generated emulator data chunks (default: 8).
 <filename>   : Name of .elc file with FLX-card e-link configuration.
```

Figure 51: List of all feconf features.

6.5.5 femu

The femu tool gives users command line control over the FELIX firmware data generators, both in the from and to host directions. The full range of features of the tool can be seen in the help text, as presented in Figure 52.

```
femu version 17040400
Show or configure 'FanOut-Select' registers and start/stop emulator.
Usage: femu [-h|V] [-d <cardnr>] [-e|E|n] [-l]
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -e|E|n      : Enable FLX-card data emulator, internal (e) or external (E) or disable emulator (n).
                When no option is given the current status is displayed.
  -f          : When disabling emulator set TOHOST_FANOUT to emulator (default: to external).
  -l          : 'Unlock' FanOut-Select registers.
  -L          : 'Lock' FanOut-Select registers.
```

Figure 52: List of all femu features

6.5.6 feto

The feto tool gives users command line control over the FELIX block timeout at the level down to individual E-links. If enabled FELIX will time out incoming data blocks taking longer than a designated period to arrive and attach a timeout trailer to the block. The block will then be transferred to the host as normal. The full range of features of the tool can be seen in the help text, as presented in Figure 53.

```
feto version 17072600
Enable, disable or display the instant time-out setting,
a setting per e-path (e-link), or the so-called global time-out
and associated time-out counter (number of clocks until time-out),
or the TTC time-out and associated counter.
Without keyword 'set/reset' the current setting of the requested
(group of) time-outs is displayed.

Usage: feto [-h|V] [-d <cardnr>] [-G <gbt> [-g <group] [-p <path>]]
            [-e <elink>] [-T] [set|reset] [<globcntr>]
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -G <gbt>    : GBT-link number.
  -g <group>  : Group number (default: all groups).
  -p <path>   : E-path number (default: all paths).
  -e <elink>  : E-link number (hex) or use -G/g/p options.
  -T          : Read or configure TTC time-out.
  set         : Enable time-out.
  reset       : Disable time-out.
  <globcntr>  : Global or TTC time-out counter value to set.
```

Figure 53: List of all feto features

6.5.7 fflash

The fflash tool is designed specifically for programming FLASH memory modules aboard BNL-711 from a .mcs formatted firmware image. Note that at this point, any reprogramming of the fpga from flash will require either a reboot or PCIe hotplug operation (see Section 4.2.6.1 for details) to return the board to normal operation with the new firmware image in place. A full listing of commands is available in Figure 54.

```
fflash version 17120600
Tool for programming, verifying, erasing or dumping firmware images
in BNL-711 FLX-card flash memory, or loading the selected firmware into the FLX-card.
Usage: fflash [-h|V] [-d <cardnr>] -f <flashnr> [-E] [-D] [-F] [-L|I]
              [-s <saddr>] [-u <uaddr>]] [<filename>] [prog]
  -h           : Show this help text.
  -V           : Show version.
  -d <cardnr> : FLX-card to use (default: 0).
  -D           : Read and display contents of the selected flash partition or flash file.
  -E           : Erase the selected flash partition.
  -f <flashnr>: Flash memory segment partition [0..3] to dump, to erase,
                to verify or to program (no default).
  -F           : Use the (slow) word-by-word instead of (fast) page programming method.
  -I           : Generate an INIT_B pulse on the FLX-card (to reset flash devices).
  -L           : Load firmware from the given flash partition into to card.
  -s <addr>    : I2C-switch I2C-address (hex, default=0x70, with -L option).
  -u <addr>    : uC I2C-address (hex, default=0x68, with -L option).
  <filename>   : Name of MCS file to dump, verify or program.
  prog         : Literal text string to initiate flash programming
                 (or else flash verification will take place).
Examples:
---------
Read and dump to screen flash memory image partition #2:
  fflash -f 2 -D
Erase flash memory partition #2:
  fflash -f 2 -E
Verify flash memory partition #2 against mcs file <filename>:
  fflash -f 2 <filename>
Program flash memory partition #2 with the contents of mcs file <filename>:
  fflash -f 2 <filename> prog
Load flash memory image partition #2 into the card:
  fflash -f 2 -L
Extra:
Read and dump to screen the memory image in mcs file <filename>:
  fflash -D <filename>
```

Figure 54: List of all fflash features.

6.6 General Debugging Tools

6.6.1 fcheck

fcheck is a debugging tool which can analyse the .dat files produced by the fdaq tool and check for data integrity issues. The tool will perform checks on a file to a specified degree of severity. For more details on each level consult Figure 55. As well as running checks, the tool can also be used to dump selected data blocks to screen, either split into data chunks or as raw data, to facilitate closer inspection of any issues found. To run the check, specify the file name and check detail level as follows:

```
$ fcheck -B <severity> testfile.dat
```

A full list of features available with fcheck can be seen in the help text, as shown in Figure 55

```
fcheck version 17102000
Usage: fcheck [-h|V] [-A] [-B <id>] [-c|C|D] [-F <blocks>] [-S <blocks>] [-e <elink>] [-t|T] [-0] <filename>
  -h          : Show this help text.
  -V          : Show version.
  -A          : Interpret chunks that could be GBT-SCA frames.
  -B <id>     : Do a check on (emulator) data blocks according to <id>,
                and display a data summary (default: 2).
                0: Check for proper block headers at 1k boundaries,
                   for each block 1 line of output is produced.
                1: Same as 0, but only when an error is found a line is output.
                2: Full integrity checking of blocks, starting from
                   the block trailer going through all chunks.
                3: Same as 2, including a check on expected emulator data payload,
                   which must constitute an incrementing byte.
                4: Same as 3, but inconsistent maximum values of L1ID are not reported.
  -c          : Display data 'raw' datablocks (default: chunk data) (option -F)
  -C          : Display chunk data bytes only, nothing else.
  -D          : Display only whole data chunks, i.e. the user's data frames.
  -e <elink>  : E-link number (hex) to filter for block check or block display.
  -F <blocks>: Dump <blocks> 1K data blocks to display (overrules data check option -B).
                Chunk types: BOTH="<<", FIRST="++", LAST="&&", MIDDLE="==", TIMEOUT="]]", NULL="@@",
                 OUTOFBAND="##" and "TE" for chunk truncation/error.
  -S <blocks>: Skip <blocks> of data blocks before starting check or display.
  -t          : Do NOT report chunk truncation/error/CRCerror.
  -T          : Do NOT report chunk CRCerror.
  -0          : Do NOT display time-out chunkdata bytes (zeroes).
 <filename>   : Name of file containing data to check or display.
```

Figure 55: List of all fcheck features

6.6.2 fedump

The fedump tool is designed to make it possible to dump data arriving at FELIX to the screen for debugging purposes. Users of the tool can filter the data stream by E-link ID and FELIX card number, as well as having the option of displaying the data in raw format. More advanced options are available, but should only be used in consultation with the FELIX developers. For more details consult the help text, as shown in Figure 56.

6.6.3 fec

The fec tool is designed for the communication with a GBT-SCA chip present on a remote hardware system connected to FELIX via GBT links. Commands are read and written using the EC link component of the GBT protocol. The tool makes it possible to send pre-programmed signals to the GBT-SCA, as well as custom command strings. For a full list of commands consult the help text, as shown in Figure 57.

```
fedump version 17091900
Dump selected E-link chunk data (optionally block-by-block) received from an FLX-card to screen.
Chunk types are delimited by:
BOTH="<<", FIRST="++", LAST="&&", MIDDLE="==", TIMEOUT="]]", NULL="@@", OUTOFBAND="##"
Usage:
 fedump [-h|V] [-A] [-c|D] [-d <cardnr>] [-e <elink>] [-i <dma>] [-I] [-0]
  -h          : Show this help text.
  -V          : Show version.
  -A          : Interpret chunks that could be GBT-SCA frames.
  -c          : Display data 'raw', block-by-block (default: chunk data).
  -d <cardnr>: FLX-card to use (default: 0).
  -D          : Display only whole data chunks, i.e. the user's data frames.
  -e <elink> : E-link number (hex) to filter out for display (default: no filter).
  -i <dma>   : FLX-card DMA controller to use (default: 0).
  -I          : Use interrupt to receive data (default: polling).
  -0          : Do not display time-out chunkdata (zeroes).
```

Figure 56: List of all fedump features.

```
fec version 17111700
*** UNDER DEVELOPMENT ***
Tool to upload various commands to a GBT-SCA chip via any 2-bit wide, HDLC encoded e-link
including a GBT link's EC channel.
Receive (and display) GBT-SCA replies using option -Z, or use fedump or fdaq.
Usage:
 fec [-h|V] [-d <cardnr>] [-i <dma>] [-I] [-G <gbt>] [-g <group>] [-p <path>] [-t <ms>] [-x <par>] [-C] [-R] [-T] [-Z] [<ops>]
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -i <dma>    : FLX-card DMA controller to use (default: 1).
  -I          : use interrupt to receive data (default: polling)
  -G <gbt>    : GBT-link number (default: 0).
  -g <group>  : Group number (default matches GBT EC 'group' = 7).
  -p <path>   : E-path number (default matches GBT EC 'path' = 7).
  -r <repeat>: Number of GPIO/ADC/DAC operations to perform (default: 1).
  -C          : Send GBT-SCA connect (HDLC control)
  -R          : Send GBT-SCA reset (HDLC control)
  -T          : Send GBT-SCA test (HDLC control)
  -t <ms>     : Time between some of the ops, in ms (default: 100).
  -x <par>    : Parameter to use in operations, e.g. GPIO number, ADC or DAC channel (default: 0).
  -Z          : Receive and display the GBT-SCA replies.
  <ops>       : String of chars indicating which operation(s) to perform:
                o=GPIO-out, i=GPIO-in, a=ADC, d=DAC, I=I2C (no-string=default: none).
Examples:
Blink VLDB (connected to GBT link #3) LED connected to GPIO #18
20 times with a rate of 5Hz (100ms ON, 100ms OFF):
  fec -G 3 -t 100 -r 20 -x 18 o
Read GPIO inputs (GBT-SCA on GBT-link #3's EC-channel) 20 times with a rate of 10Hz:
  fec -G 3 -t 100 -r 20 i
```

Figure 57: List of all fec features.

6.6.4 fuptest

The fuptest tool makes it possible to transfer data from the FELIX data generators (with more fine-grained pattern control than with ()fupload), to defined E-links across a GBT connection. The tool will then expect to receive the same data back via loopback (via the same link or another). The full range of features of the tool can be seen in the help text, as presented in Figure 58.

Note: This tool works with FULL mode firmware versions, but uses GBT links up to the front-end.

6.6.5 fplayback

fplayback is an advanced testing tool (GBT mode only) with the ability to upload data to the FELIX card for transfer out of a the GBT link (multiple E-links) and then receive the same data back before confirming it's integrity by comparison with the original. The data files expected by this tool are the of the .dat format produced by `fdaq`. The tool requires that your FELIX card be connected either with external loopback fibres or for a longer loop via a remote front-end system. In order to use the tool pass the .dat filename as well as optional parameters for amount of blocks to transfer (-F, default all) and to data from a given E-link (-e). For example, to send 500 blocks from 'testfile.dat' for E-link 5 do the following:

```
fuptest version 17022800
Uploads test data to every E-link of the given width of the given FLX-card in turn;
optionally select a single GBT (-G).
Checks whether the e-link is valid and configured on the FLX-card,
unless option -c is given.

Expects to receive the data in a loopback on the same e-link number,
unless option -e is given.
The data is checked byte-for-byte for correctness.

Usage: fuptest [-h|V] [-d <cardnr>] [-b <size>] | [-G <gbt>]
               [-s <bytes>] [-r <chunks>] [-P <patt>] [-w <width]
               [-c] [-e] [-I] [-R]
  -h         : Show this help text.
  -V         : Show version.
  -b <size>  : DMA (cmem_rcc) memory buffer size to use, in MB (default 16, max 4096).
  -c         : Do not check whether e-link is valid on FLX-card.
  -d <cardnr>: FLX-card to use (default: 0).
  -e         : Loopback not necessarily on the same e-link number (default: receive = upload e-link).
  -G <gbt>   : GBT-link number (default: all).
  -I         : use interrupt to receive data (default: polling)
  -P <patt>  : Test data pattern: 0=incr, 1=0x55/0xAA, 2=0xFF, 3=incr-per-chunk (default:0).
  -r <chunks>: Test data chunk count: upload <chunks>*<bytes> bytes of data (default: 30).
  -R         : Do not receive replies (use e.g. 'fedump' for that).
  -s <bytes> : Number of bytes per chunk to upload (default:32).
  -w <width> : E-link width in bits (2, 4, 8 or 16, default:2).
```

Figure 58: List of all fuptest features

```
$ fplayback -e 5 -F 500 testfile.dat
```

For an `fplayback` to be successful there are 2 conditions to be met: firstly, the integrity check should not return any errors (provided the original file is without errors), and the number of chunks (and bytes) per E-link written to the resulting file should be identical to the original file.

For a full list of options for fplayback you can view the help information, as shown in Figure 59.

```
fplayback version 17100900
Upload data from FLX-card data blocks from an 'fdaq'-generated file to an FLX-card.
Usage:
 fplayback [-h|V] [-d <cardnr>] [-F <blocks>] [-i <dma>] [-e <elink>] [<filename>]
  -h         : Show this help text.
  -V         : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -e <elink> : Upload only the data blocks for E-link number <elink>.
  -F <blocks>: Number of blocks to upload (default: all in file).
  -f <speed> : Speed up default upload rate of about 8MB/s by factor <speed> (default:1)
  -i <dma>   : FLX-card DMA controller to use (default: 1).
 <filename>: Name of file with data blocks to upload.
```

Figure 59: List of all fplayback features

6.7 Remote Hardware Command and Configuration Tools

6.7.1 fic

The fic tool is designed for communication with a GBTx chip present on a remote system connected to FELIX via GBT links. Commands are read and written using the IC link component of the GBT protocol. With this tool it is possible to read and write data to a specific GBTx address for the propagation of command and configuration information as part of debugging or preparation of front-end components for a data taking session. For a full list of commands consult the help text, as shown in Figure 60.

```
fic version 17082800
Tool to read or write GBTX registers through the IC-channel of an FLX-card GBT link:
read or write data byte from or to the given GBTX register address
at the receiving end of the given FLX GBT-link or
write to multiple consecutive GBTX registers with data read from a file.
Provide a file name *or* option -a with an address and an optional
additional byte value to read or write a single register.
Without option -a or file name all registers are read out and displayed
either in one IC read operation or one-by-one (option -o).
Usage:
 fic [-h|V] [-d <cardnr>] [-G <gbt>] [-I <i2c>] [-T] [-a <addr> [<byte>] | <filename>]
  -h          : Show this help text.
  -V          : Show version.
  -a <addr>   : GBTX register address (hex) to read or write.
  -d <cardnr>: FLX-card to use (default: 0).
  -G <gbt>    : GBT-link number.
  -I <i2c>    : GBTX I2C address.
  -o          : When reading all registers, do it one-by-one (default: one multi-reg read op)
  -T          : Display time the operation took (in us).
 <byte>       : Byte value (hex) to write to GBTX register <addr> (option -a).
 <filename>   : Name of file with GBTX register data to upload.
Examples:
  fic -G 1 -I 3
  fic -G 1 -I 3 gbtx-config.txt
  fic -G 1 -I 3 -a 34
  fic -G 1 -I 3 -a 34 56
```

Figure 60: List of all fic features.

6.7.2 fgconf

The fgconf tool makes it possible to read and write GBTx registers accessible via i2c through a GBT-SCA chip. For a full description please consult the help output as per Figure 61:

```
fgconf version 17040700
Tool to read or write GBTX registers via an I2C-channel of a GBT-SCA chip,
connected to any FLX-card GBT (2-bit HDLC) E-link:
read or write byte from or to the given GBTX register address or
write to multiple consecutive GBTX registers with the contents of a file.
(ASCII file: 1 byte value (hex) per line,
 e.g. TXT file generated by the GBTX Programmer tool).
Provide a file name *or* use option -a with an optional additional byte value
to read or write a single GBTX register or without option -a to read all registers.
Usage:
 fgconf [-h|V] [-d <cardnr>] [-G <gbt> [-g <group> -p <path>]] [-e <elink>] [-R] [-W]
        -C <ichan> -I <iaddr> -a <addr> [<byte>] | <filename>
  -h          : Show this help text.
  -V          : Show version.
  -d <cardnr>: FLX-card to use (default: 0).
  -G <gbt>    : GBT-link number.
  -g <group> : Group number (default: 7=EC).
  -p <path>  : E-path number (default: 7=EC).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -R          : Reset GBT-SCA.
  -W          : Read writable registers only (default: all).
  -C <ichan> : GBT-SCA I2C channel.
  -I <iaddr> : GBTX I2C address.
  -a <addr>  : GBTX register number (decimal).
 <byte>       : Byte value (hex) to write to GBTX register <addr> (option -a).
 <filename>  : Name of file with GBTX register data to write to consecutive registers.
=> Examples:
Read all registers of GBTX (I2C address 1) connected to GBT-SCA I2C-channel 0,
GBT-SCA connected to FLX-card GBT link 3 EC-link:
  fgconf -G 3 -C 0 -I 1
Read GBTX register 32:
  fgconf -G 3 -C 0 -I 1 -a 32
Write 0xA5 to GBTX register 32:
  fgconf -G 3 -C 0 -I 1 -a 32 A5
Write contents of GBTX-conf.txt to GBTX registers:
  fgconf -G 3 -C 0 -I 1 GBTX-conf.txt
```

Figure 61: List of all fgconf features.

7 Felixcore Application and NetIO

7.1 Operational Principles

The FELIX core application (called *felixcore*) is the central process of a FELIX system. The user interacts with the felixcore application via network endpoints to receive data from E-links or to send data to E-links. Systems such as software RODs, DCS, calibration and monitoring systems all connect with FELIX via felixcore.

Felixcore also supports monitoring of operational data via a web front-end and publishing of E-link information via the FELIX bus system.

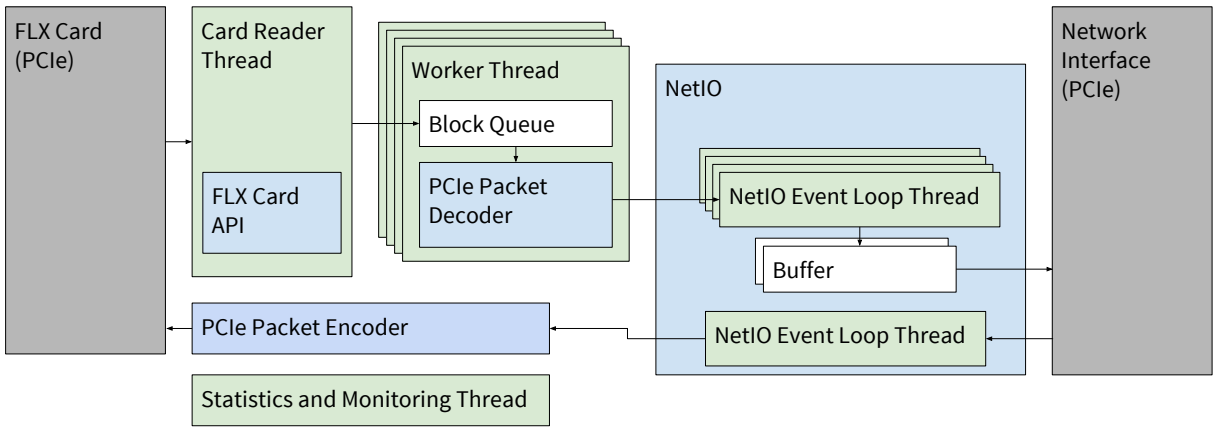


Figure 62: The architecture of the FELIX core application.

Figure 62 shows a diagram of the architecture of the FELIX core application. In the case of a system with multiple FLX cards, one application can be run per card.

7.2 Configuration

The configuration parameters of the felixcore application are listed in Figure 63. The parameters can be passed to the application either on the command line or as part of a file.

```
felixcore - FELIX Core Application

Usage: felixcore [options]

  Run with card: 'felixcore'
  Run with file: 'felixcore -f <file> --notoflx'
  Run with data generator: 'felixcore -g --notoflx'

All Options:

General Options:
  -t, --threads N                       Run with the specified number of threads (0 = number of cores) [default: 0]
  -p, --port N                          Run on selected port number [default: 12345].
  -r, --recv_port N                     Receive slow control data on this port [default: 12340].
  -P, --busy_port N                     Receive busy control signals on this port [default: 12341].
  -B, --netio_backend (posix|fi_verbs)  Use the given NetIO backend [default: posix]
  -l, --logfile <file>                  Write logs to the given filename.
  -b, --buffer N                        Free Buffer size in kBlocks per thread [default: 2000].
  -g, --data_generator                  Use internal data generator.
  --ttc_generator                       Use internal ttc generator elink #0.
  --queue (rwq|tbb)                     Use queue 'rwq' or 'tbb' [default: rwq]
  --nocheck                             Do not check block before dispatch.
  --notoflx                             Do not start the To-FLX thread
  --felix_id <id>                       Elink offset for this FelixCore [default: 0].
  --data_interface <iface>              Interface to publish data [default: default].
  --monitoring_interface <iface>        Interface to publish monitoring information [default: default].
  --noweb                               Disable web server.
  -w, --web_port N                      Port for webserver [default: 8080].

Commandline Options:
  -h, --help                            Display this help.
  -V, --version                         Display version.
  -c, --config <file>                   Load configuration from the specified YAML file.
  --config-write <file>                 Store configuration in the specified YAML file.
  -f, --file <file>                     Use the given filename as input.

Card Options:
  -m, --memory N                        Allocate CMEM memory in Gbyte [default: 2].
  --init                                Execute init only.
  --interrupts                          Use interrupts.
  --poll_time T                         Poll time in us [default: 1000].
  --dma D                               Use dma channel [default: 0].
  --emu_to_host <pattern>               Pattern to switch on emulators to Host [default: 0].
  --emu_from_host <pattern>             Pattern to switch on emulators from Host [default: 0].
  --elinks <elinkrange>                 Elink numbers/number range.

Data Generator Options:
  --fixed_chunks                        Use fixed chunk size.
  --chunk_size_fixed N                  Fixed size of chunks [default: 1018].
  --rate_control_bool                   Use rate control.
  --chunk_size_min N                    Minimum size of chunks [default: 128].
  --chunk_size_max N                    Maximum size of chunks [default: 4096].
  --e_link_min N                        Minimum number of elinks [default: 50].
  --e_link_max N                        Maximum number of elinks [default: 150].
  --e_link_ID_min <id>                  Minimum ID of elinks [default: 1].
  --e_link_ID_max <id>                  Maximum ID of elinks (<2048) [default: 255].
  --period T                            Period in ms [default: 100].
  --max_overshoot T                     Max overshoot in ms [default: 10].
  --duration T                          Duration in s (0 = run forever) [default: 0].

Debug Options:
  --data (real | short | long | mixes | none)  Stream data as given type [default: real].
  --nostats                             Disable statistics.
  --display_stats                       Print out statistics.
  --nohistos                            Disable histograms.
  --trace                               Enable tracing.
```

Figure 63: Command line interface options for the felixcore application. Each option can also be set in the configuration file. The option key in that case is the long option without the leading dashes.

7.3 Monitoring

7.3.1 FelixCore Native Monitoring

The felixcore application provides monitoring and status information via a web front-end. By default the web front-end is available on port 8080. To access the web front-end use a web browser and point it to *http://hostname:8080*. Figure 64 shows a screenshot of the web app.

Figure 64: Screenshot of the integrated felixcore monitoring web app

7.3.2 Monitoring with felix-web-mon

Felix-web-mon is a user oriented web visualisation application for real time monitoring of FELIX metrics through live charts as well as visualisation of historical data. It uses the FELIXbus system to auto-discover FELIX nodes in the same local subnet. Metrics are continuously recorded into a *mongodb* database, making it possible to plot historical charts at a later time. Users should make use of this tool if they require more flexible monitoring with the option of historical archiving of monitoring data.

7.3.2.1 Compilation

Felix-web-mon is currently provided as standalone package. Users wishing to compile it should get the source and instructions from gitlab at the address below.

https://gitlab.cern.ch/atlas-tdaq-felix/felix-web-mon

7.3.2.2 Usage

When installed on a machine (see README.md in gitlab for detailed installation instructions), the web application can be accessed under the address <hostname>:8000. Here users will be presented with a front page as shown in Figure 65.

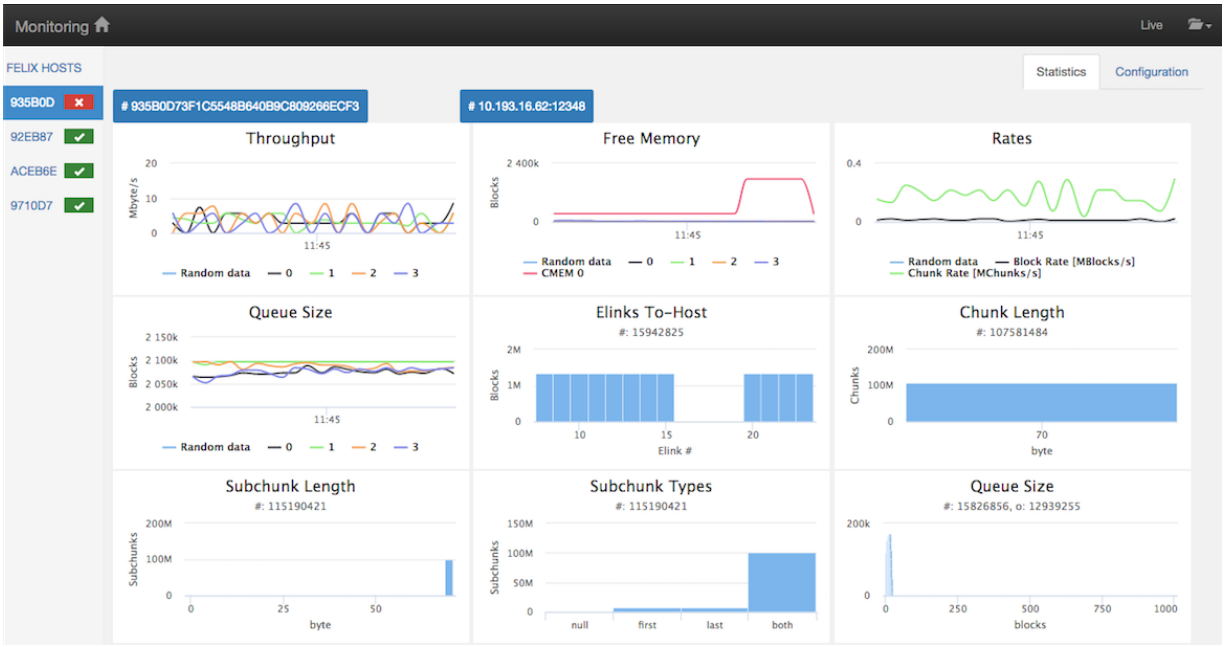


Figure 65: felix-web-mon front page

The user interface consists of the following parts:

- **Menu bar:** as shown on the top left of Figure 65. A simple menu with 2 links: a folder icon for history data plots, and a live link to come back to live mode.
- **Status bar:** as shown in Figure 66. Displays some information like the client-server connection status, the number of connected hosts and the mode 'live' or 'history'. Figure-12: Status bar
- **FELIX hosts list:** as shown in Figure 67. Displays the FELIX hosts that are connected and automatically discovered by the FELIXbus API. An icon next to the name of each item shows the status 'disconnected' (red) or 'connected' (green) of each host.
- **Charts area:** In this panel there are tabs Statistics, Configuration and History (hidden by default) for each connected FELIX host. The Statistics tab (Figure 14), loaded by default, shows live charts of metrics. Clicking on an item in the FELIX hosts list on the left displays the chart area of the corresponding host.
- **The Configuration tab:** showing the configuration data of corresponding FELIX host (Figure 68).
- **The History tab:** shown only when a user clicks to load history data on the folder icon in the menu bar (Figure 69).

Figure 66: felix-web-mon status display

Figure 67: felix-web-mon host list

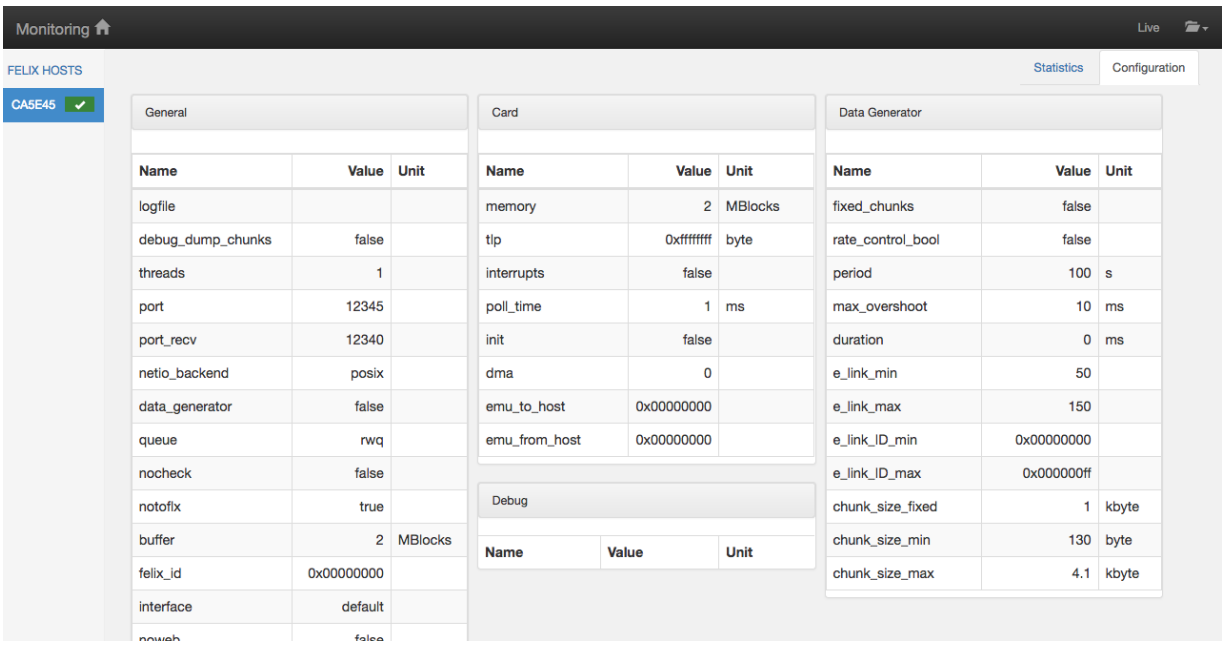


Figure 68: felix-web-mon config page

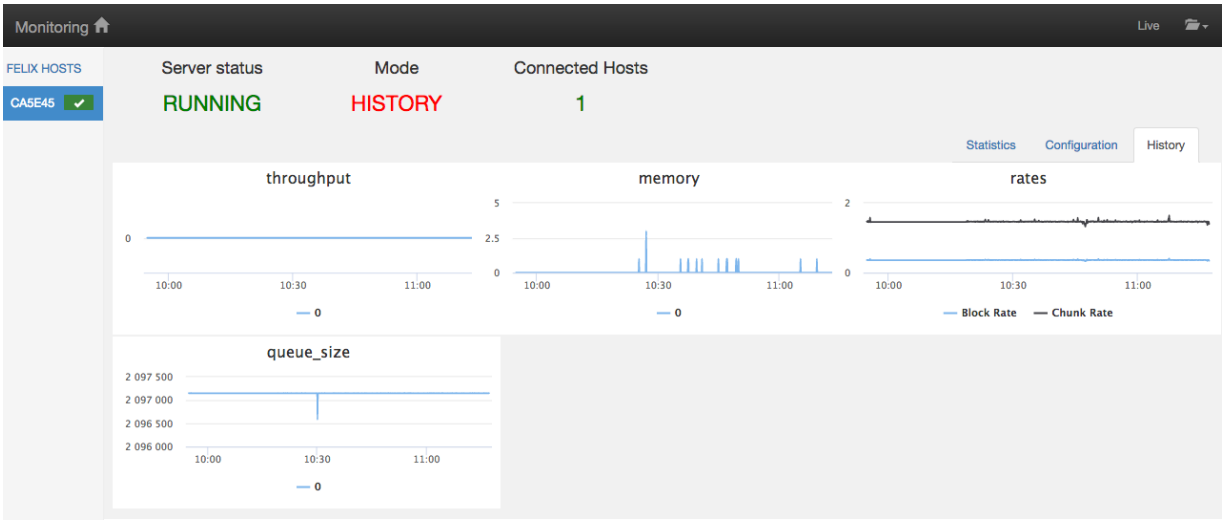


Figure 69: felix-web-mon history page

7.4 FelixCore Examples

When running with multiple threads, FELIX will publish data on multiple TCP/IP ports. The examples are all started with 1 thread using the option `-t 1`. This ensures that all data is published on a single TCP/IP port (default 12345), which facilitates debugging. To read out data from a felixcore application that is started as in the examples below, use a client (for example netio-cat or fatcat), and point it to port 12345 of the FELIX host. Of course, running with only one worker thread limits performance and, depending on the workload, FELIX might not be able to keep up with the load. In that scenario increase the number of worker threads accordingly. The tool `felix-bus-list` can be used to obtain the mapping of E-links to TCP ports of a FELIX system.

A running felixcore instance can be stopped by pressing `Ctrl+\`.

7.4.1 Tests without an FLX Card

- Starting felixcore with input from a file (no card required):

  ```
  $ felixcore -t 1 -f path/to/file.blocks --notoflx
  ```

- Starting felixcore with the internal software datagenerator (no card required):

  ```
  $ felixcore -t 1 --data_generator --ttc_generator --notoflx
  ```

7.4.2 Tests with an FLX Card

- Starting felixcore with one processing thread and emulators enabled. Emulators are configured to send data via PCIe to the FELIX host software.

  ```
  $ felixcore -t 1 --emu_to_host 0xff
  ```

  E-link and emulator data configuration can be adjusted with the `elinkconfig` tool.

- Starting felixcore with one processing thread and emulators enabled. Emulators are configured to send data via detector links to external receivers or via loopbacks back to the FLX card:

  ```
  $ felixcore -t 1 --emu_from_host 0xff
  ```

  E-link and emulator data configuration can be adjusted with the `elinkconfig` tool.

- Send a message to a running felixcore instance that will forwarded to E-link 15.

  ```
  $ felix-dcs -H 192.168.15.1 -e 15 -m "hello world!"
  ```

  Note: The tool felix-dcs is obsolete and will be discontinued in the future. The functionality will be integrated in the new tool `fatcat`.

7.5 Connecting to a felixcore instance using NetIO tools

The felixcore application uses the NetIO publish/subscribe system to distribute data. The tool `netio-cat` can be used to analyze published data. For example,

```
$ netio-cat subscribe --host 192.168.15.2 -t 15 -t 42 -e raw
```

will let `netio-cat` subscribe to E-links 15 and 42 of the felixcore application running on the host `192.168.15.2` with the default port 12345. The encoding is set to *raw*, which will simply write a hexdump of each received message. For other formatting and subscription options, see `netio-cat -h`.

7.6 Connecting to a felixcore instance using FATCAT

FATCAT is an advanced analysis and test client for FELIX and the successor of multiple ad-hoc tools like felix-client and felix-dcs. The application is currently experimental. For help options see

```
$ fatcat -h
```

In the future fatcat can be used to debug data streams coming from FELIX, send data to detectors via FELIX, record data to disk, manage data subscriptions among multiple FELIX hosts, benchmark FELIX systems and analyze recorded data.

7.7 Discovering E-links with the FELIX BUS system

Clients that want to receive data from or send data to specific E-links need to know by which FELIX instances the desired E-links are managed. The FELIX BUS system is used for this purpose. FELIX instances broadcast at regular intervals information about connected E-links. Clients can retrieve this information and build internal lookup tables using the library libfelixbus.

The E-Link IDs published by felixcore are global E-link IDs, i.e. they uniquely identify E-links across all FELIX hosts. This requires that the FELIX ID (command line option `-felix_id`) is set to a unique number for each felixcore instance. The default FELIX ID 0 is fine to use for tests where only a single FELIX is running.

The tool `felix-buslist` can be used to display the tables:

```
$ felix-buslist -t
Tables in FelixBus (ctrl\ to quit)

FelixTable::print(): table.size()=4
PeerId                           FelixID                          Address
1950EDDFD4821AF225D99EBCE9B22152 0468680B8D9D0388D8D1078B725D2E3B tcp://10.193.16.62:12345
1950EDDFD4821AF225D99EBCE9B22152 238B9263BA79A05378652F433A25C949 tcp://10.193.16.62:12348
1950EDDFD4821AF225D99EBCE9B22152 4E418C3646578B46FD939AF6AC5FFB5B tcp://10.193.16.62:12346
1950EDDFD4821AF225D99EBCE9B22152 C87FAE7D74FC1D99BFB454ACD74EBA23 tcp://10.193.16.62:12347

ElinkTable::print(): table.size()=10
PeerId                           FelixId                          ElinkId
1950EDDFD4821AF225D99EBCE9B22152 238B9263BA79A05378652F433A25C949 3735579
1950EDDFD4821AF225D99EBCE9B22152 C87FAE7D74FC1D99BFB454ACD74EBA23 3735580
1950EDDFD4821AF225D99EBCE9B22152 4E418C3646578B46FD939AF6AC5FFB5B 3735591
1950EDDFD4821AF225D99EBCE9B22152 0468680B8D9D0388D8D1078B725D2E3B 3735618
1950EDDFD4821AF225D99EBCE9B22152 238B9263BA79A05378652F433A25C949 3735676
1950EDDFD4821AF225D99EBCE9B22152 C87FAE7D74FC1D99BFB454ACD74EBA23 3735701
1950EDDFD4821AF225D99EBCE9B22152 4E418C3646578B46FD939AF6AC5FFB5B 3735729
1950EDDFD4821AF225D99EBCE9B22152 0468680B8D9D0388D8D1078B725D2E3B 3735732
1950EDDFD4821AF225D99EBCE9B22152 238B9263BA79A05378652F433A25C949 3735739
1950EDDFD4821AF225D99EBCE9B22152 C87FAE7D74FC1D99BFB454ACD74EBA23 3735741
```

The example shows that there is one felixcore instance running (peer ID 1950...) with four threads (FELIX ID 0468..., 238B..., 4E41..., C87F...), each reachable on a separate TCP port. This felixcore instance handles 10 different E-Links which are distributed among the four worker threads.

7.8 Debugging

7.8.1 Using the FelixCore event tracing framework

FelixCore includes a trace framework that can be used to gain a better understanding about latencies added by individual parts of a communication chain or to get detailed information about the dataflow of a single message through the system. The trace framework operates by logging events with precision timestamps to a file.

To enable the trace framework, start the felixcore application with the option `--trace`. FelixCore will then timestamp events and record these events in a file "trace.csv" in the current working directory.

The contents of trace.csv will contain of lines similar to this:

```
20605900,MSG_RECV,0
20606530,FROMHOST_BLOCK_WRITE_START,0
20606584,FROMHOST_BLOCK_WRITE_COMPLETE,0
```

The CSV file contains three columns.

1. The first column contains the timestamp of the event in microseconds.
2. The second column contains the type of the event. Currently we have these events:

   TOHOST_BLOCK_READ Issued when a 1 kB block is read from the FLX card

   FROMHOST_BLOCK_WRITE_START Start of DMA transfer of blocks to the FLX card

   FROMHOST_BLOCK_WRITE_COMPLETE DMA transfer to the card completed

   MSG_RECV A message was received from the network for an E-link

   MSG_SEND A data chunk is published and is being sent to clients in the network
3. The third column contains an E-Link id that associates an event with an E-link if applicable.

Note that the trace.csv file is *not sorted by timestamp*. If needed, the events in the file need to be sorted in a post-processing step.

8 Resources for Front-End Developers

This section is aimed at collecting useful information for front-end developers to aid the design and implementation of front-end firmware/hardware for interaction with FELIX. Useful tips based on experience so far will also be presented, in a section that will grow over time as more feedback is received.

8.1 FELIX Firmware Modules for Front-end Users

The FELIX team have produced a number of self contained firmware modules which are intended for integration into front-end firmware both for testing and production purposes. These will make it possible to test data transfer functionality from the output layer of the front-end firmware to FELIX and beyond, before integrating more of the front-end logic. These modules can be divided up between GBT and FULL mode use cases.

8.1.1 Downloading Firmware Source

A full description (including diagrams) of the modules discussed below, as well as the relevant firmware source, is available on the FELIX project distribution site:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/dist/examples/

The site contains multiple revisions, for compatibility with different FELIX firmware versions. GBT-compatible packages are labelled 'ElinkInterfaceSources' and FULL mode-compatible packages are labbeled 'FullmodeInterfaceSources'. Please consult the documentation within the files for compatibility information.

8.1.2 GBT Test Modules

From a GBT perspective the modules provided depend on whether the GBT implementation is in an FPGA or with a GBTX chip. Common to both is a simple data generator module, which generates an incrementing counter and can be attached to the input port of the GBT module to provide a basic data source for link testing.

8.1.2.1 GBT-FPGA

For FPGA-based GBT a module will be provided to wrap and drive the GBT link in communication with FELIX (in both directions). All modules will be fully compatible with the official GBT-FPGA core [16].

8.1.2.2 GBTx

For GBTx chips all that is needed is to connect the provided data generator to a chip e-port, thus providing data on one E-link across the GBT.

8.1.3 FULL Mode Test Modules

8.1.3.1 Link Layer Tests

For FULL mode implementations the FELIX developers provide a link layer test package, making it possible to verify functionality at transceiver level (e.g clock jitter stability, cleaning and configuration). Users should be able to integrate this into their front-end design for basic tests before implementing higher level link protocols.

8.1.3.2 Protocol Tests

Once the link layer is verified, users can integrate the 'stream controller' module, also provided by the FELIX developers, which manages the FULL mode link protocol and adds e.g. start and end of packet markers. This is recommended for use not just in testing but also final implementation. Alongside this module a simple data generator is also provided which can be used for testing data transfer across the link.

8.2 General Hints and Tips

8.2.1 Known Technical Requirements for FELIX Communication

- E-link "chunks" or packets are even multiples of bytes or 8b/10b symbols. If an odd number of bytes are received from the front end, FELIX will add an extra padding byte. In the to-front end direction, the length must be an even number of bytes.
- Synchronization of 8b/10b encoding requires two consecutive comma characters.

8.2.2 Examples of Design Best Practice based on Current Experience

- For GBT-mode transmission to FELIX, use 8b/10b encoding on the E-links. Avoid the non-encoded fixed length or variable length formats, because no resynchronization is possible if bits are lost or repeated on the E-link. Comma symbols are used to align to *10-bit symbols* in the bit stream. They are considered idles and can be inserted in the data stream anywhere. Transmit "frequent" pairs of commas. This will minimize data loss when FELIX tries to resynchronize when the symbol boundary is lost due to a missed or repeated bit on the E-link.

  Start-of-Packet (SOP) and End-of-Packet (EOP) symbols delineate *packets*, e.g. event boundaries. This allows FELIX to easily resynchronize to event boundaries should synchronization be lost.

  EOP is not strictly required. However, if not present, a long pause in the E-link data will cause the packet just before the pause to wait in a FELIX buffer until the next SOP.

- The E-link clock, input, and output data rates are independent. The only restriction is that within a GBT E-link group all the clocks must have the same frequency, all the data inputs the same data rate and all the outputs the same data rate. However groups can be setup independently from each other. Read the GBTx manual carefully to understand the GBTx group restrictions and bit order. Note, however, that a clock output is only available if its corresponding Tx is enabled. This means, for example, that a bank running with 320 Mb/s E-links can supply only **two** clocks, but they can be 40, 80, 160 or 320 MHz.

- In 8b/10b encoded E-links, FELIX can be asked to assert BUSY by sending BUSY-ON and BUSY-OFF symbols (i.e. out-of-band symbols that can be sent any time, even within data packets). This should be done only in exceptional cases or at start of run. It should not be the normal mode of protecting against buffer overflow. Instead, complex dead time should be defined to prevent most buffer overflows.

- The event data sent to FELIX are not expected to be ATLAS-standard event fragments. FELIX just transports the data to the "Data Handler" (a.k.a. Software ROD) where detector specific software may transform the data as required and format it into ATLAS-standard event fragments for the ATLAS Read out system (ROS).

- In addition to sending all events to the Data Handler, FELIX can send all, or a sample of, events to other network end points for monitoring. Extra monitoring data may be included as packets separate from event data packets in the E-link data stream by using FELIX's stream IDs at the start of the packet.

- DCS information may be included as packets separate from event data packets in an E-link data stream by using FELIX's stream IDs at the start of the packet.

- Any 80 Mb/s E-link can be used to connect to a GBT-SCA ASIC. The E-link clock must be configured to use 40 MHz, i.e. the data is sent in DDR mode.

- The "EC" link can be used as an ordinary E-link at 80 Mb/s; its E-link clock may be either 40 or 80 MHz.

- Fiber connections: *to be added*

- Broadcasts to the front end: *to be added*

- TTC: FELIX can send TTC Level-1 Accept information on any E-link declared as a "TTC" E-link. "TTC" E-links can be 80, 160 or 320 Mb/s E-links, to transfer 2, 4 or 8 TTC bits on every BC clock. The contents of the TTC word is defined by the FELIX configuration and can be chosen from the ten bits in Table 5. Note: In all three cases, the E-link clock can be 40 MHz, i.e. BC clock The data is sent with FIXED latency.

Table 4: Possible TTC bits that can be sent on an E-link defined as a TTC E-link.

Table 5: Below is the list of bits decoded from the TTC system that can be chosen to be sent on an E-link defined as a TTC E-link.

Brcst[7]	Brcst[6]	Brcst[5]	Brcst[4]	Brcst[3]	Brcst[2]	ECR	BCR	B-chan	L1A

"Brcst[i]" are the eight TTC broadcast bits sent on the TTC "B-channel" which the user controls via the TTCvi VME module. Sending the raw B-channel is also an option, sent one bit per BC clock. More options can be requested to suit the needs of specific detectors. Note that currently, the chosen bits are refreshed every bunch crossing by those sent on the fiber from the TTCvi module. This is unlike the legacy TTCrx ASIC where, for some bits (not BCR or ECR), sending the coresponding bit from the TTCvi *toggled* the current state of the coresponding TTCrx ouput bit. In future, FELIX will support both modes of opeation.

As an example, the NSW uses the broadcast bits to send an 8-bit TTC word with the following bits:

L1A, L0A, BCR, ECR, EC0R, test_pulse, soft_rst, SCA_rst

8.2.3 Frequently Asked Questions

- Is GBT wide mode supported?
  Yes.

- Is GBT 8b/10b mode supported?
  No.

- Is the phase of the eight "utility" clocks fixed with respect to the E-link clocks?
  Yes, there is a fixed relationship with the E-Link clocks.
  Note that the eight utility clocks have *worse* jitter than the E-link clocks.

- Can the GBT output a 40 MHz E-link clock, use that clock in 40 MHz DDR mode for the to-frontend link, but accept data on the uplink at 160 or 320 Mb/s? (Assuming the FE ASIC multiplies the 40 MHz to 80, 160 or 320 MHz.)
  Yes, that is possible. Also the to-frontend link can receive at 80, 160 or 320 Mb/s.

- Is there a maximum packet length on the E-link in 8b/10b mode?
  No.

Appendices

A Setting up a TTC System for use with FELIX

This section is meant to help users of FELIX systems with the set-up of a TTC system. It is a work in progress, maintained by Markus Joos (CERN). Figure 70 shows the final cabling of TTCvi and TTCvx modules for a TTC setup with B-channel. The A-channel carries the Level-1 Accept; the B-channel carries BCR and the other TTC commands. The TTCvi-TTCvx pair should have already been tuned. If not, see Section A.1 below. Note: For a TTCex this may look different. A list of all the materials you will require to set up a TTC system is presented in Table 6.

Figure 70: Image of cabled TTC system with B-channel connections

Table 6: Materials needed to set up a TTC system

Item	Source	Remarks
VMEbus crate	Can be rented from the CERN Electronics Pool	Other crates may do as well
VMEbus master	We recommend a SBC from Concurrent Technologies (ATLAS standard). Support can be given for VP717, VP917 and VP-E24	
TTCvi VMEbus card	Can be rented from the CERN Electronics Pool (but the Pool may be out of stock)	The TTCvi is no longer in production. Make sure the VME base address switches are set to match your software.
TTCvx VMEbus card or TTCex VMEbus card	TTCvx and TTCex can be rented from the CERN Electronics Pool (but the Pool may be out of stock)	The TTCvx/ex is no longer in production. The TTCvx has a LED driver, the TTCex has a laser driver
3 LEMO cables (1 or 0.5 ns)		
1 optical multi-mode (TTCvx) or single-mode (TTCex) fiber with ST connectors on both ends		Max length of the fibre: TTCvx: 20 m; TTCex: 100 m
TTCoc	ATLAS (not clear who to ask; Maybe P. Farthouat)	TTC fan-out; needed if you have several FELIX
optical attenuator	Are these available somewhere?	Needed only for use with a TTCex without TTCoc. The optical attenuator has to be a single-mode attenuator of 3-20 dB and has to be connected directly to the TTCex output. The FTPDA-R155 should work with a TTCvx without attenuator. In case of a TTCex an attenuator of 3 dB is recommended for the FTPDA-R155. The FTPDA-R155 has a sensitivity of -31 dBm and saturates at +1 dBm.

If you need to tune the TTCvi-TTCvx pair, you need in addition:
2 LEMO cables (5-10 ns)
2 LEMO Y-adapters
2 LEMO-BNC adapters
2 50 Ohm terminators (Only required if your oscilloscope has no internal termination.)

A.1 Tuning a TTC system

If your TTCvi-TTCvx pair has not been tuned, follow the instructions in this section. Cable the TTC system as shown in Figure 71. Note: for a TTCex this may look different. For more information please consult the section "Tuning procedure 2" of the TTCvi manual (http://www.cern.ch/TTC/TTCviSpec.pdf).

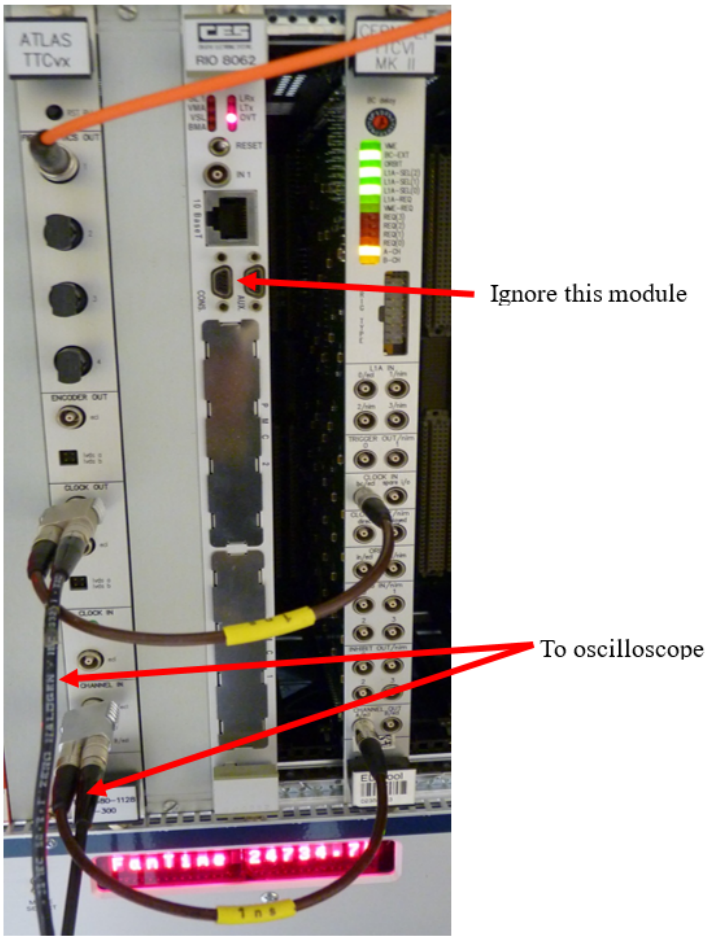


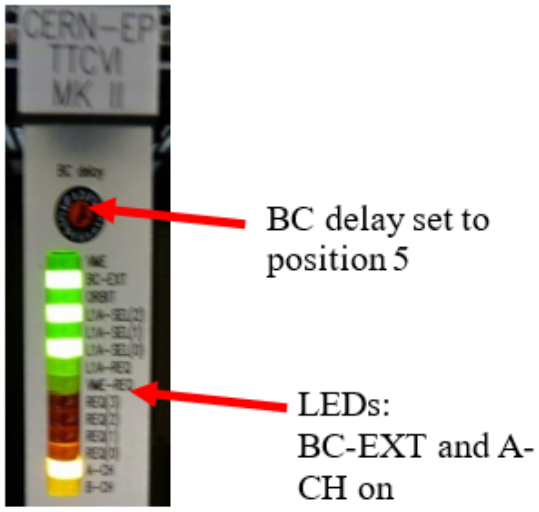
Figure 71: Image of cabling for tuning a TTC system

Note: The question has come up if channel A and channel B are correctly cabled in the picture above. Here is a reply from the TTC expert (Sophie Baron):

> A "good" configuration when channel B is not used is indeed to have it tied to "1". And it is right that having Channel B connected to OUTPUT B gives a static "1" on channel B. However, the termination scheme at the TTCex inputs keeps as well unconnected channel inputs (both B and A) to "1" by default (it is negative ECL logic, and the Vin is at -2.08V by default). Therefore, both schemes could be used identically. One additional remark: of course, if you leave both A and B unconnected at the input of the TTCex, you will have both channels A and B to "1", and this is not good as the TTCrx needs to see two different behaviours on A and B to be able to differentiate them (the rule is that the A-channel must not have more than 11 consecutive "1" whereas B can have any type of sequence).

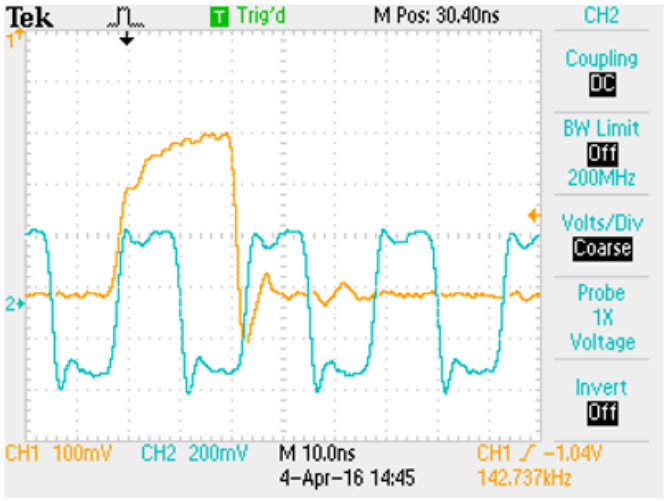
This description can be broken down into the following points:

1. Connect the TTCvi **A/ecl CHANNEL OUT** output to the TTCvx **A/ecl CHANNEL IN** input via a Y-adapter.
2. Connect, via a Y-adapter, one of the **TTCvx CLOCK OUT/ecl** outputs to the **TTCvi CLOCK IN bc/ecl** input. Check that the BC_EXT indicator is lit on the TTCvi as shown below. The TTCvx internal clock may be used.



3. Set the TTCvi trigger mode (= 5) to random at the highest rate (100 kHz) and disable the event/orbit/trigger-type transfers. In order to do this write 0x7005 to the D16 VMEbus CSR1 register at offset 0x80. This can be done easily for vme_rcc_test. Note: The A24 base address of the TTCvi in the CERN reference system in TBED is 0x555500. This should light up the TTCvi A-Ch yellow indicator and the A/ecl CHANNEL OUT output should now carry 25 ns long trigger pulses.

4. With an oscilloscope look at the TTCvx Channel-A input in respect to the clock output, as shown below.



CH1 (yellow): Channel in
CH2 (blue): Clock out

5. Adjust the TTCvi BC delay switch such that the rising edges of the Channel-A pulses occur within 4 ns before to 2 ns after the rising edges of the clock signal.

6. Setting the delay switch in position 2 and using 1 ns long interconnecting cables for the clock and the A and B channels corresponds to the above mentioned timing criteria. Note of MJ: Even though I used 1 ns cables, I had to set the switch to position 5 (see picture above) in order to meet the requirement of step 5.

A.2 Guide to TTC Channel B

The following section describes the structure of the TTC 'B channel' data stream, and how it may be decoded and operated by users. The information in this section is provided courtesy of Alessandra Camplani and the LAr group.

The data stream arriving through TTC B channel can be of two types: short broadcast commands or long individually-addressed commands/data.

Short broadcast commands are used to deliver messages to all TTC destinations in the system, while long individually-addressed commands/data are used to transmit user-defined data and instructions over the network to specific addresses and sub-addresses. These two types of command have different dedicated frame formats, as shown in Figure 72:

The difference between the two command types can be illustrated with the example below. When not in use the B channel IDLE state is set to 1. When a sequence of commands is sent, the data transmission state changes from 1 to 0. After the first zero received it is possible to distinguish between short broadcast and long address commands: if the second bit in the stream is a 0 then the command is a short broadcast, if it is a 1 then the command is of long address type.

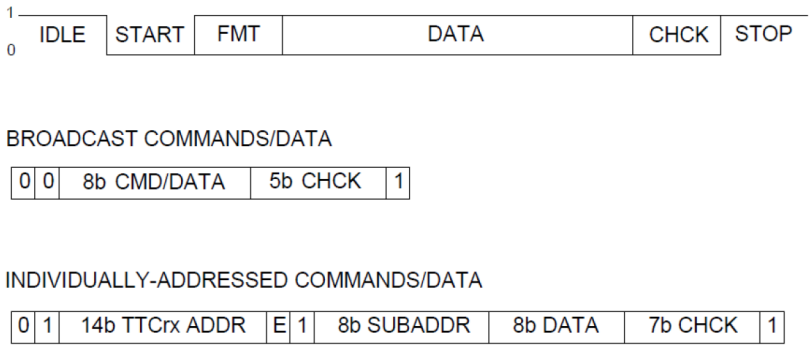


Figure 72: Frame formats for the B channel commands.

```
IDLE=111111111111
    Short Broadcast, 15 bits:
    00TTDDDDEBHHHHH:
        T= test command, 2 bits
        D= Command/Data, 4 bits
        E= Event Counter Reset, 1 bit
        B= Bunch Counter Reset, 1 bit
        H= Hamming Code, 5 bits
    Long Addressed, 41 bits
    01AAAAAAAAAAAAAAE1SSSSSSSSDDDDDDDDHHHHHHH:
        A= TTCrx address, 14 bits
        E= internal(0)/External(1), 1 bit
        S= SubAddress, 8 bits
        D= Data, 8 bits
        H= Hamming Code, 7 bits
```

The short broadcast command type is used to send two important values: the Bunch Counter Reset (BCR) and the Event Counter Reset (ECR).

The BCR is used to reset the bunch crossing counter, which is increased every clock cycle on the 40 MHz clock. This is a 12-bit counter, also called BCID. A BCR command is sent roughly every 89 μs, corresponding to the time that a bunch needs to do an entire circuit of the LHC. During this time the BCID counter reach its maximum value, 3564 counts.

The ECR is used to increase the event reset counter. The periodicity of this reset is decided by each experiment, with ATLAS having it set to 5 seconds. The event reset counter combined with the L1A counter gives the Extended L1ID (EVID). This is a 32-bit value consisting the L1A counter in the lower 24 bits, and the event reset counter in the upper 8. Every time that an ECR is received the upper counter is increased by 1 and the lower part is reset to zero. Every time that a L1A is received the lower part is increased by 1.

BCID and EVID values are used as a label for the data accepted by the trigger.

The long address command type is used to transport another important value: the Trigger Type (TType). Each L1A transmission is followed, with variable latency, by an 8-bit TType word. This word is generated

inside the LVL1 Central Trigger Processor (CTP) and distributed from the CTP to the TTCvi modules for each of the TTC zones in the experiment via the corresponding LTP modules.

The presence of a Trigger Type within long address commands is announced by a sub-address (8 bits) set to 0.

Sub-Trigger	physics	ALFA	FTK	LAr demonstrator	Muons	Calorimeter	ZeroBias	Random
Bit	7	6	5	4	3	2	1	0

Table 7: Trigger type 8-bit word: Each bit represent the sub-detector which fired the trigger or the data type.

As shown in Table 7, each bit has a specific role. In calibration mode, bits 0 to 2 can be used to distinguish between up to eight different possible types of calibration trigger within each sub-detector. Bits 3 to 6 are used to indicate which sub-detector or subsystem fired the trigger. Bit 7 represents physics trigger-mode when set to 1, and calibration mode when set to 0.

A.2.1 B channel decoding firmware

An effort is under way to provide a centrally maintained firmware module to decode TTC B-channel data. In the short term, users are advised to refer to a version produced for LAr front-ends by Alessandra Camplani. The module code can be found in gitlab:

`https://gitlab.cern.ch/atlas-lar-ldpb-firmware/LATOME-ttc`

The code itself is in the folder *code_ttc* and the files dedicated to TType decoding are: *Bchan_top.vhd*, *SMdecoding_cnt.vhd* and *TType_decoding*. The simulations for this specific part can be found in the *simulation* folder. Here there is a testbench for the *Bchan_top* entity and another one for *TType_decoding* entity.

Development of this module is ongoing, with the *latome_ttc* branch being actively maintained and kept up-to-date.

A.2.2 Channel B decoding software

In order to test channel decoding, it is recommended that users employ the menuRCDTtcvi application, provided as part of the ATLAS TDAQ software release. Within the application select 'BGO menu' and then option 13 'send asynchronous command'. From here it should be possible to select either a short of long command. In the case of a short command simply enter the data word to be sent. For a long command enter an address 0 (for broadcast), 0 for internal registers, subaddress 0 for trigger type, and the data word to be sent.

A.3 Useful documents

You may find additional useful information in this document from the ATLAS LAr group:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/community/CPPM_MiniFELIX_tests_results_and_TTC_system_experience.pdf

B BNL-711 Technical Information

This appendix will collect technical information for the BNL-711 board which may be relevant to user test stand installations.

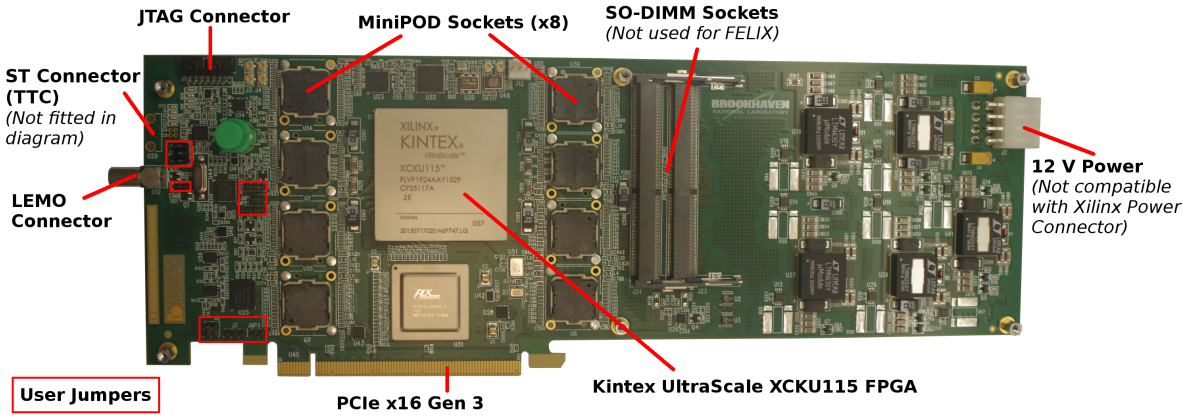


Figure 73: The BNL-711 V1.5 board.

B.1 User Jumper Map and Functional Specification

The BNL-711 provides a number of I/O connectivity options. These are selectable by modifying the position of the user jumpers shown by the red boxes in Figure 73. A specific map of the relative position and name of each jumper is presented in Figure 74. A detailed description of the function of each jumper, and to which board configuration options it relates, is available in the sections below.

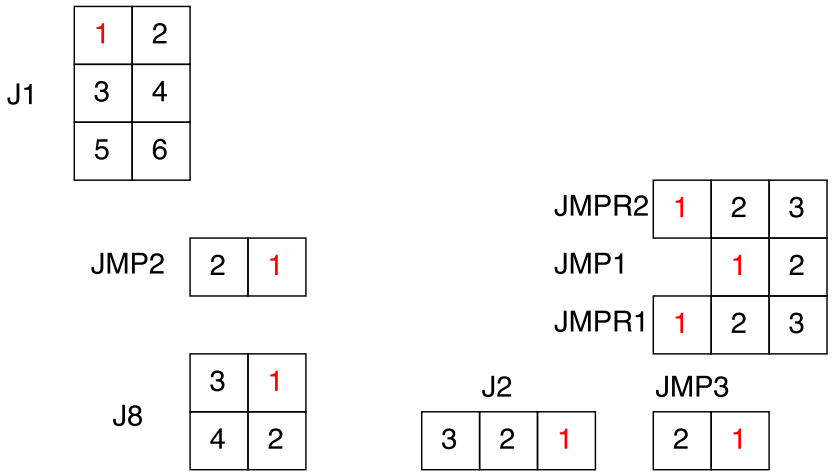


Figure 74: BNL-711 V1.5 User Jumper Map

B.1.1 J1

For uC configuration with 6-pin ISP programmer.

1	MISO
2	VTG-SYS25
3	SCK
4	MOSI
5	RSTn
6	GND

B.1.2 J2

PRSNT selection.

1	PRSNT_FPGA
2	PRSNT
3	PRSNT1

2&3 are connected.

B.1.3 J8

Backup I2C/SMB connector.

1	SYS33
2	PCIE_SCL
3	GND
4	PCIE_SDA

B.1.4 JMP1

Connect FPGA_PROG_B to the uC_FPGA_PROG_B (PC5). Connected by default.

1	uC_FPGA_PROG_B
2	FPGA_PROG_B

B.2 JMP2

Connect FPGA_INIT_B to the uC_FPGA_INIT_B (PD2). Connected by default.

1	uC_FPGA_INIT_B
2	FPGA_INIT_B

B.2.1 JMP3

WAKE_N from PCIe to FPGA. NOT connected by default.

1	PCIE_WAKE_N
2	PCIE_WAKE_N_FPGA

B.2.2 JMPR1 & JMPR2

JMPR1: FLASH_A25 selection.

1	GND
2	uc_FLASH_A25
3	SYS25

JMPR2: FLASH_A26 selection.

1	GND
2	uc_FLASH_A26
3	SYS25

The FPGA firmware has the highest priority to set FLASH_A25 and FLASH_A26. The Jumpers have lowest priority.

If uC is used, when uC_FLASH_A is '1', the FLASH_A will be '0'; when uC_FLASH_A is '0', the FLASH_A will be '1'.

If Jumpers are used, when it is connected '1', the FLASH_A will be '0'; when it is connected '0', the FLASH_A will be '1'.

As default, the first Flash partition can be used, then 2&3 are connected.

B.3 MiniPOD Connectivity Map

The BNL-711 hosts 4 MiniPOD Tx/Rx Transceiver pairs, located in two banks either side of the FPGA, as shown in Figure 73. The transceiver sockets have a specific logical order, presented in Figure 75. Users should connect their optics according to this map to ensure the correct link order is preserved.

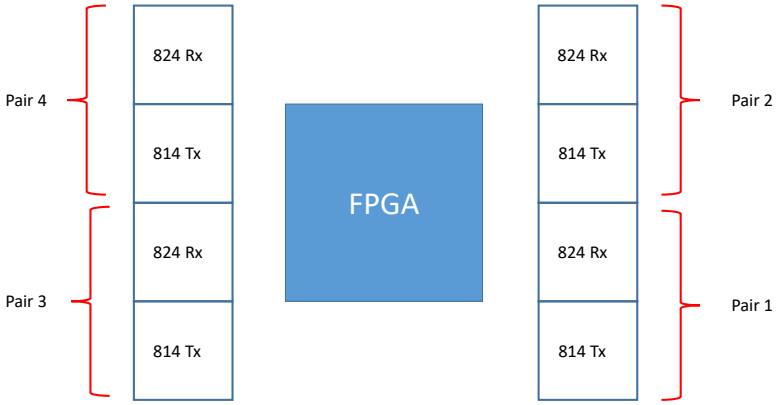


Figure 75: BNL-711 MiniPOD Connectivity Map.

C Guide to FELIX Data Structures

- Data buffered in the FPGA per E-link or per FULL mode link and transferred under DMA control
- Fixed block size of 1 kB
- The blocks are transferred into a contiguous area, functioning as a
- circular buffer, in the main memory of the PC.
- The DMA runs continuously, thereby eliminating DMA setup overheads and achieving high throughput (about 12 GB/s for the 16-lane interface of the FLX-711).
- Event fragments or other types of data arriving via the FE links are referred to as “chunks” and can have an arbitrary size.
- 1 kB blocks of E-links or FULL mode links are multiplexed into a single stream.

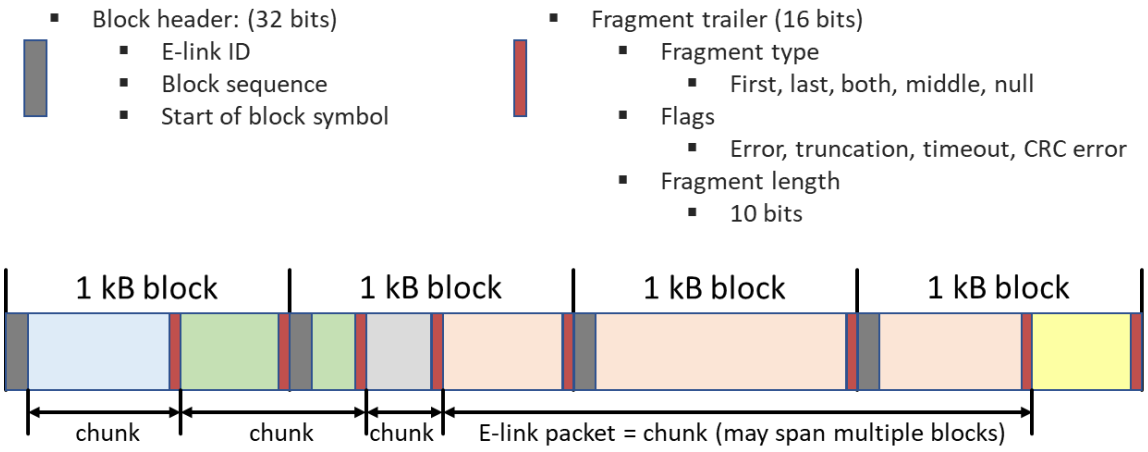


Figure 76: FELIX block structure

From FPGA to PC

Block header

Data is transferred as 1 kByte blocks with this block header

Block header as u_long (32-bit word) read from memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xab								0xcd								Sequence number								GBT						E-group	E-path

Block header as u_char (bytes) read from memory

7	6	5	4	3	2	1	0
GBT-low			E-group		E-path		

7	6	5	4	3	2	1	0
Sequence number							GBT-high

7	6	5	4	3	2	1	0
0xcd							

7	6	5	4	3	2	1	0
0xab							

Block header as u_int_16_t (u_short) (16-bit words) read from memory

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sequence number								GBT				E-group		E-path	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xab								0xcd							

Sub-chunk trailer

Sub-chunk trailer read as u_int_16_t (u_short) (16-bit word) from memory

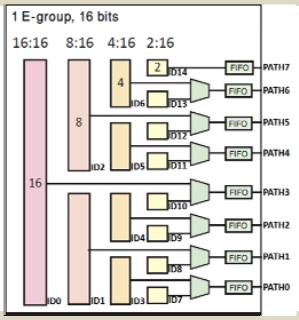
Payloads of the blocks

are organized as chunks that may extend over several blocks. Each chunk has a 16-bit trailer, if it extends over more than one block then each part of the chunk ("sub-chunks") has its own 16-bit trailer

Type			T	E	C	sub-chunk length in bytes	
0	0	0					<- signals fill pattern caused by timeout, all other bits are also 0
0	0	1					<- first part of chunk consisting of more than one part
0	1	0					<- last part of chunk consisting of more than one part
0	1	1					<-chunk consisting of one part
1	0	0					<-not first and not last part of chunk consisting of more than one part
1	0	1					<- E-link timeout
1	1	0					<- reserved
1	1	1					<- out of band

on a 16-bit boundary, if the number of bytes is odd, a padding byte is added at the end

T: Truncation flag, 1 if chunk was truncated
E: Error flag, 1 if an error occurred
C: CRC20 Error (FULL mode) Reserved in GBT mode



Out of band trailers (implemented in Full mode only)

Out of band trailers start width "111", but the other fields are defined here, the fields T, E, C and length are undefined.

1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 Indicates that the frontend is in busy state for the given Full mode link 0xE05C

Default emulator chunk payload

Internal header of chunk payload as loaded in emulator memory in .bit files. This payload can be replaced by the elinkconfig program

Internal chunk header as u_char (bytes) read from memory

byte 0

7	6	5	4	3	2	1	0
0xaa							

byte 1

7	6	5	4	3	2	1	0
0							

b

7	6	5	4	3	2	1	0
length msb							

byte 2

7	6	5	4	3	2	1	0
length lsb							

byte 3

7	6	5	4	3	2	1	0
L1ID msb							

msb: most significant bits
lsb: least significant bits

byte 4

7	6	5	4	3	2	1	0
L1ID lsb							

byte 5

7	6	5	4	3	2	1	0
0xbb							

byte 6

7	6	5	4	3	2	1	0
0xaa							

byte 7

7	6	5	4	3	2	1	0
E-link width							

TTC: L1Ainfo

Proposal for TTC information ("L1AInfo") output via dedicated E-link as chunk payload, Phase-II compatible, FMT=0x1 for format shown

Words read as u_long (32-bit word) read from memory

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FMT								length (20)								reserved								BCID							

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
extended L1ID								L1ID																							

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
orbit																															

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
trigger type																reserved															

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
L0ID																															

Words read as u_long (32-bit word) read from memory

Proposal for TTC information for testing, emulated using emulator memory

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FMT								length (8)								reserved								BCID							

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
extended L1ID								L1ID																							

L1ID typically about 0 .. 25

From PC to FPGA

Data is transferred as 32 byte messages with a 2-byte header and a 30-byte payload

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GBT					E-Group			E-path				length			eom

eom: 1 if last part of message
length: count of 2 byte words

Figure 77: List of FELIX Data Structures

D Guide to Using FELIX with GBT-SCA

This appendix is included with thanks to Paris Moschovakos and the DCS team.

D.1 Introduction

The Slow Control Adapter (GBT-SCA)ASIC is part of the GBT chip-set and it is dedicated for the slow control of the front-end boards. It features several sub-devices that facilitate both Frornt end configuration and monitoring environmental variables (voltages, temperatures, etc.) on and around the detector. The sub-devices are ADCs, DACs, general purpose IO, and controllers for I2C, SPI and JTAG. An SCA is connected to a GBTx via any 2-bit E-link. The E-link must be operated in 40 MHz DDR mode (80 Mb/s) with HDLC encoding. Up to 41 SCAs can be potentially connected to a single GBTx when FELIX is configured accordingly.

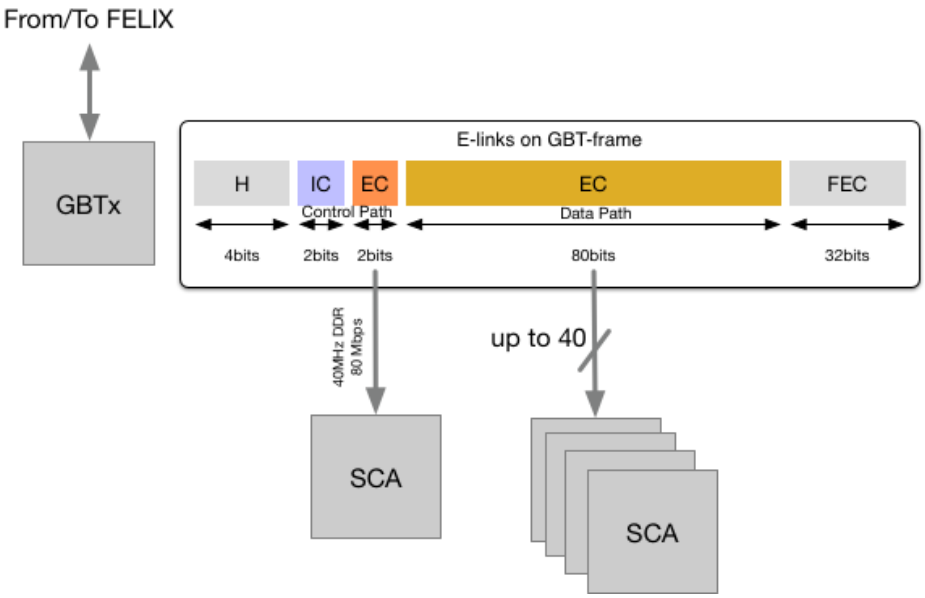


Figure 78: GBT frame paths and e-links

D.2 Typical test setup

A typical test setup consists of a board that houses a GBTx that is connected to FELIX via an optical fibre and to a GBT-SCA ASIC via an e-link.

A Versatile Link Demo Board (VLDB) (https://espace.cern.ch/GBT-Project/VLDB/default.aspx) was designed that can be directly plugged into a FELIX board and hosts both a GBTx and an SCA. (The VLDB demo board can be procured from the GBT group.) Such a setup is shown in Figure 79.

To simplify the evaluation of the setup, the VLDB possesses two LEDs which are connected to two general purpose outputs of the SCA. This can be used to validate the functionality all the way from the FELIX

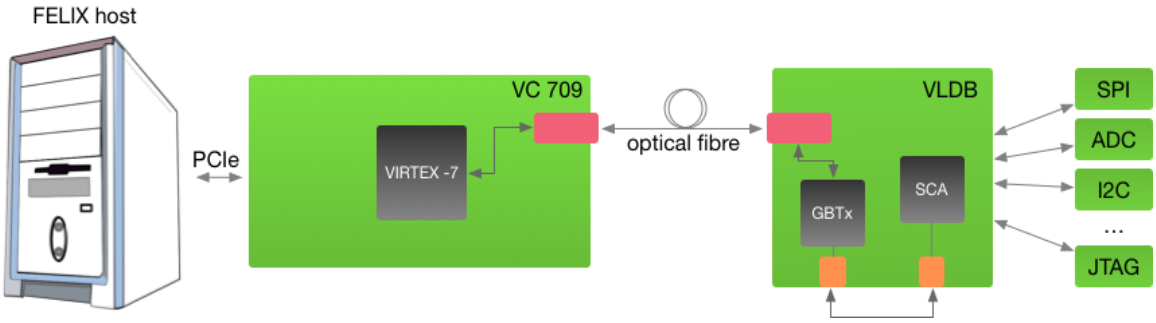


Figure 79: Evaluation setup with an SCA on VLDB. SCA and GBTx are interconnected externally in VLDB via mini-HDMI connectors J32 (PRIMARY) and J33 (SCA PORT)

host to the SCA itself. In order to do that, the "fec" tool, of the ftools family as mentioned in Section 6.6.3, can be used.

The following line requests from the SCA at <gbt_link_number> where the VLDB is connected, to blink its LED 100 times.

```
$ fec -G <gbt_link_number> -r 100 -x 18 o
```

D.3 Operation to set up an SCA e-link

A configuration procedure is needed both for FELIX and the GBTx itself. The configuration is mostly a description of the setup at hand and the mapping of the e-links that are connected. There are also some setup specific parameters to be configured.

In the case where the SCA is connected to the dedicated EC e-link, one should just check that it is enabled via the elinkconfig GUI. That specific e-link is pre-configured with the appropriate HDLC SCA encoding and corresponding bit endianness and can be used directly for any SCA.

In the special case that one wants to use one of the data path e-links, one should configure FELIX via elinkconfig accordingly. SCA uses HDLC encoding instead of the typical 8b/10b which is the standard for the data e-links as can be seen in Figure 81. Moreover, the bit orientation is different than the other data e-links. By selecting the HDLC format in the drop-down menu, elinkconfig takes care both for the orientation and the encoding, indicating that an SCA is connected to that specific GBT group and path.

D.4 Low level operations with fec tools to configure and establish basic communication

The fec tool, as described in Section 6.6.3, is the dedicated "ftool" to use for the SCA EC e-link handling. A number of operations to the various SCA interfaces is possible using its arguments. Depending on your setup please check the full list of possible operations at Figure 57.

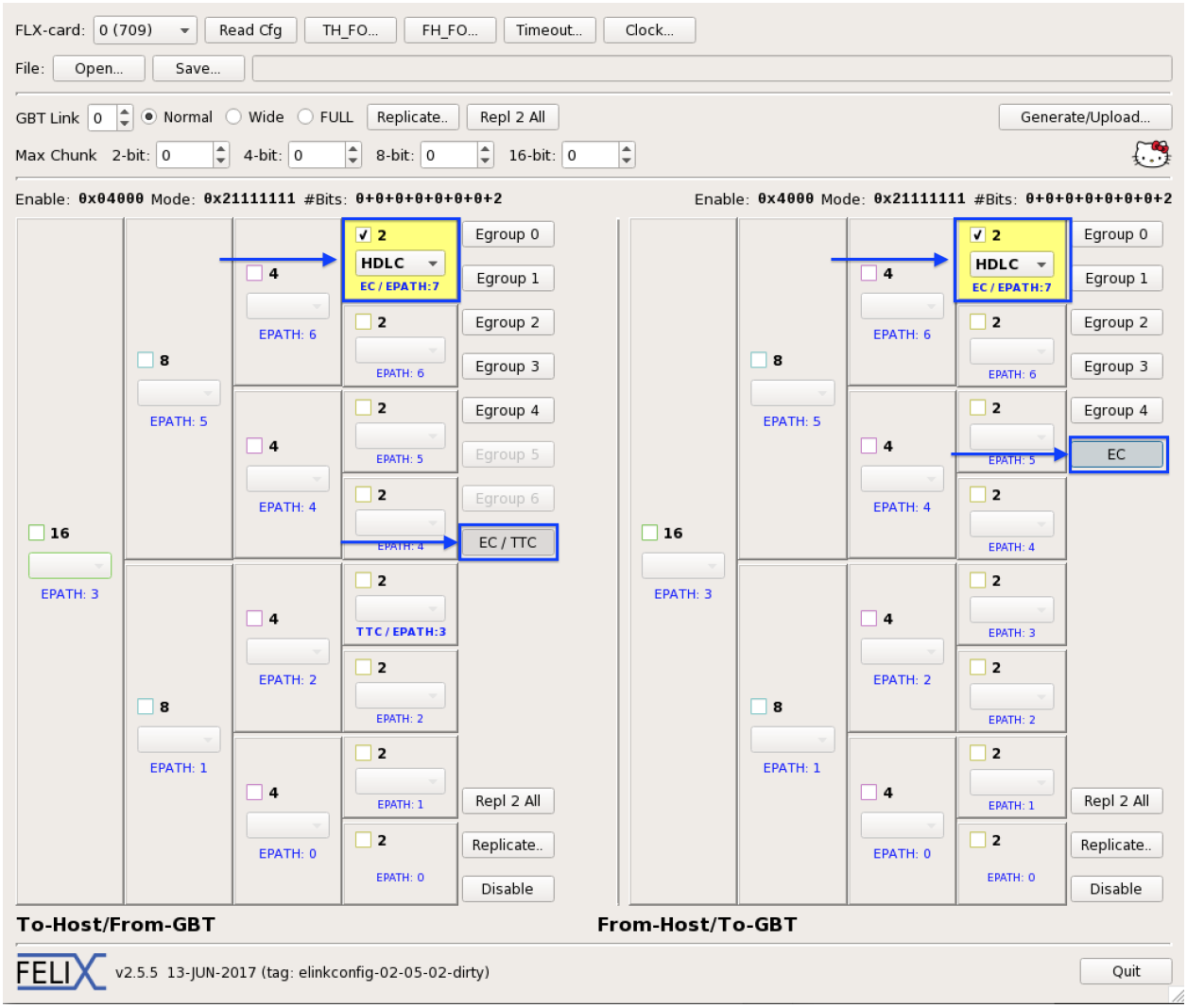


Figure 80: SCA EC e-link

D.5 The integrated production system – Introduction

The on-detector DCS system that handles the slow control traffic and the configuration of the front-end electronics, based on the GBT-SCA, is part of FELIX ecosystem and closely integrated into it. A proposed solution is presented on the following scheme. The slow control and configuration traffic, unlike physics data, has different requirements in terms of throughput, latency, availability and reliability.

FELIX DCS is the software that handles the SCA traffic arriving at FELIX card. Towards the FELIX clients it is based on the middleware Open Platform Communications Unified Architecture (OPC UA, of the OPC foundation, (https://opcfoundation.org/) which is an industry standard for secure and reliable exchange of data in industrial automation and other controls-related areas.

The server/client architecture that the platform uses, allows for different purpose clients to be served by a single server per FELIX host. The data flow to/from the ATLAS control room, not only serves the control and monitoring data of the detectors' conditions but also implements the configuration path of the

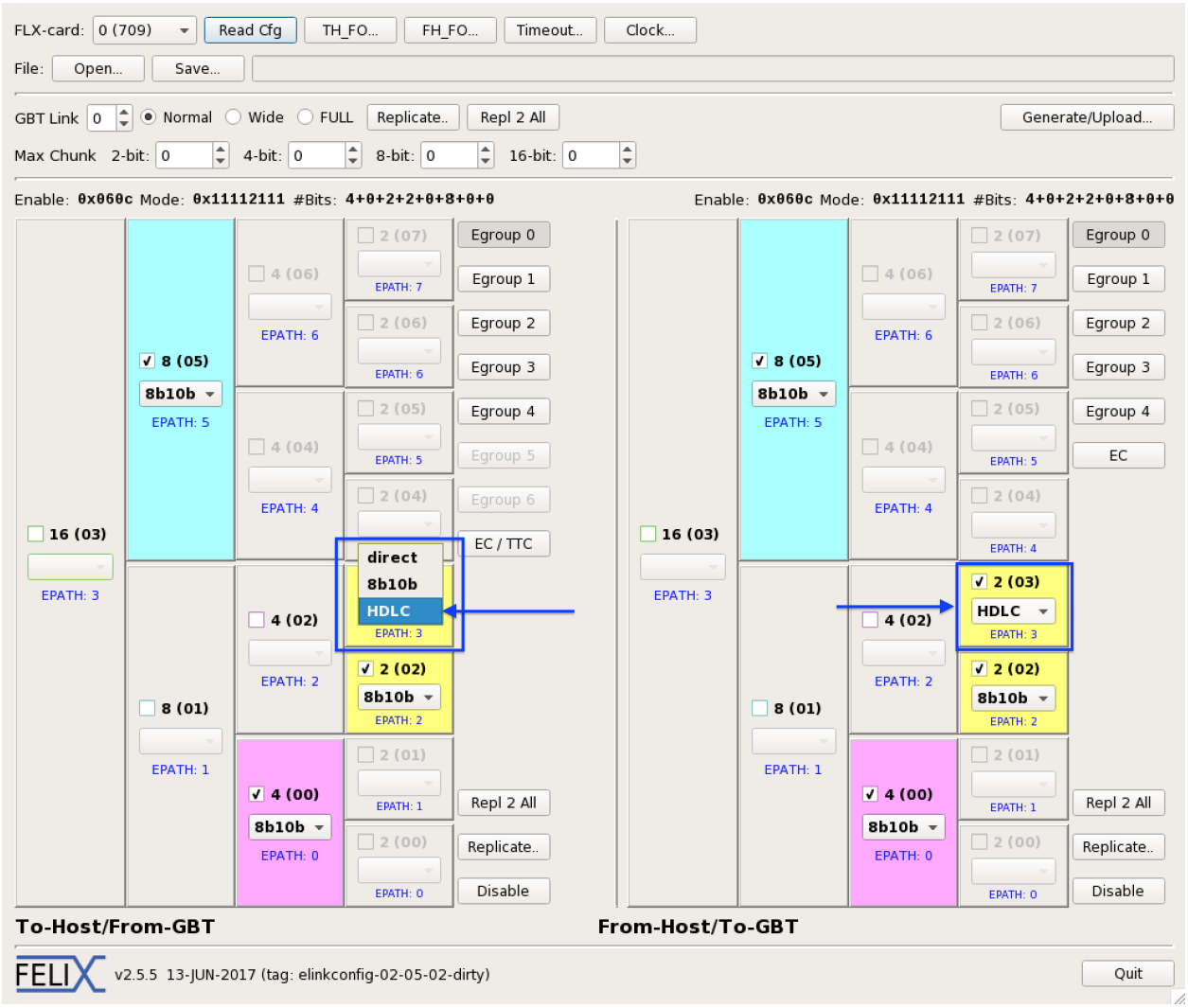


Figure 81: SCA HDLC encoding

on-detectors electronics and their initialization for data taking or calibration. In addition, system experts can monitor the status of the employed technology and get statistics and other information in order to diagnose the various system layers.

All those requirements potentially imply many different OPC UA clients that would like to receive SCA data from the setup at the same time. The chosen OPC UA architecture will ensure the reliable and seamless data delivery and the compatible integration into the current DCS systems. This means that OPC clients in both DCS and a detector configuration server can communicate with the same SCA ASIC and the OPC server will correctly arbitrate their access.

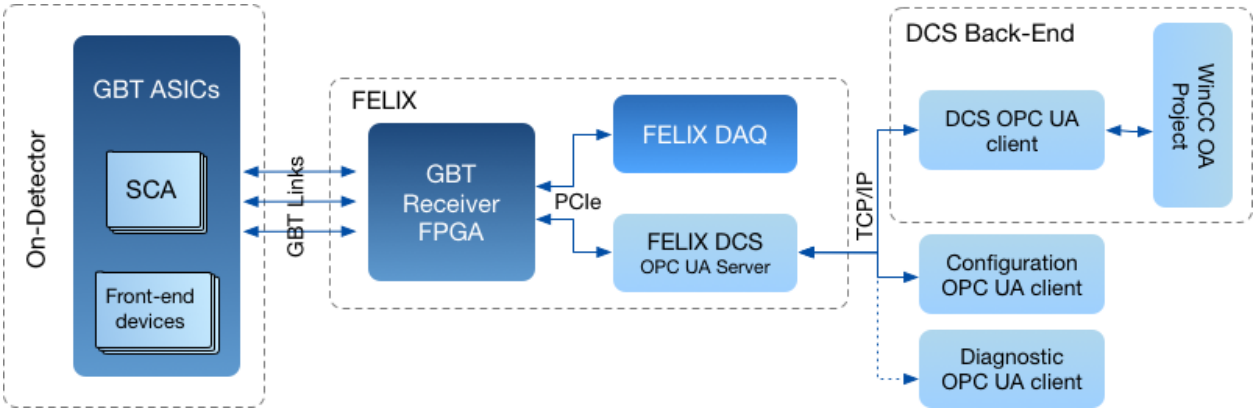


Figure 82: OPC UA for GBT-SCA Architecture

D.6 The SCA software (SCA-SW), its demonstrators and the OPC-UA SCA Server

D.6.1 SCA-SW library

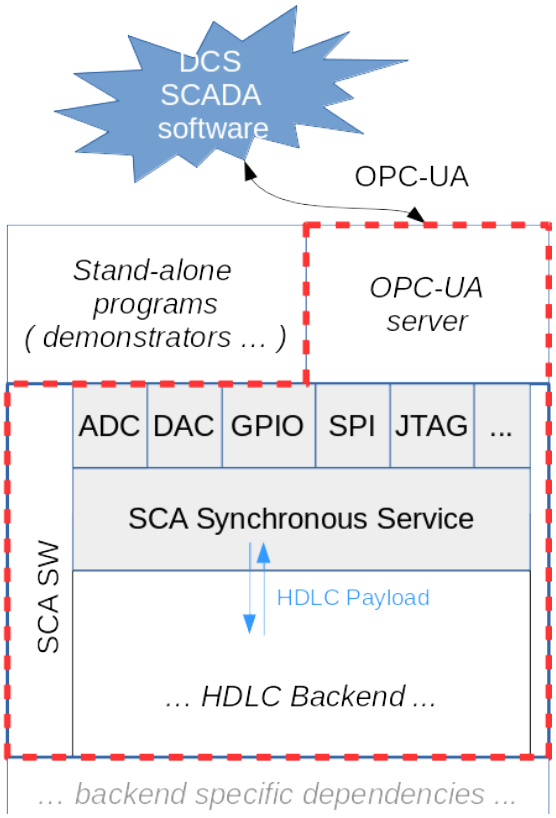


Figure 83: SCA-SW Library

A software library called SCA-SW has been designed and implemented. Its purpose is to provide software support for the SCA chip by providing a high-level programming interface (for example, a function that invokes an SCA ADC conversion and returns the converted voltage as a float number).

The following key decisions have accompanied the design process:

- The library should be a modular piece of software supporting SCA chip(s) no matter how it is physically connected to the host system. Therefore the core part of the library operates on protocol data units of the SCA chip (here picture to SCA frame format), which normally would be encapsulated in the HDLC protocol. There are a number of predefined "HDLC backends" which are services to send such encapsulated SCA requests and receive replies.
- The library should scale from the simplest use cases up to scenarios of thousands of SCAs.
- The library should be able to profit from concurrency features of the host system, including mutli-core and multi-threaded operation.
- The library should be written in a chosen version of the standard C++ dialect.
- The library should be designed with reliability and robustness as a key design choice because it would serve critical, 24/7 communication.

The out-of-the-box SCA-SW, as of October 2017, includes among its backends the NetIO backend which enables seamless communication with the FELIX software ecosystem, and particularly with the felixcore application which can route the traffic between an application based on SCA-SW and any SCA connected through fibers to chosen FELIX machine.

Being backend-agnostic, the addressing scheme shown in Table 8 has been chosen for the library to identify a given SCA in case of NetIO.

Table 8: Addressing scheme

Backend type	Discovery variant	SCA address to use
NetIO	FELIX mapper not used	simple-NetIO://direct/hostname/port1/port2/elink Where: 1. Hostname is a hostname of the FELIX machine handling given traffic 2. Port1 is the TCP/IP port on which FELIX will receive and transport further through fibers to the SCA. Typically it is 12340. 3. Port2 is the TCP/IP port on which FELIX will distribute the replies of the SCA chip. Typically it is 12345. 4. E-link is a two-digit hexadecimal E-link identifier. For example, 3F would mean the EC link of the first fibre of the FLX card. You can use "felink" tool to compute the E-link identifier. Note that neither decimal format nor prefix/suffix are supported (e.g. it's illegal to put 0x3f instead of 3F).
	FELIX mapper used	To be defined later

The SCA-SW provides a number of demonstrators for various SCA components, like a demonstrator to print out ADC conversion results, program a VMM3 chip through SPI, etc.

D.6.2 SCA OPC-UA server

The provided OPC-UA server implementation for the SCA is based on the SCA-SW (explained above) intending to profit from all features of the SCA-SW library and providing a high-level and user-friendly OPC-UA address space to OPC-UA clients.

The OPC-UA server has been designed and implemented using the quasar framework (see https://github.com/quasar-team/quasar). Its design is presented in Figure 84.

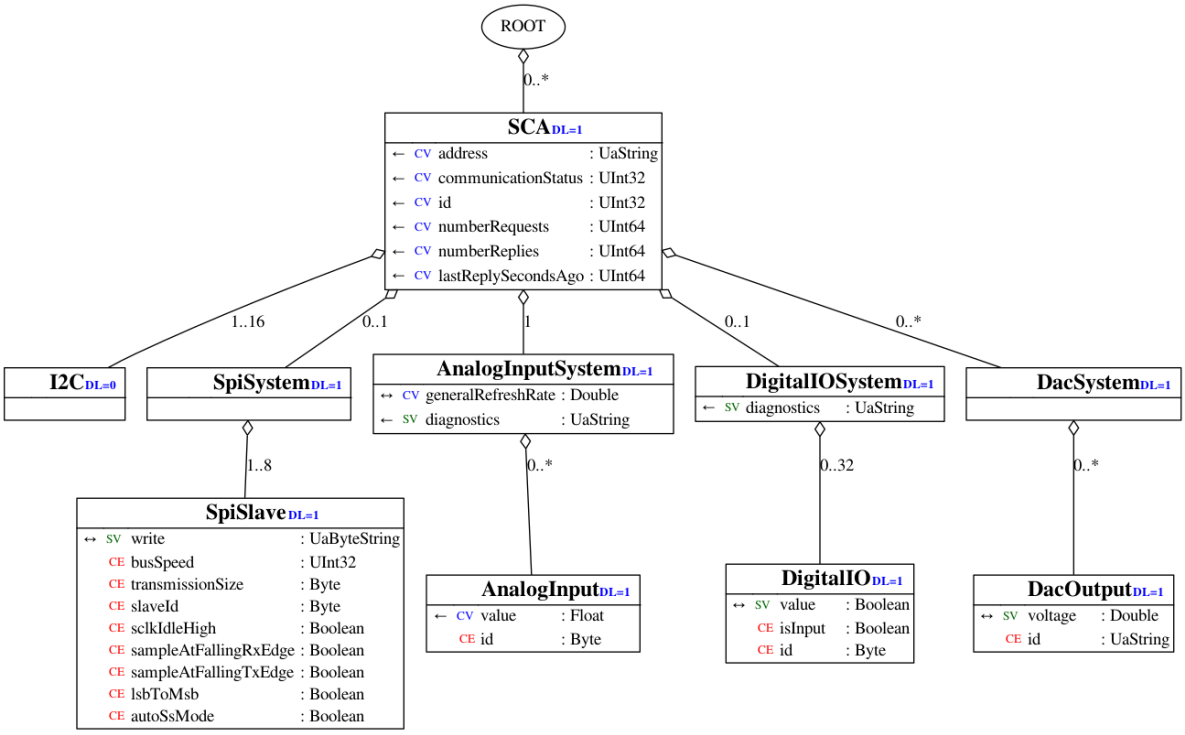


Figure 84: The quasar design diagram of the OPC-UA server for the SCA.

As of January 2018, the server supports the following functionality:

- Communication with any number of SCAs, through NetIO or any other HDLC backend. Each SCA is identified in its address-space by a name, and its unique 24-bit "SCA identifier" which is written by the chip manufacturer in the SCA silicon. The identifier is read using the SCA-SW library when a connection to given SCA is opened.
- Up to 32 ADC channels per SCA which are polled with the configured conversion frequency. Note that channel 31 has no external connection; it is connected to the on-chip temperature sensor that monitors the SCA temperature.

- Up to 32 General Purpose I/O pins per SCA. Each pin can be configured as an input or output through the server config file.
- Up to 4 DACs per SCA; the DACs take the desired voltage as a float (0..1V).
- Up to 8 SPI slaves per SCA. The SPI configuration (like speed, phase, mode . . .) can be configured in the server config file.
- Up to 16 independent configurable I2C master controllers

The JTAG controller is in-progress and not yet available as of January 2018.

D.7 References

1. SCA-SW gitlab repo
(https://gitlab.cern.ch/atlas-dcs-common-software/ScaSoftware)
2. Twiki for SCA-SW end-users,
(https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGbtScaOpcUa)
3. Twiki for SCA-SW developers
(https://twiki.cern.ch/twiki/bin/viewauth/Atlas/ScaOpcUaUserGuide)
4. OPC-UA SCA gitlab
(https://gitlab.cern.ch/atlas-dcs-common-software/ScaSoftware)

References

[1] CERN, *GBT & Versatile Link*,
URL: https://ep-ese.web.cern.ch/content/gbt-versatile-link.

[2] F Vasey et al., *The Versatile Link common project: feasibility report*,
Journal of Instrumentation **7**.01 (2012) C01075,
URL: http://stacks.iop.org/1748-0221/7/i=01/a=C01075.

[3] CERN GBT Project, *The GBTx Manual*, V0.14 (2016),
URL: https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxManual.pdf.

[4] *GBT Module for the FELIX Project*,
URL: https://twiki.cern.ch/twiki/pub/Atlas/GBT2LAN/FELIX_GBT_MANUAL.pdf.

[5] ATLAS Felix Group, *Specifications for the FELIX FULL mode link*, URL: https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/docs/FullMode.pdf.

[6] Xilinx, *Xilinx VC709 Development Kit*,
URL: http://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html.

[7] Supermicro, *Supermicro X10SRA-F Motherboard Model Specification*, 2016,
URL: http://www.supermicro.nl/products/motherboard/Xeon/C600/X10SRA-F.cfm.

[8] CERN TTC FMC project, URL: http://www.ohwr.org/projects/optical-cdr-fmc/wiki.

[9] TTC group, *CERN TTC homepage* (), URL: http://ttc.web.cern.ch/TTC.

[10] Analog Devices Inc., *ADN2814: Continuous Rate 10 Mb/s to 675 Mb/s Clock and Data Recovery IC with Integrated Limiting Amp*,
URL: http://www.analog.com/static/imported-files/data_sheets/ADN2814.pdf.

[11] Silicon Labs Inc., *Si5345/44/42 Rev D Data Sheet – 10-Channel, Any-Frequency, Any-Output Jitter Attenuator/ Clock Multiplier*,
URL: http://www.silabs.com/SupportDocuments/TechnicalDocs/Si5345-44-42-D-DataSheet.pdf.

[12] Silicon Labs Inc.,
Si5324 Data Sheet – Any-Frequency, Any-Output Precision Clock Multiplier / Jitter Attenuator,
URL: https://www.silabs.com/documents/public/data-sheets/Si5324.pdf.

[13] Xilinx, *Xilinx Vivado Design Suite*, 2016,
URL: https://www.xilinx.com/products/design-tools/vivado.html.

[14] Linear Technology, *LTC2991 Data Sheet – Octal I2C Voltage, Current, and Temperature Monitor*,
URL: http://cds.linear.com/docs/en/datasheet/2991ff.pdf.

[15] The Versatile Link Developers, *The Versatile Link Common Project*, 2008,
URL: https://espace.cern.ch/project-versatile-link/public/default.aspx.

[16] CERN GBT-FPGA project,
URL: https://espace.cern.ch/GBT-Project/GBT-FPGA/default.aspx.