Standalone Programs

Gareth Rogers

Contents

Overview

This document will outline the stand set up for the Fed9USoftware, FEDtesterSoftware, FEDtesterUtils and the ICUtils directories, how they are used and how they rely on the rest of the Tracker online software.

When starting a new shell session the .bash_profile should be called, which will setup the environment variables required for running with the online software. If the .bash_profile has been called automatically then the environment variable $DAQBASE should be defined. You can check this using the command

echo $DAQBASE

If is found to be undefined or it does not point to correct directory then the .bash_profile will need to be run manually. This is done by changing to the $HOME directory and sourcing the .bash_profile using the following command.

. .bash_profile

The first dot is very important. If there is no initial dot present then the .bash_profile will be executed instead of sourced and none of the environment variables set up will persist in the shell. An example is shown in figure 1.

```
[xdaq@te2papeete xdaq]$ . .bash_profile

Setting up tracker online software environment.
No database can be used
$DAQBASE=/home/xdaq/local/2004

Tracker software recompilation will use SBS bus adapter.
[xdaq@te2papeete xdaq]$
```

Figure 1

This sets up four environment variables ICUTILS, FED_ROOT, FEDTERSTER_ROOT and FTUTILS_ROOT. These can be used to change to the directories containing the latest version of the ICUtils, Fed9USoftware, FEDtesterSoftware and FEDtesterUtils code. Each the code with each directory can be compiled by calling gmake in the top level directory, which the environment variables put you in. ICUtils and FEDtesterSoftware are independent of any of the other directories and should be compiled first. Fed9USoftware is dependent upon the existence of the ICUtils libraries and should be compiled next. FEDtesterUtils is dependent upon FEDtesterSoftware and Fed9USoftware (and hence ICUtils) libraries and should be compiled last.

Updating From CVS

The Fed9USoftware and ICUtils software is all contained in a CVS repository located on heppc41.hep.ph.ic.ac.uk, which is located in Imperial College. In order to access this repository a user name and password are required, Jonathan Fulcher [1] should be contacted if you do not already have your own.

In order to access the CVS repository a tunnel may need to be setup on the host machine to the CVS repository. For this two shells will be required. In the first shell the tunnel should be created, which is done using the command

ssh –L 2401:localhost:2401 ‘username’@heppc41.hep.ph.ic.ac.uk

where ‘username’ is you CVS user name. You must then enter you password at the prompt. This shell must be left open and cannot be used for any else. In the second shell you should change to the $FED_ROOT or $ICUTILS directory and type ‘cvs login’ entering your user name and password when prompted. A ‘cvs update’ can be performed to check for the latest version of the code. If newer files exist they will be downloaded.

More information about using CVS can be found in the file /home/xdaq/help/cvs.help, which written by Ian Tomalin, RAL.

ICUtils

This is typically found in /home/xdaq/local/ICUtils and the xdaq_profile will set up the environment variable $ICUTILSDIR which hold the absolute directory path. The command 'cd $ICUTILSDIR' can be used to change to this directory from any other.

This is a utility class can contains only examples of how to include it in your own code. The Fed9USoftware is however heavily dependent up on this library. A User is likely to only need to compile or update this directory from the Fed9USoftware CVS repository. The ICUtils directory can be updated from CVS by following the direction given under the section heading Updating from CVS. ICUtils is compiled by changing to the directory $ICUTILSDIR and entering 'gmake'. This will recompile the libraries if necessary.

Fed9USoftware

The Directory Structure

There may two directories containing the Fed9USoftware. One will contain the latest official release of the FED software that is compatible with online software and a second development version that is the latest code from the CVS repository. The release software is referred to by symbolic links in the $XDAQ_ROOT directory, the development version is referred to by the environment variable $FED_ROOT. The release software should be in a directory called Fed9U/Fed9USoftware/ and will only need to be compiled once or if there is a new release. In which case the install instructions accompanied with the release should be followed. The following instructions refer to the install and maintenance of the development version.

All the Fed9USoftware is located in the directory Fed9USoftware/, which is typically found in /home/xdaq/"online software dir"/Fed9UDevelopment/Fed9USoftware/, where "online software dir" refers to the directory in which all the online software is kept. The environment variable $FED_ROOT stores the absolute file path for Fed9USoftware/, it is however not part of the standard Fed9USoftware setup. It is instead required by the FED Tester, so if the FED Tester is not installed on the system is may not be present. It can be added to the .bash_profile if desired.

```
[xdaq@heplnw13 help]$ cd $FED_ROOT
[xdaq@heplnw13 Fed9USoftware]$ ls
CVS                     Fed9UDocumentation  Fed9USupervisor  licence.txt
FED9U_BUILD_VERSION_2   Fed9UGuiConfigure   Fed9UUtils       Makefile
Fed9UDescriptionFiles   Fed9UNonStd         Fed9UVMEBase     Makevars
Fed9UDevice             Fed9UStandAlone     Fed9UVMEDevice   ReleaseMakefile
[xdaq@heplnw13 Fed9USoftware]$
```

Figure 2

The Fed9USoftware directory contains several subdirectories (shown in figure 2), which all have a common structure. Each subdirectory contains three more directories on called src/ for the source code, include/ for the header files and obj/ for the object files created by the compiler. Any libraries created are stored in the top level of the subdirectory along with the make file. All the Fed9USoftware stand alone programs are contained in $FED_ROOT/Fed9UStandAlone/ and the source code for them in src/. The directories Fed9UUtils/ and Fed9UDevice/ each contain a shared object library, libFed9UUtils.so and libFed9ULib.so respectively, in their top level and a header file called Fed9UUtils.hh and Fed9ULib.hh respectively in their include/ directories. Both of these paths must be present in the environment variable $LD_LIBRARY_PATH in order for the Fed9UStandAlone executives to run.

Compiling

The Fed9USoftware can be updated from the Fed9USoftware CVS repository. Instruction for accessing and using the repository can be found under the section heading Updating from CVS.

Compiling the Fed9USoftware is a straight forward affair (providing there are no errors). The top level $FED_ROOT/ directory contains a file called Makevars, this contains all include and library paths required for the Fed9USoftware to compile and a file called Makefile (as can be seen in figure 2), this lists all the dependencies for the different directories and ensure that they are compiled in the correct order. In $FED_ROOT/ calling 'gmake' will recompile all the Fed9USoftware. A small section of typical compiler output is given in figure 3. Each of the subdirectories contains a Makefile in its top level directory and the Fed9USoftware Makefile simply calls these. These Makefiles contain the target and any custom includes and libraries for that directory. As each subdirectory contains its own Makefile it can be compiled on its own by entering the appropriate subdirectory and calling 'gmake'. This method has no knowledge of any directory dependencies, so the User must ensure all directories it is dependent upon are already compiled.

```
**** Compiling obj/Fed9UXMLDescription.o from src/Fed9UXMLDescription.cc
g++ -o obj/Fed9UXMLDescription.o src/Fed9UXMLDescription.cc -c -I/home/xdaq/NEW/2004//root/include/ -Iinclude  -I/home/xdaq/NEW/2004/
/TriDAS/daq/itools/packages/xdaq-shell/include/ -I/home/xdaq/NEW/2004//TriDAS/daq/itools/packages/i2o/include/ -I/home/xdaq/NEW/2004/
/TriDAS/daq/itools/packages/i2o/include/i2o/shared/ -I/home/xdaq/NEW/2004//TriDAS/daq/toolbox/include/ -I/home/xdaq/NEW/2004//TriDAS/
daq/toolbox/include/linux/ -I/home/xdaq/NEW/2004//TriDAS/Auxiliary/xerceslinuxx86/include -I/home/xdaq/NEW/2004//TriDAS/daq/extern/xe
rces/linuxx86//src/ -I/home/xdaq/NEW/2004//TriDAS/Auxiliary/i2o/include/i2o/ -I/home/xdaq/NEW/2004//TriDAS/daq/xdaq/include/ -I/home/
xdaq/NEW/2004//TriDAS/daq/xoap/include -I/home/xdaq/NEW/2004//TriDAS/daq/evbim/include/ -I/home/xdaq/NEW/2004//TriDAS/daq/toolbox/inc
lude/solaris/ -I/project/xdaq/TrackerColumn/include/ -I/home/xdaq/NEW/2004//FecSoftware/FecSupervisor/include/ -I/home/xdaq/NEW/2004/
/hal//generic/include/ -I/home/xdaq/NEW/2004//hal//generic/include/linux/ -I/home/xdaq/NEW/2004//hal//busAdapter/include/ -I/home/xda
q/NEW/2004//SBS/1003/v2_0/include/ -I/home/xdaq/NEW/2004//SBS/1003/v2_0/1003/v2_0/include/ -I/home/xdaq/NEW/2004//root/include/ -I../
Fed9UNonStd/include/  -I../Fed9UUtils/include/ -I/home/xdaq/NEW/2004/Fed9USoftware/ICUtils//include/ -I../Fed9UVMEBase/include/ -I../
Fed9UVMEDevice/include/ -I../Fed9UDevice/include/  -I../Fed9ULib/include/ -Dlinux      -DLITTLE_ENDIAN__     -DCPU=x86 -DSOAP__      -D
toolbox           -DBT1003   -DVXILINUX   -DXERCES=2 -DTBHARDWARE -ansi -pedantic -Wall -Wno-long-long -pipe -W -O2 -g
In file included from /home/xdaq/NEW/2004/TriDAS/daq/extern/xerces/linuxx86/src/xercesc/dom/DOMWriter.hpp:318,
                 from /home/xdaq/NEW/2004/TriDAS/daq/extern/xerces/linuxx86/src/xercesc/dom/DOM.hpp:111,
                 from include/Fed9UXMLDescription.hh:17,
                 from src/Fed9UXMLDescription.cc:17:
/home/xdaq/NEW/2004/TriDAS/daq/extern/xerces/linuxx86/src/xercesc/framework/XMLFormatter.hpp: In method
`xercesc_2_3::XMLFormatTarget::XMLFormatTarget (const xercesc_2_3::XMLFormatTarget &)':
/home/xdaq/NEW/2004/TriDAS/daq/extern/xerces/linuxx86/src/xercesc/framework/XMLFormatter.hpp:538: warning: base
class `class xercesc_2_3::XMemory' should be explicitly initialized in the copy constructor
```

Figure 3

Linking with Non Standard Libraries

Each of the Fed9USoftware subdirectory Makefiles has two lines in it. One specifies custom includes and the other specifies custom libraries. The custom includes requires just the directory path to the folder containing the headers, which must be preceded by a -I. The custom libraries require both the path to the directory containing the library, this time preceded by a –L and the name of the library to use. The library file name can be in one of formats lib*.so or lib*.a, where * can be anything, the format for defining a library is –l*, the lib and .so/.a are not required the make file automatically looks for files in that format. An example of some custom includes is shown in figure 4.

```
LINKTOHAL=1

include ../Makevars

CUSTOM_INCLUDES=-I/project/xdaq/Daq_Cms_Like/DeviceDriver/Tsc_Driver
 -I$(FEDTESTER_ROOT)/FEDtester/include

CUSTOM_LIBS=-L$(QTDIR)lib/ -lqt -L/usr/X11R6/lib -lX11 -L/home/xdaq/
root/lib/ -lCore -lCint -lGpad -lGraf -lMatrix -lHist -lpthread -ldl
 -L$(FEDTESTER_ROOT)FEDtester/ -lFEDtester -lFed9ULib -lFed9UUtils -
lGenericHAL -lICUtils
```

Figure 4

FEDtester

The FED Tester can drive the 96 optical inputs of the CMS Tracker Front End Driver (FED) with different analogue patterns and timing settings thus allowing the full functionality of the FED to be tested. FED Tester project home page can be found here at [2]. The full documentation and details of the FED Tester can be found there and any questions should be directed to Greg Iles at Imperial College London, whose contact details can be found on the FED Tester home page [2].

The Directory Structure

The environment variable $FEDTESTER_ROOT contains the absolute file path for the FEDtesterSoftware home directory. The FEDtesterSoftware has a similar directory structure to the Fed9USoftware, which is illustrated in figure 5. All of the FEDtesterSoftware code and libraries can be found in $FEDTESTER_ROOT/FEDtester, the stand alone programs can be found in $FEDTESTER_ROOT/FEDtesterStandAlone/.

```
[xdaq@te2papeete FEDtesterSoftware]$ cd $FEDTESTER_ROOT
[xdaq@te2papeete FEDtesterSoftware]$ ls
Docs        FEDtesterConfigFiles  FEDtesterStandAlone  INSTALL      Makefile       README   VERSION
FEDtester   FEDtesterProfile     FEDtesterUtils       licence.txt  MakeRules.gmk  _System
[xdaq@te2papeete FEDtesterSoftware]$
```

Figure 5

The FEDtesterUtils is a collection of libraries and stand alone programs that can be used to configure both the FEDtester and the FED. The environment variable $FTUTILS_ROOT contains the absolute path of the FEDtesterUtils home directory. The directory structure of the FEDtesterUtils directory is shown below.

```
[xdaq@te2papeete FEDtesterUtils]$ ls
Fed9UReadout       FEDtesterUtilsProfile  FTeFrameMaker    licence.txt    README
Fed9UTimingCal     FTeAohCal              FTeFrameMakerEx  Makefile       _System
Fed9UTimingCalEx   FTeAohCalEx            INSTALL          MakeRules.gmk  VERSION
[xdaq@te2papeete FEDtesterUtils]$
```

Figure 6

Updating and Compiling

The latest version of the FEDtesterSoftware and the FEDtesterUtils can be found on the FEDtester site [2], which has instruction for installing the software. The FEDtesterSoftware or FEDtesterUtils tar file should be downloaded and extracted to a suitable location. Before the software can be compiled the .bash_profile will have to be modified so that it setups up the relevant environment variables for use with the FEDtester. This is explained in the $FEDTESTER_ROOT/INSTALL and $FTUTILS_ROOT/INSTALL files. Once the environment has been setup the software can be compiled by using the command ‘gmake’ in the top level directory of either $FEDTESTER_ROOT or $FTUTILS_ROOT. The subdirectories can be individually compiled by typing ‘gmake’ in the top level of the subdirectory. In the case of the FEDtesterSoftware this method has no knowledge of dependencies and should only be used if it is known the other directories compilations are up to date. The FEDtesterUtils subdirectories are not dependent on each other.

Creating Fed9UDescription Files

A Fed9UDescription file contains all the settings required to configure a FED. It is required when constructing all of the standalone that interact with the FED. As a result all the directories containing a standalone capable of talking to the FED have a program that can create Fed9UDescription objects, update its settings and save them to file. In the Fed9USoftware release there is a program called Description.exe in $FED_ROOT/Fed9UStandAlone which will generate Description files in the .fed file format. Each program with in $FTUTILS_ROOT has an executable called FedDescription_’project_name’.exe, where ‘project_name’ is the name of the name of the subdirectory it is located in. These all save there files to a subfolder called FedDescriptionFiles/ with a predetermined file name. The only exception is in $FTUTILS_ROOT/Fed9UReadout/. This contains a description file that has a User defined file name and the ability to save in either .fed or .xml format.

Each of the source files for the description generating executables is based upon the same designed. In order to modify the description files that are generated by the executables the source code must be modified. The source code can be found in ‘executable directory’/src/ and will have the extension .cxx. Each file has the same layout. A Fed9UDescription object is created using the default constructor, which loads the default settings. By using the Fed9UDescription class’ set methods the settings can be overridden. The object is then saved to file. In order to modify the saved settings the User must open

the appropriate source file and update the relevant settings (the Fed9UDescription documentation can be consulted if the User is unsure of the appropriate method to use), save the changes and then recompile the executable using ‘gmake’ in the top level directory of the project. Once this has been done the User only need run the appropriate description file generating executable and a new file will be generated. The generated file can be copied to a file with a different name if a copy is needed.

There is a program with a GUI interface that can be used to generate description files. It is located in $FED_ROOT/Fed9UGuiConfigure. It requires QT version 3 to be installed on the system and a slightly different method is required to recompile. In the Fed9UGuiConfigure directory first ‘qmake’ should be called, which is then followed by a ‘gmake’. The executable Fed9UGui will be generated, which when run will generate a description file that must be saved in the .xml format. Figure 7 shows an example of the Fed9UGui.

```
[xdaq@te2papeete Fed9UGuiConfigure]$ Fed9UGui &
[3] 16292
[xdaq@te2papeete Fed9UGuiConfigure]$ []
```

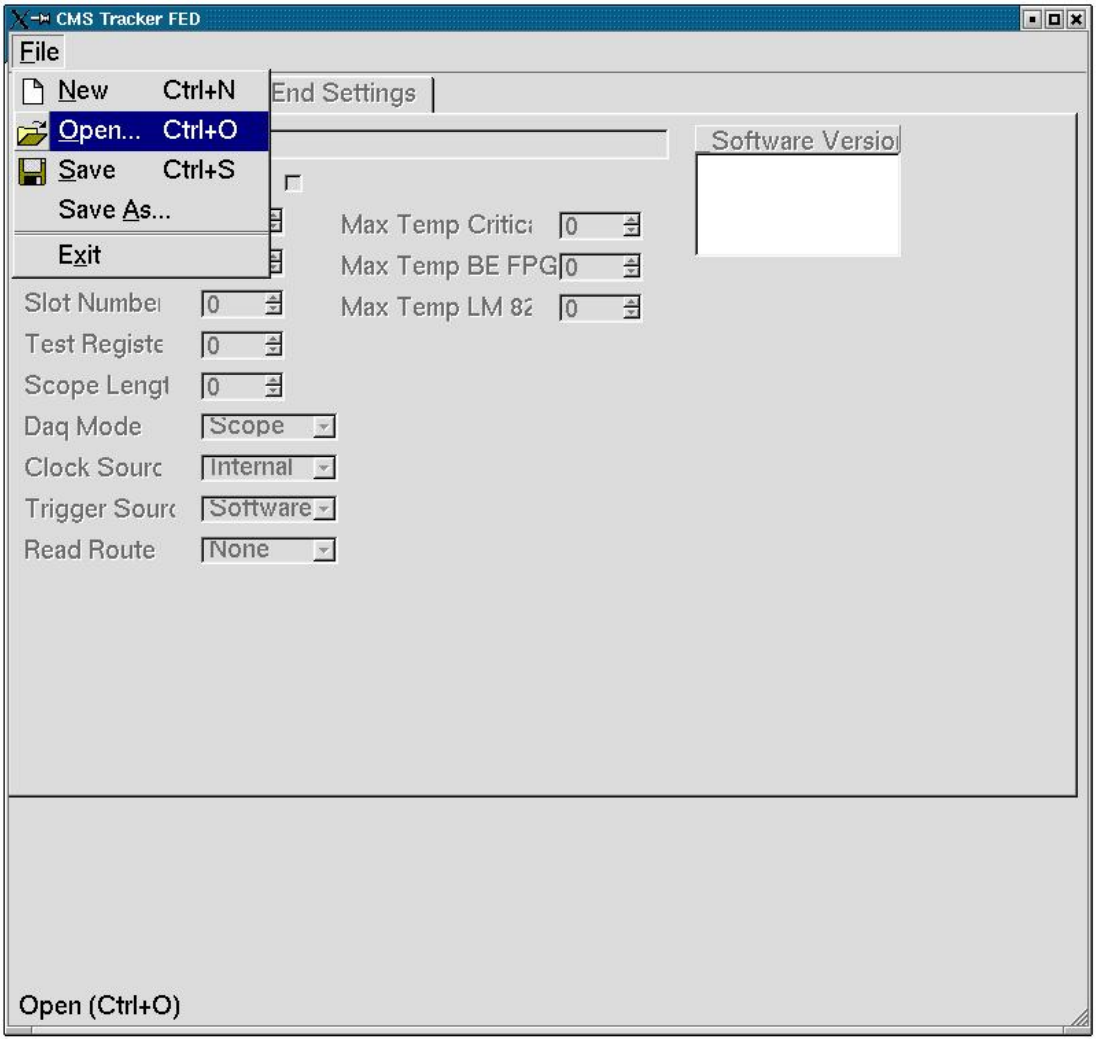

Figure 7

ReadoutThreaded

ReadoutThreaded is a standalone executable that uses a FEDtesterEnsemble class to drive the FEDtester and Fed9UVmeDevice to drive the FED. It has two threads of execution. One is used to continually poll the FED to check for events. The other is a command line User Interface to control the program. ReadoutThreaded configures the FED from a description file that is loaded as a command line argument in either .fed or .xml format. It is located in the directory $FTUTILS_ROOT/Fed9UReadout, along with all the files that are needed to configure and run ReadoutThreaded.exe.

Configuring the FEDtester

ReadoutThreaded uses the class FEDtesterEnsemble to configure and drive the FEDtester. A FEDtester ensemble is a group of up to four FED Testers, which is the number required to drive all 96 channels on a FED. A FEDtesterEnsemble object provides a single access point to the four FEDtesters of the ensemble. The class FEDtesterEnsemble can be used to drive up to four FED Testers and in the case where less than four FED Testers are present, it is recommended that FEDtesterEnsemble constructed with four FED testers and the FEDtesterMap.txt (explained in more detail below) can be used to disable the Testers that are not present. The FEDtesterEnsemble class can requires up to ten different configuration files to run, the FEDtesterMap.txt, the FEDtesterEnsembleConfigFile.txt, four APV frame data files and four APV frame pointer files. A brief description of each file is given below, but it is not detailed enough to create the files from. For a more detailed description consult the FED Tester documentation [2].

The FEDtesterMap.txt tells the FEDtesterEnsemble how many Testers are present, their base address, which bus adaptor should be used, and which FEDtester is to be considered the master. There can only be one FEDtester master and it is used to distribute the TCS to the other FEDtesters. The most flexibility in the setup is provided when settings for four FED Testers are used in the configuration file and the unused ones are set to OFF.

FEDtesterEnsembleConfigFile.txt contains configuration details for each of the Testers present in the ensemble. It contains the channel mapping for FEDtester channels to the MUX's, which provide the channels with the simulated multiplexed APV frames and FEDtester channels to FED channels mapping. It also contains the configuration settings for each FEDtester in the ensemble. The configuration settings are based on the FEDtesterConfiguration.txt used with the FEDtesterApplication class, which can be used to configure a single FED. There are four settings that the author has found useful in this file. The type of triggers that are sent, repetitive or random (there could be other options, but this is unknown to the author), the number of clock cycles between repetitive triggers (400 is equivalent to 100kHz) and the files paths for the APV frame data and pointer files, described below. Configuration file can contain up to four file paths for the pointer file and four more for the data file, one for each FEDtester in the ensemble. Even if there are four FEDtesters present in the ensemble there is no need for each to be driven from a different pointer or frame file, but the same path must then be specified for each.

The APV frame data file contains all the information needed to create a simulated APV frame. The first 70 lines contain the tick marks that are sent at the beginning of each APV frame. Then starting at line 71 there are 233 data frames, each one 140 lines long. Each new frame starts at line 140n + 70, where n <= 233. The APV data frame strip ordering is the same as that sent from an APV. This means that any pattern written to the frame must first be disordered from physical strip order to APV.

The APV frame pointer file contains six columns and 1024 rows. Each element in the file points to the line number in the APV frame data file and uses the equation

$$\text{Line N}^{o}. = 140n + 70, \text{ where } n <= 233.$$

Each column represents an APV and pairs of columns are used to generate the multiplexed APV frames. Columns 0 and 1 represent mux0, 3 and 4 mux1, and 5 and 6 mux2. With each new trigger that the FEDtester sends the line number is incremented and a new row is used to select the APV data frames, until the end of the file is reached, at which point the FEDtester loops back to the beginning. To send the same frame for each APV this file should just be filled with a single number representing the line number of that desired frame in the pointer file. If it is filled with random numbers then random frame will be sent from the APV frame data file.

The directory $FTUTILS_ROOT/Fed9UReadout/ contains an example FEDtesterConfigurationFile.txt and example Fed9UDescription file for use with ReadoutThreaded. These are contained in the subdirectories FTeConfigFiles and Fed9UDescriptionFiles respectively. These however will contain generic settings and may not be configured correctly for the User's setup. The analogue optical hybrids (AOH) gain settings and the FED timing calibration on the individual channels can be configured automatically using libraries provided in FEDtesterUtils. The AOH can be configured using the program FTeAohCal.exe, which is located in the folder $FTUTILS_ROOT/FTeAohCalEx. The FED

timing calibration can be done using the program found in $FTUTILS_ROOT/Fed9UTimingCalEx. More information about how to use both programs can be found in $FTUTILS_ROOT/README.

Configuring the FED

The FED is initialised from the settings taken from the Fed9UDescription file that is given to ReadoutThreaded.exe as a command line argument. It can handle files in both the .fed and .xml format. These can be generated by the program FedDescription_Fed9UReadout.exe, which is located in the same directory as ReadoutThreaded.exe ($FTUTILS_ROOT/Fed9UReadout). For an explanation on how to manipulate the files created by FedDescription_Fed9UReadout.exe see the section Creating Fed9UDescription Files. When ReadoutThreaded.exe is running the FED registers can be manipulated through the class UserMenu, which can be accessed through option 'u' in the ReadoutThreaded.exe User Interface.

Using the UserMenu

The UserMenu provides a constructor that takes a constant reference to a Fed9UDescription object and uses this object to construct a Fed9UVmeDevice object. The Fed9UDescription object will not be changed by UserMenu. This is because Fed9UVmeDevice creates a local copy of the description object it is constructed with. Each setting that is changed using the UserMenu is updated in the Fed9UVmeDevice local copy of the Fed9UDescription. The UserMenu provides a method to get a copy of this updated description object from Fed9UVmeDevice.

The interface presented to the User upon construction of a UserMenu object is a simple text based one. The interface provides a means to access each of the individual FED registers through the interface provided by Fed9UVmeDevice. The User is initially presented with eight different options, which are selected by entering the appropriate character. As shown in figure 8.

```
[xdaq@heplnw189 Fed9UStandAlone]$ Fed9URalStandAlone.exe ../Fed9UDescriptionFiles/zero_supp_crate0_slot_18_100high_50low_0and3FE.xml
Start of main()
****************** mode string node value = ZERO_SUPPRESSION
WARNING: if you set ttcrxBrcstStrTwoFineDelay, then this will overwrite the ttcrxL1AcceptFineDelay if the ttcrxClock40DeskwedTwo is set to DISABLE.
Program started...
VMEDummyBusAdapter : constructor has been called :
                     The "verbose" flag is set to 0
                     The "memoryMode" flag is set to 1
VMEDummyBusAdapter : opening Device number 0 with baseAddress : 00120000
                     memory-mapping-mode : 1
                     mapped the address space to memory address 842e5c0
                     reserved 0x10002 (dec: 65538 ) bytes for memory mapped operation

###########################################################################
#-------------------------------------------------------------------------#
#----------------------- FED Operation Menu ------------------------------#
#-------------------------------------------------------------------------#
###########################################################################
# Please enter a choice:                                                  #
#                                                                         #
#  1 Front End Fpga Commands                                              #
#  2 Back End Fpga Commands                                               #
#  3 Vme Fpga Commands                                                    #
#  4 High Level Commands                                                  #
#  5 TTCrx Commands                                                       #
#  6 Voltage Monitor Commands                                             #
#  7 System ACE Commands                                                  #
#  8 Load A New Description                                               #
#                                                                         #
#  x exit                                                                 #
#                                                                         #
###########################################################################
```

Figure 8

The User will then be prompted with a submenu with options for accessing the individual FED registers. Almost all the submenus have the option to exit, which will take the User back to the top level menu and all menu's can handle invalid options. Below shows an example of the UserMenu being used to select the clock source. The clock source is a VME command and is located in the VME FPGA menu (option 3) and the clock source option is d. The clock source is a read/write option so the User must select which they wish to do. In the case of read the clock source is returned and the case of write the User must enter the appropriate option depending on which clock they wish to use. This is demonstrated in figure 9.

```
################################################################################
#------------------------------------------------------------------------------#
#----------------------- FED Operation Menu -----------------------------------#
#------------------------------------------------------------------------------#
################################################################################
# Please enter a choice:                                                       #
#                                                                              #
#  1 Front End Fpga Commands                                                   #
#  2 Back End Fpga Commands                                                    #
#  3 Vme Fpga Commands                                                         #
#  4 High Level Commands                                                       #
#  5 TTCrx Commands                                                            #
#  6 Voltage Monitor Commands                                                  #
#  7 System ACE Commands                                                       #
#  8 Load A New Description                                                    #
#                                                                              #
#  x exit                                                                      #
#                                                                              #
################################################################################
3

################################################################################
#------------------------------------------------------------------------------#
#----------------------- VME Fpga Operation Menu ------------------------------#
#------------------------------------------------------------------------------#
################################################################################
# Please enter a choice:                                                       #
#  Read Only                                                                   #
#  a Read Event Counter        b VME Firmware ID          c Has Event         #
#                                                                              #
#  Read or Write                                                               #
#  d Select Clock Source       e Serial EPROM     f FED Serial Number         #
#                                                                              #
#  Write Only                                                                  #
#  h Purge Event                          i FED Reset                          #
#                                                                              #
#  x exit                                                                      #
#                                                                              #
################################################################################
d
Type in 0 to write or 1 to read:
0
Please enter the clock number
1 is the on board clock.
2 is the TTC clock.
4 is the Back plane clock.
1
You have written the clock value of 1

################################################################################
#------------------------------------------------------------------------------#
#----------------------- FED Operation Menu -----------------------------------#
#------------------------------------------------------------------------------#
################################################################################
# Please enter a choice:                                                       #
#                                                                              #
#  1 Front End Fpga Commands                                                   #
#  2 Back End Fpga Commands                                                    #
#  3 Vme Fpga Commands                                                         #
#  4 High Level Commands                                                       #
#  5 TTCrx Commands                                                            #
#  6 Voltage Monitor Commands                                                  #
#  7 System ACE Commands                                                       #
#  8 Load A New Description                                                    #
#                                                                              #
#  x exit                                                                      #
#                                                                              #
################################################################################
```

Figure 9

Using ReadoutThreaded

Upon start up ReadoutThreaded will reset and configure the FED Tester Ensemble from a FEDtesterEnsembleMap.txt and a FEDtesterEnsembleConfigFile.txt, which are defined at compile time. Next the FED is reset and initialised from the description file given as a command line argument. At this point the program is split into two threads of execution.

The original thread setups up the readout loop. This continually polls the FED to check for events. When an event is found it is readout and checked for errors using Fed9UEvent. The Fed9UEvent object contains the event data from the FED and ReadoutThreaded can easily be modified to use the

Fed9UEvent object to save or display the event data. At present the ReadoutThreaded saves the event data in a format which can be read by the DisplayEvent standalone programs and using the plot graph option from the User Interface and as a raw event dump of the bytes, a copy of the description file used during that run is also saved.

The second thread of execution is the User Interface and provides the User with several options for the control and configuration of the FED and FEDtester. ReadoutThreaded.exe can send either software or hardware triggers, depending on the FED setup. The hardware triggers are sent by the FED Tester via the backplane and are used to drive the FED with frames from the Tester. When each event is readout and checked by Fed9UEvent there is the option of enabling the debug print out, which provides more details about the checks that are being done. This is quit a verbose process though and so unless an error is expected it is not recommended that it be enabled. The control of the readout is done through the options readout and polling. Display counters prints to the screen the current values for the total level 1 accept counter, the total and current number of unread frames in the QDR and the number of unread bytes in the QDR. Once the frames have been read out and saved to file in a folder called eventData/ in the main Fed9UReadout directory. The option 'G' can be used to plot a graph of the saved events. The User must just give the event number they wish to plot. This is given in the standard screen print out when an event is saved. After the graph is finished with File → Quit root must be used to cleanly exit root and return control back to ReadoutThreaded. The canvas will not be closed completely until ReadoutThreaded is exited. ReadoutThreaded can be used to perform a timing scan on either the FED or the FED Tester. In the case of the FED Tester it is used to set the global phase of the mux's. The mux is used to combine and output two APVs of data. The FED timing scan increments the fine skew from 0 to 25 in order to increase amount of skewing that is applied to each FED channel. The channels are increased one at time. This is quite useful if it is found that out of synch error bits are found in the event header file.

Display Event

The class Fed9UEvent is used to check the FED events for data format errors and can be used to retrieve the information stored in an event. It is also used to save the event data to a file. There exist several standalone programs to produce graphs of this data using ROOT. A brief description of each of these is given below.

DisplayEvent.exe is part of the Fed9USoftware release and can be found in $FED_ROOT/Fed9UStandAlone. It can plot a single channel and takes the path and name of the file you wish to plot as a command line argument.

DisplayEventAll.exe is located in $FTUTILS_ROOT/Fed9UReadout. It will plot all the channels stored in the event file and also takes the path and name of the file you wish to plot as a command line argument. It is part of the FEDtesterUtils software package.

DisplayEventChoose.exe also takes the path and file name as a command line argument and is stored in $FTUTILS/Fed9UReadout/. This however gives you the option of whether you want to display individual channels or all the channels on an FPGA. In the case of the individual channels you can select how many you wish to display. It also displays some other potentially useful information, such as the low and high cluster thresholds. This information is obtained from a Fed9UDescription file that should be saved at the same time as the event data was taken. At present it can only handle files that are in the .fed format. DisplayEventChoose.exe uses the class DisplayEventClass to produce the graphs. It is this class that ReadoutThreaded uses to plot graphs of the data it records. It is also part of the FEDtesterUtils software package.

SoakTest

The soak test is the provided as a method with the class Fed9UDevice in the Fed9USoftware release. There is a standalone program called soakTest.exe, which is located in $FED_ROOT/Fed9UStandAlone that can be used to perform a User defined number of tests. A single soak test sets all the registers on the FED (including if desired each writable byte of the EPROM memory) and read them back to check they have written correctly. SoakTest.exe can be used to test the whole range of registers on the FED by incrementing each register value from is minimum value to its maximum.

soakTest.exe requires only two arguments the number of times to perform the test and whether the EPROM memory is to be tested or not. Testing the EPROM at least doubles the time taken to perform a test, which can be quite significant if the whole range of the FED registers is tested. A single soak test will set all the FED registers and check them once. For the whole range of all the registers to be tested the soak test would have to be performed 4096 times, however most registers are below 8 bits and there are only one or two that are greater.

The soak test works by loading a Fed9UDescription object to the FED and then comparing those values to those read back by the FED. The Fed9UDescription object is generated using the Fed9UUtils class Fed9UCreateDescription. A Fed9UCreateDescription object keeps track of how many Fed9UDescription objects it has created and sets the FED registers value depending on that. This enables it to create a description with a set of FED register values that have been incremented compared to the previous on. The constructor takes a single argument that is used to seed the starting number, which is defaulted to 0. The method to generate a Fed9UDescription object requires the crate number, slot number and location of the HAL address table. If there are more than one crate being controlled by the pc then the crate number will be relevant, otherwise it is just 0. The slot number is slot that the FED occupies in a crate and the HAL address table is required by Fed9UVmeBase for communication with the FED, it provides the address of the registers on the FED. These are all given in the Fed9UDescription object used to create the Fed9UDevice object and in the soak test method will be retrieved from there.

The soak test adds the details of its run to the Fed9U.log file. The start and finish time of the test is logged, the number of tests performed and also any errors that occur. The test number is logged along with the error that occurred. The soak test can catch all errors that occur generating a useful message where possible. The present of a catch(…) statement prevents the series of tests being aborted prematurely due to an error in a single soak test.

The soak test will check each register on the FED that it can be written to and read from. By performing multiple tests it can be used to test the whole range of values for the FED registers. It logs all test information to the Fed9U.log file, creating a record of the run and any errors that were encountered. It is uses Fed9UCreateDescription to generate Fed9UDescription objects, which at present increment the FED values sequentially but could easily be altered to generate different test patterns.

References

[1] mailto:jr.fulcher@imperial.ac.uk

[2] http://www.hep.ph.ic.ac.uk/~ilesgm/projects/fed_tester/fed_tester.html

[3] Authors email ggr1@aber.ac.uk