Testing and Programming the Integrator/ Digitizer Card for the Beam Loss Monitoring System

Prakrit Nepali Shrestha

Physics Department, The College of Wooster

Supervisor: Craig Drennan

Proton Source, Accelerator Division

SIST Program

Fermi National Accelerator Laboratory

Batavia Illinois 60510

(Dated: August 7, 2013)

Abstract

The beam loss monitoring system is one of the most widely used systems in particle accelerators. Beam losses are used for beam tuning and accelerator diagnostics. The BLM systems implemented at Fermilab are comprised of two fundamental component, the BLM detector and the VME crate. In this project a program to configure and test the Integrator/ Digitizer card in the VME crate was developed. In this project, we have successfully developed user interface to program and configure the FPGA via the EEPROM for the Digitizer/ Integrator card. Furthermore, routines were developed to test different components of the card.

Contents

I. INTRODUCTION

Fermilab's mission of understanding the nature of matter and energy is conducted through high energy and high intensity particle physics research. Particle accelerators accelerate particles to high energies. At Fermilab, Hydrogen ions are initially accelerated through a linear accelerator up to 400 MeV. Next the beam enters the Booster ring that uses magnets to bend the beam into a circular path. One of the most important beam diagnostic systems needed in such accelerators is the Beam Loss Monitor (BLM). In a perfect system, installing a BLM system would be illogical and unnecessary, however since we do not possess such a machine, it is necessary to install this system. As the particles are accelerated in the Booster, controls must be adjusted to ensure that the beam stays in the accelerator and does not scrape against the inside. This beam is really powerful and can generate losses that could damage the magnets and other components by inducing stray radiation. The beam loss monitoring system is one of the most widely distributed systems in most particle accelerators. Beam losses are used for beam tuning and accelerator diagnostics.

The BLM systems implemented at Fermilab are comprised of two fundamental component: the BLM detector and Integrator and Readout Electronics.
The BLM detector is an argon filled glass cylindrical ionization chamber with nickel electrodes. The gas is 1 atm pure argon with an active volume of about 110 cm^3. This design was chosen to be extremely radiation hardened [1]. The inner electrode, anode (high voltage) is placed in the center and the outer electrode, cathode (signal) surrounds the inner electrode as a cylinder. The argon gas fills the space between these electrodes. Argon gas, being inert has a very low electron attachment to form negative ions, and hence the electron drift velocity is about 0.5 cm/µs which gives a large prompt signal. The detector calibration is about 70 nC/Rad and is extremely stable [2]. The radiation induced current at the cathode is brought to the integrator electronics crate via RG-58 coaxial cables. The electronics are powered and integrated in VME crates. The full set of electronics constitutes a number of boards handling various functionalities. The crate consists of a the following cards:

· Control Card (CC): setup and control the overall system,

· Timing Card (TC): generate and distribute timing signals to other cards,

· Abort Card (AC): contain the abort decision logic,

· High Voltage Card (HV): provide high voltage bias for the loss monitor tubes,

· Integrator/ Digitizer Card (DC): integrate and digitize the current from ionization chambers.

In addition to these cards, there is a VME slot 1 crate processor card (CP) that communicates data to the main control room via the Accelerator Controls Network (ACNET). This project is limited to studying and testing the Integrator/ Digitizer Card. This paper explains in brief about the mechanism of the Digitizer Card, programming the Field Programmable Gateway Arrays (FPGAs) implemented in the module and testing the module itself.

II. DIGITIZER CARD

The digitizer card is the primary data collector in the system. It is a dual FPGA based module that performs integration and digitization of the current from up to four ionization chambers. A TI/ Burr-Brown ACF2101 dual integrator is implemented to switch the BLM charge rapidly between the two integrators so as to collect all the charge produced. As one channel is integrating for 20 µs, the other is being digitized and reset. The integrated signal is then sent to the Analog-to-Digital Converter (ADC) to produce a number. The ADC converter used is Analog Devices AD7654. Figure. 1 shows a simplified illustration of the integrator circuit.
The digitizer has a 16 bit resolution and the scaling is such that one digitizer count represents 15.26 fC of charge in the integrator. Prior to each beam injection a pedestal value is measured to compensate for intentional and unintentional offsets or noise in the system. This signal is digitized and then subtracted from each integration period [3].

Data acquisition starts upon receipt of a TTL trigger. The 20 µs integrated analog signal is digitized into a 16 bit word. These 20 µs samples are summed into 80 µs samples and then divided by 4, the average is then written to a First In First Out (FIFO) memory. Data is collected for 40 ms at a rate of 12.5 kHz (80µs) for each Booster cycle to obtain 500 samples per cycle. There are a number of BLM channels in a particular location around the booster ring ranging from 12 – 24. Figure 2 shows a schematic of this process. These base 80 µs integration samples are transferred over the VME bus to be summed in different manners to represent accumulation of beam loss for different purposes [4]. A mathematical scaling for the integrator can be found in Appendix A.

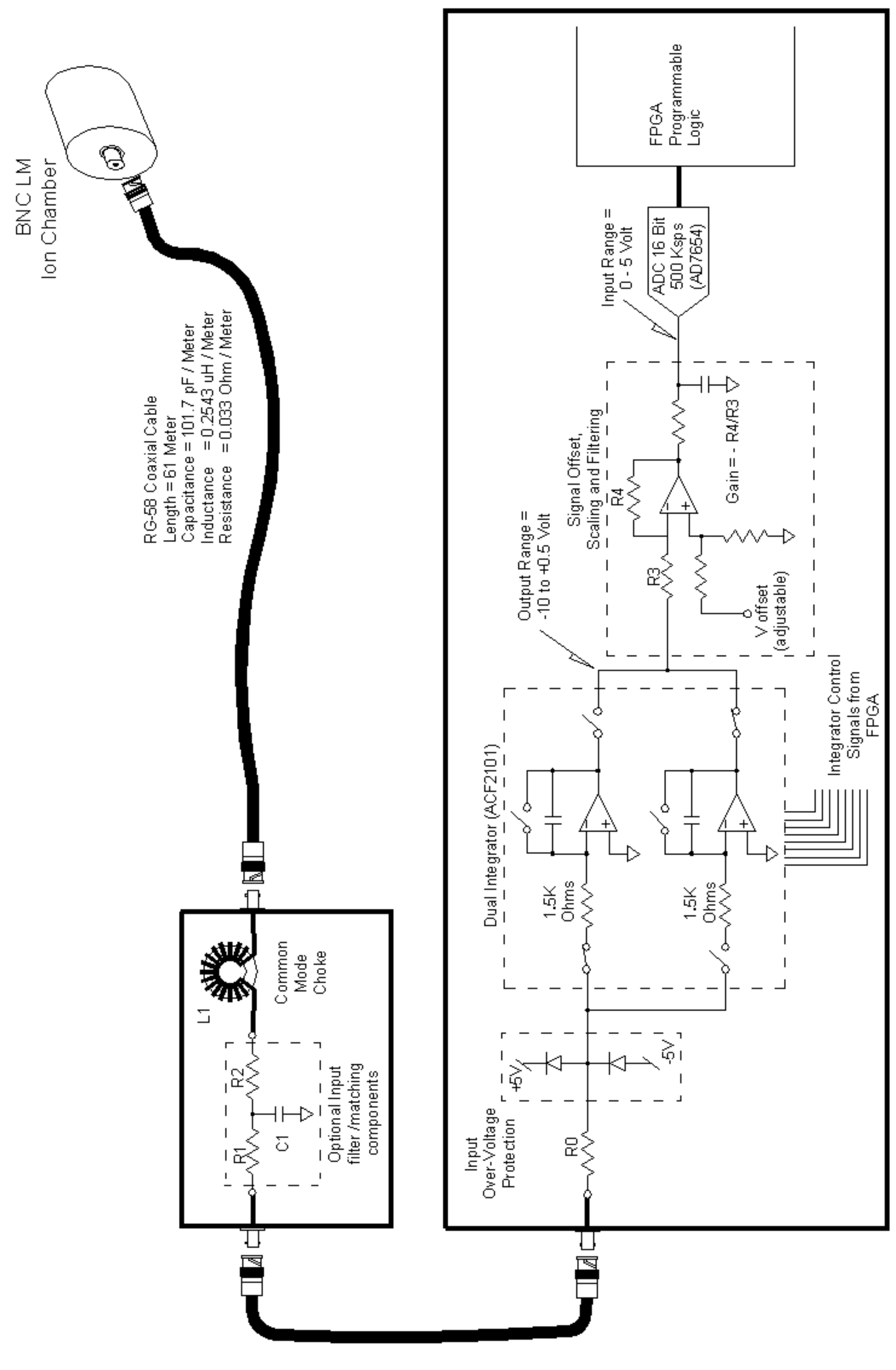

BNC LM
Ion Chamber
RG-58 Coaxial Cable
Length = 61 Meter
Capacitance = 101.7 pF / Meter
Inductance = 0.2543 uH / Meter
Resistance = 0.033 Ohm / Meter
L1
Common Mode Choke
R1
R2
C1
Optional Input filter /matching components
Input Over-Voltage Protection
R0
+5V
-5V
Dual Integrator (ACF2101)
1.5K Ohms
1.5K Ohms
Integrator Control Signals from FPGA
Output Range = -10 to +0.5 Volt
Signal Offset, Scaling and Filtering
R3
R4
V offset (adjustable)
Gain = - R4/R3
Input Range = 0 - 5 Volt
ADC 16 Bit 500 Ksps (AD7654)
FPGA Programmable Logic

FIG. 1: Figure shows a simplified BLM Integrator circuit. Current induced at the ion chamber is transferred via the RG-58 cable to the integrator card where the signal is integrated, digitized and sent to the FPGA.

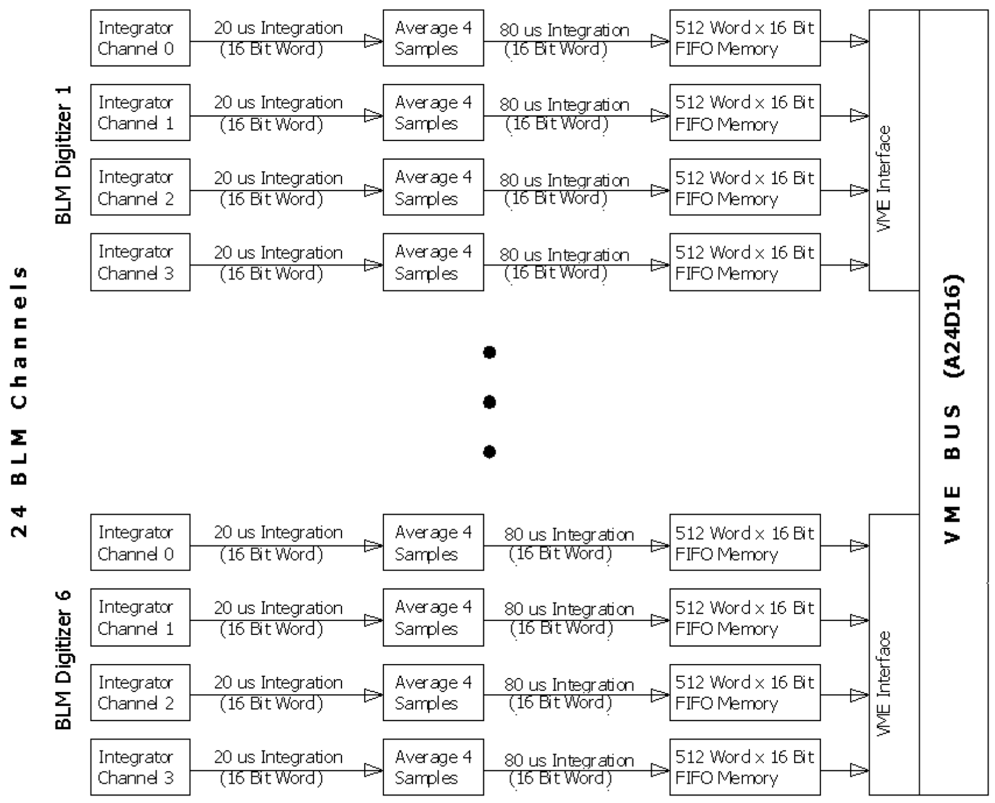


FIG. 2: Simplified block diagram of the BLM digitizer data processing. For each digitizer, the 20 µs signals are accumulated and the base 80 µs signal is averaged by 4 and written to the FIFO [4].

III. PROGRAMMING THE BOARD

The Integrator/ Digitizer board is controlled using software routines written in the C programming language. The program files along with the startup scripts are stored in the 'nova.fnal.gov' server in Fermilab's cloud computing environment. To communicate with the board we use an open source terminal emulator software called Tera Term. Tera Term communicates with the CPU (Motorola MVME2300 - MPC 603p) via the RS232 serial port. The information is then sent to the VxWorks OS. VxWorks is a real time operating system that is used for memory management and networking of the board. The VxWorks OS is stored in the non-volatile memory which communicates with Nova via the ethernet network. Figure 3 shows a schematic for the communication between Nova and Tera Term.

Similar to any other hardware and operating system, it is necessary to boot the MVME crate by either holding down the 'ctrl+x' keys on Tera Term or by pressing the 'rst' button on the MVME crate processor. The boot script is loaded from '/fecode-

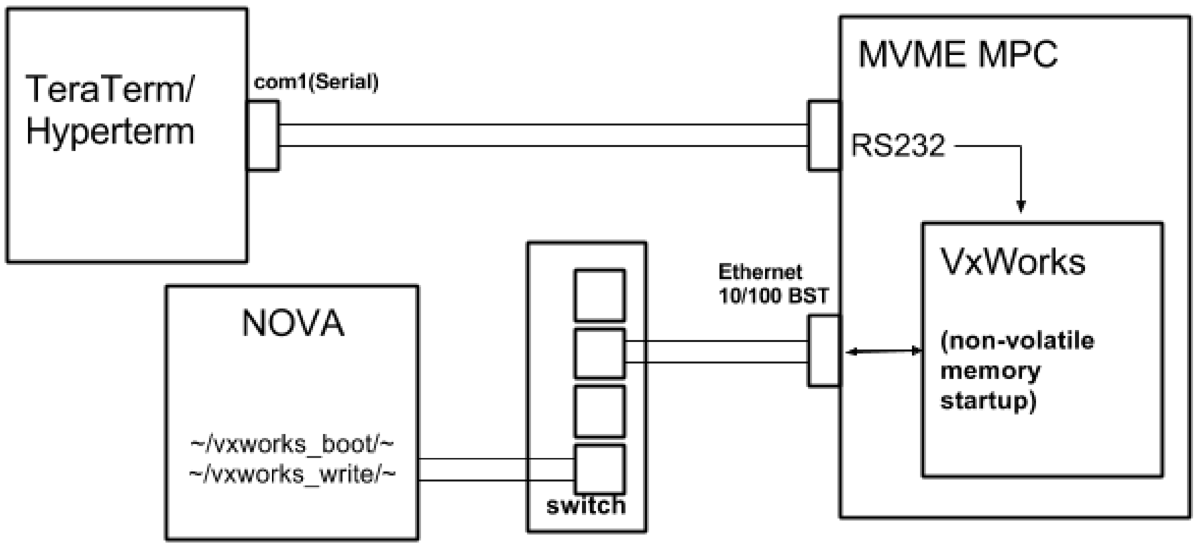


FIG. 3: Schematic for communication between Tera Term and Nova.

bd/vxworks_boot/fe/blrfd3/startup'. There are several startup scripts available for different testing purposes. The primary startup script used for miscellaneous testing is located in '.../blrfd3/startup_wallerTests'. The script is loaded by commenting out the other startup script files. When booting, the program files wallerTests.o, vme_test1.o and blmdev.o located in the same folder (.../blrfd3) are also loaded. These program files are compiled and generated within their respective directories by typing the 'make' command which invokes Makefile and by typing the 'pub' command, these files are copied into the '/blrfd3' folder. Analogous to the '/vxworks_boot' directory there is another folder, '/vxworks_write' is attached as a NSF drive for reading and writing data during run time. Any file that is to be loaded for writing or testing must be contained in this folder. In TeraTerm, the folder located in '/fecode-bd/vxworks_write/fe/proton/blrfd3' can be substituted as '/remote'.

A. User Interface for Programming FPGAs

An FPGA is a semiconductor device that can be programmed after its fabrication. It allows the programmer to access the product features and reconfigure the logic chip for specific applications (field-programmable). There are two FPGAs implemented to control the integrators, interface with the signal digitizers and the digital-to-analog-converters (DACs). The digitized signal from the ADC's are routed to both FPGAs. The upper FPGA, referred to as DCINTEG manages the sequencing and readout of the integrator channels and does scaling and averaging of the readings. The lower FPGA, referred to as DCSUMS manages a number of varying length sums of the integrator data and is the interface for the DAC analog outputs.

The FPGAs are manufactured by Altera and hence the development environment used to write and compile the FPGA code is Quartus. Before FPGA code is loaded into the module, a *.bst file must be created using ATMEL Programming System [5].

The FPGAs on the digitizer card are configured from the EEPROM device AT17LV002A at each power up. The contents of the EEPROM can be read and rewritten from the MVME CPU. Once the EEPROM is successfully rewritten, the FPGA will be configured with new firmware at next power up [3]. Previously, in order to program the FPGA, an ATMEL programming cable was connected between the parallel port of the computer and configuration connector. Now, once the *.bst file has been created and copied in the appropriate folder the user is able to program the FPGA directly via the CPU and the VME bus. In order to do this, the user may communicate with the CPU via Tera Term and enter either 'menufpga' or 'cmd' command when prompted with the 'BLRFD3–>' prompt to enter the respective programming modes. In the Menu/'menufpga' mode, the user is prompted with questions to indicate the specific digitizer board (iBoard) in the crate, which FPGA (INTEG/ SUMS) is to be accessed, and the name of the file (fname) to be read or written to. Similarly, in the command line/ 'cmd' mode, the user is able to do all this with a single line of command. The user has the following options for programming the FPGAs:

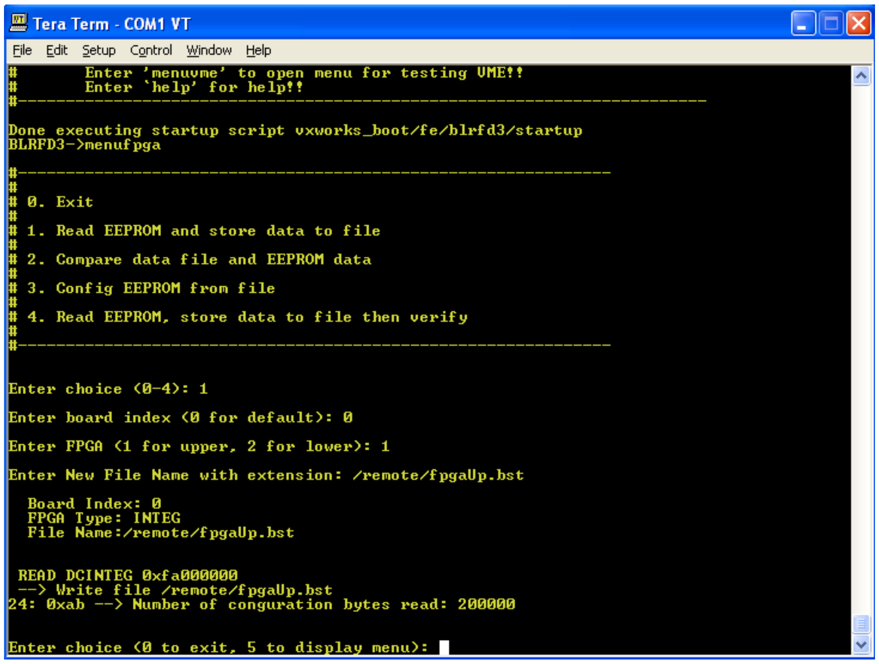
- Read EEPROM then store data to file: This will read the current EEPROM configuration for the selected FPGA from the board whose address is specified by iBoard. The data is then written to the file specified by fname in the ATMEL *.bst format.

- Config EEPROM from file: This will write the configuration EEPROM for the selected FPGA from the board whose address settings are specified by iBoard. The EEPROM is configured with data from the file specified by fname. The file is expected to be in the ATMEL *.bst format. Optionally, the data is read back out of the EEPROM and compared to the original file specified by fname.

- Compare data file and EEPROM data: This will read the configuration EEPROM for the selected FPGA from the board whose address settings are specified by iBoard. The data is then compared to the data in the file specified by fname. The file is expected to be in the ATMEL *.bst format.

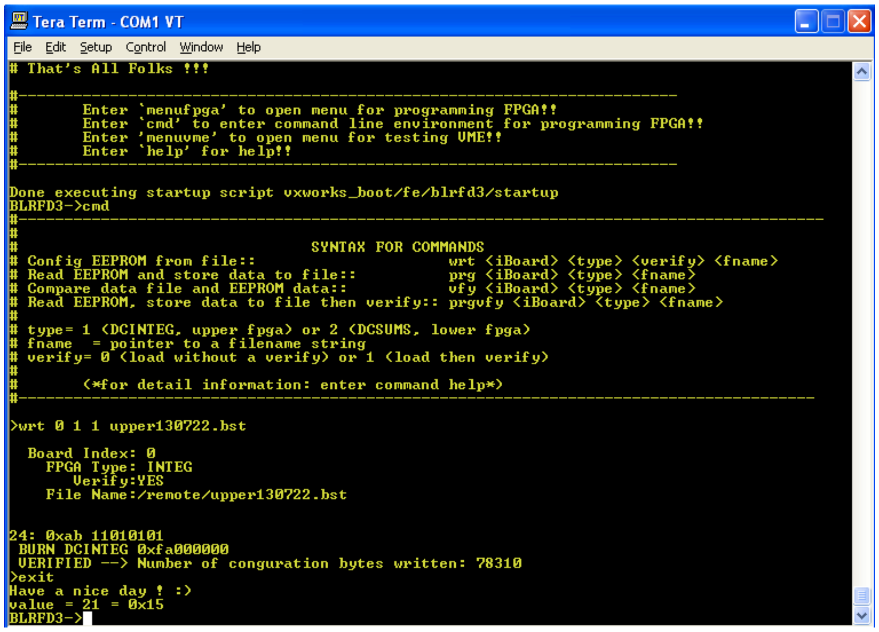
- Read EEPROM, store data to file then verify: This will read the current EEPROM configuration for the selected FPGA from the board whose address is specified by iBoard. The read data is then compared to the data in the file specified by fname.

The user interface also offers a help function to the user that describes the syntax and function of each command in detail. The user may access the help function by entering 'help'. Fig. 4 and Fig. 5 show a screen shot of the two modes that can be used for programming the FPGA. Once the EEPROM is successfully rewritten, the new FPGA configuration will be effective at the next power up.



```
#        Enter 'menuvme' to open menu for testing VME!!
#        Enter `help' for help!!
#---------------------------------------------------------------------------

Done executing startup script vxworks_boot/fe/blrfd3/startup
BLRFD3->menufpga

#----------------------------------------------------------------
#
# 0. Exit
#
# 1. Read EEPROM and store data to file
#
# 2. Compare data file and EEPROM data
#
# 3. Config EEPROM from file
#
# 4. Read EEPROM, store data to file then verify
#
#----------------------------------------------------------------

Enter choice (0-4): 1

Enter board index (0 for default): 0

Enter FPGA (1 for upper, 2 for lower): 1

Enter New File Name with extension: /remote/fpgaUp.bst

  Board Index: 0
  FPGA Type: INTEG
  File Name:/remote/fpgaUp.bst


 READ DCINTEG 0xfa000000
 --> Write file /remote/fpgaUp.bst
24: 0xab --> Number of conguration bytes read: 200000


Enter choice (0 to exit, 5 to display menu):
```

FIG. 4: Screenshot of the menu displaying the options to the user in Tera Term for programming the FPGA. Here, the user read the current configuration from the EEPROM and saves it as a file named fpgaUp.bst.



```
# That's All Folks !!!

#-------------------------------------------------------------------------
#        Enter `menufpga' to open menu for programming FPGA!!
#        Enter `cmd' to enter command line environment for programming FPGA!!
#        Enter 'menuvme' to open menu for testing VME!!
#        Enter `help' for help!!
#-------------------------------------------------------------------------

Done executing startup script vxworks_boot/fe/blrfd3/startup
BLRFD3->cmd
#------------------------------------------------------------------------------------------
#
#                                 SYNTAX FOR COMMANDS
# Config EEPROM from file::                        wrt <iBoard> <type> <verify> <fname>
# Read EEPROM and store data to file::             prg <iBoard> <type> <fname>
# Compare data file and EEPROM data::              vfy <iBoard> <type> <fname>
# Read EEPROM, store data to file then verify:: prgvfy <iBoard> <type> <fname>
#
# type= 1 (DCINTEG, upper fpga) or 2 (DCSUMS, lower fpga)
# fname  = pointer to a filename string
# verify= 0 (load without a verify) or 1 (load then verify)
#
#        (*for detail information: enter command help*)
#------------------------------------------------------------------------------------------

>wrt 0 1 1 upper130722.bst

  Board Index: 0
    FPGA Type: INTEG
       Verify:YES
    File Name:/remote/upper130722.bst


24: 0xab 11010101
 BURN DCINTEG 0xfa000000
 VERIFIED --> Number of conguration bytes written: 78310
>exit
Have a nice day ! :)
value = 21 = 0x15
BLRFD3->
```

FIG. 5: Screenshot of the command line mode for programming the FPGA. The program initially displays the syntax for all possible commands for the user and the user is able to program the FPGA using a single line command. Here, the user configured the EEPROM from the upper130722.bst file for the INTEG. Once configuration is complete, the data from the file and the data from the EEPROM is verified.

B. Testing the Digitizer Card

The digitizer card can be tested to ensure that the module is working as intended. For the purpose of this project, we performed three main types of tests.

For the first test, we supplied two known pulse inputs using a Tektronix AFG3102 Signal Source. The first pulse provides the TTL trigger to begin the measurement. The second pulse, represents the charge pulse from the BLM. There is a delay between the trigger pulse and the signal pulse to allow for the measuring the pedestal before the active integration begins. Figure 6 illustrates the connection of the test equipments [5]. By comparing the digitized output obtained from the module with the oscilloscope traces, the front panel inputs can be evaluated.

For the second test, the on board Digital-to-Analog-Converter (DAC) is used to send a fixed current to the input of the integrator. This value is set from the Test DAC register written in the FPGA. During this test mode, it is necessary to remember to turn off all external input sources and triggers. There is a DIP switch that works with the code to identify the board address and to select modes for testing the board. For the MVME 2301 the base

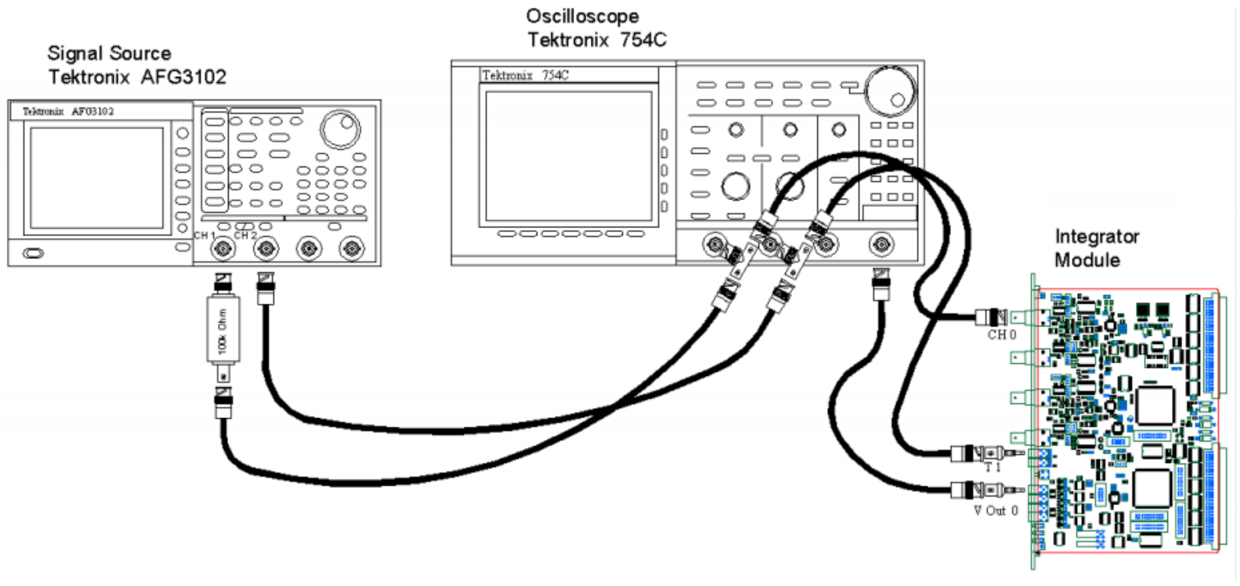


FIG. 6: Schematic that shows the connection from the signal source and the module. The signals are also connected to an oscilloscope for comparison purposes.

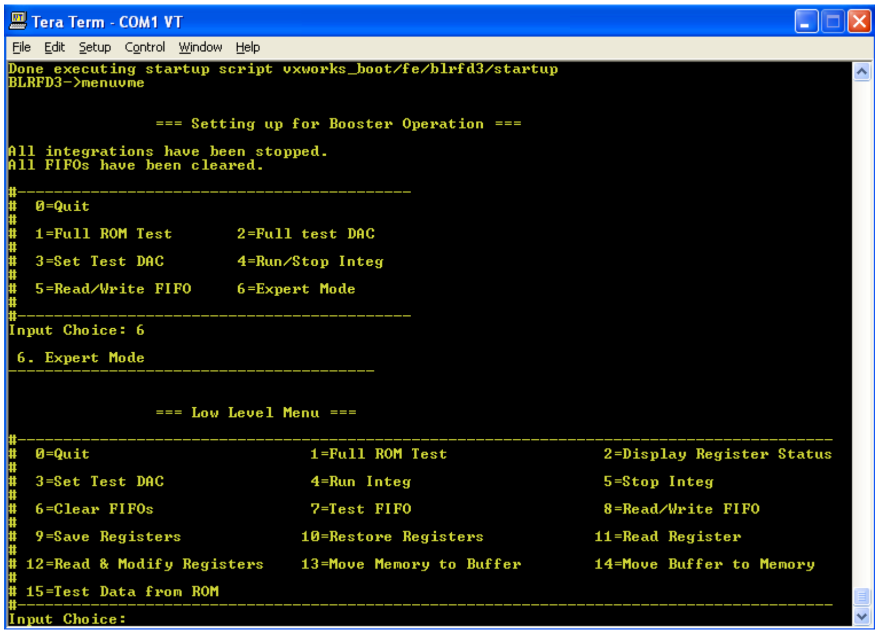
address is 0xfa000000. FPGA registers are accessed using memory addresses computed as follows:

$$\text{VMEAddress} = \text{0xfa000000} + (\text{iBoard} << 20) + \text{Reg\_Offset}.$$

A list of the memory addresses used along with their descriptions can be found in Appendix B.

To start the test, ensure that DIP switch 7 or command register (0x01000) bit 14 is set to logic 1 position to nullify the effect of the external trigger at T1. Once DIP switch 7 has been set, the Alternate Integrator Control register at address offset 0x01034 is enabled. By writing 0xff00 to this address, the front panel signal will be turned off and the test input will be switched on [6]. The board is now ready for testing.

The test input can be set by writing to the Test DAC Setting register at address offset 0x01048. The full scale output for the DAC is 0x7fff which is equivalent to 9.97 V. The converted analog signals are sent back to the integrator to be integrated and digitized. A 40 ms data acquisition cycle can be started and stopped by writing anything to address offset 0x01010 and 0x01012 respectively. The 500 samples collected are stored in the FIFO at address offsets 0x01020, 0x01022, 0x01024 and 0x01026 for channel 1, 2, 3 and 4 respectively. By comparing the data from the FIFO to the test DAC settings, the Integrator and Digitizer can be evaluated. The program for testing the module offers a high level and a



```
Done executing startup script vxworks_boot/fe/blrfd3/startup
BLRFD3->menuvme


                === Setting up for Booster Operation ===

All integrations have been stopped.
All FIFOs have been cleared.

#--------------------------------------------
#  0=Quit
#
#  1=Full ROM Test       2=Full test DAC
#
#  3=Set Test DAC        4=Run/Stop Integ
#
#  5=Read/Write FIFO     6=Expert Mode
#
#--------------------------------------------
Input Choice: 6

 6. Expert Mode
----------------------------------------


                === Low Level Menu ===

#------------------------------------------------------------------------------------------
#  0=Quit                           1=Full ROM Test                  2=Display Register Status
#
#  3=Set Test DAC                   4=Run Integ                      5=Stop Integ
#
#  6=Clear FIFOs                    7=Test FIFO                      8=Read/Write FIFO
#
#  9=Save Registers                10=Restore Registers             11=Read Register
#
# 12=Read & Modify Registers       13=Move Memory to Buffer         14=Move Buffer to Memory
#
# 15=Test Data from ROM
#------------------------------------------------------------------------------------------
Input Choice:
```

FIG. 7: Screenshot shows the high level and the low level tests developed for testing the module.

low level test mode for the user. The high level mode allows the user to perform a full DAC test. This option performs the full DAC test as described earlier. In addition, this mode reads the signal for each of the 4 channels, calculates the pedestal and displays the RMS, mean, minimum and maximum value. The high level mode also allows the user to check and modify the DAC status and, read/write and store data in the FIFO. The data in the FIFO is then be stored in a file of format 'fifoYYMMDD_HHmm.txt' where YYMMDD is the current date and HHmm is the current time in the given format. The low level test mode allows users to perform more controlled operations. It allows the user to read and modify registers at desired address offset. The user is able to perform each step in the entire process individually and check the status of the board. A screenshot of the program developed to test the module is shown in Fig. 7.

A third approach is to replace the digitized results of the integrated test DAC signal with fixed values from the ROM memory within the FPGA. The ROM data skips the integration and the digitization step and is directly stored into the FIFO. This test provides the option to skip the averaging of the data as well. The data stored in the FIFO is again compared to an expected data file computed from the known ROM values. The motivation behind creating this test was to check the update status of the data sent to ACNET. This test can be used to test the Averaging and the transfer of data via the VME bus. To switch the data acquisition from the test DAC to the ROM, a terminator (standard 50 Ω resistor) must be plugged into the Trigger Two (T2) input on the front panel. Similarly, in order to write the ROM data into the FIFO, the DIP switch 5 or command register bit 12 must be set to logic 1 position. Figure 8 shows simplified schematic for this process.

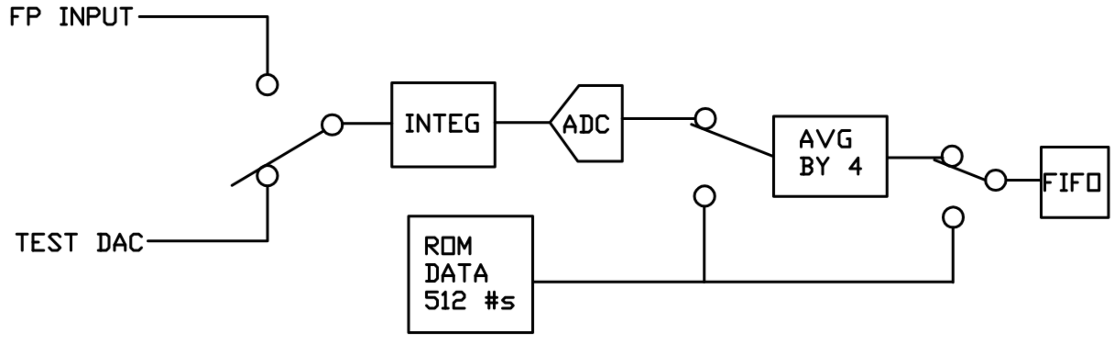


FIG. 8: A simplified block diagram of the BLM digitizer test. The diagram shows how the switches determine the source of the test data.

Similar to the full DAC test, the program also offers a full ROM test mode that reads test data from the ROM memory and writes it to the FIFO. The data written in the FIFO is then compared to a computed/expected file that the user could average by plugging in a terminator at T2 in the front panel. The expected data is stored into a file named either 'average_rom_data.txt' or 'rom_data.txt' depending on the averaging. These two data are then compared and the result is displayed.

In order to start testing the module, the user must open Tera Term and enter 'menuvme' command when prompted with the 'BLRFD3–>' prompt to enter the high level testing mode. From this mode, the user may enter the low level testing mode by selecting the appropriate option.

IV. CONCLUSION

In this project, we have successfully developed user interface to program and configure the FPGA via the EEPROM for the Digitizer/ Integrator card. The next immediate step is to develop the program to further test more specific components of the card. We have also developed several routines to test different components of the module. There can be up to 16 DCs in the crate, hence it was necessary to account for multi-board programming and testing. Although the program allows for multi-board programming and testing, no such test has yet been performed. The next step in this project would be to program the FPGA and perform tests for multiple boards.

V. ACKNOWLEDGMENTS

I would like to thank my supervisor, Craig Drennan for explaining the details about the BLM data acquisition process and the Digitizer/Integrator card. I would like to thank Diane Engram, Elliott McCrory and the entire SIST committee for giving me this opportunity to spend my summer at Fermilab.

[1] R. E. Shafer et al., *Comments on the Tevatron BLM System,* Fermilab BEAMS-DOC-790, July 2003.

[2] R. E. Shafer et al., *A Tutorial on Beam Loss Monitoring,* in proceedings of Beam Instrumentation Workship (NIW02), pp. 44-58, 6-9 May 2002, Upton, New York, USA.

[3] A. Baumbaugn et al., *Beam Loss Monitor Upgrades at Fermi National Accelerator Laboratory,* August 2011.

[4] C. Drennan, *Booster Beam Loss Monitor Data Acquisition and Presentation Specification.* Fermilab BEAM-DOCS-3723, December 2011.

[5] J. Lackey, C. Drennan *Booster Wire Scanner Integrator*. Fermilab BEAMS-DOC-3723, October 2009.

[6] C. Drennan, *Interfacing to the Booster BLM Upgrade Integrator/Digitizer VME Module.* Fermilab BEAM-DOCS-3780, February 2011.

Appendix A: BLM Digitizer Calibration – Scaling for the Integrator

The combinations of integrators and ADC results in a conversion between Coulombs of charge and the resulting 16 bit digitized output. In the default low range mode, the integration opamp contains a 100 pF feedback capacitor. This produces a voltage out of the integration opamp of 1.0×10^{-14} V/C. The integrator output is then scaled to fit the input range of the ADC by a scale factor of 0.483.

The integrator output voltage digitized each 20µs sampling interval is the sum of the charge collected in the previous 20µs interval to get a final sample of 80µs. The voltage at the input to the ADC is

$$\mathrm{V(t)} = \frac{0.4827}{1.0 \times 10^{-14}} \int_{\mathrm{t}}^{\mathrm{t+T/4}} \mathrm{Q_{in}(\tau)d\tau},$$

where T = 80µs. Assume that the k$^{\mathrm{th}}$ digitized 20µs integration sample as Y(kT/4). The digitizer output a 16 bit value and has an input voltage range of 5 V.

$$\mathrm{Y(kT/4)} = \frac{2^{16}}{5}\mathrm{V(kT/4)}.$$

The digitized value written to the FIFO memory on the module and read by the front end processor is the average of 4 of these 20µs integration samples. These can also be described as scaled 80µs integration samples.

$$\begin{aligned}
\mathrm{A(kT)} &= \frac{2^{16}}{5}\frac{1}{4}\left[\mathrm{V(kT/4) + V(2kT/4) + V(3kT/4) + V(kT)}\right] \\
&= \mathrm{G_1}\left[\int_{\mathrm{(k-1)T}}^{\mathrm{kT/4}} \mathrm{Q_{in}(\tau)d\tau} + \int_{\mathrm{kT/4}}^{\mathrm{2kT/4}} \mathrm{Q_{in}(\tau)d\tau} + \int_{\mathrm{2kT/4}}^{\mathrm{3kT/4}} \mathrm{Q_{in}(\tau)d\tau} + \int_{\mathrm{3kT/4}}^{\mathrm{kT}} \mathrm{Q_{in}(\tau)d\tau}\right] \\
&= \mathrm{G_1}\int_{\mathrm{(k-1)T}}^{\mathrm{kT}} \mathrm{Q_{in}(\tau)d\tau},
\end{aligned}$$

where $\mathrm{k} = 1, 2, 3, \ldots, 500$ and $\mathrm{G_1} = \frac{2^{16}}{5}\frac{1}{4}\frac{0.4827}{1.0\times10^{-14}} = 15.817 \times 10^{12}$ bits/C 7.

Appendix B: Description of Specific Registers

TABLE I: Internal Registers and Control

Register	**Address**	**R/W**	**Description**
Command Register	0x01000	–	See Table II
Start DAQ Command	0x01010	W	Send command to start the 40ms data acquisition.
Stop DAQ Command	0x01012	W	Send command to stop the 40ms data acquisition.
Clear FIFO Command	0x01014	W	Send command to clear the data FIFO's.
Channel 1 Data FIFO	0x01020	R	Data FIFO output port. This FIFO channel is also mapped to the address range 0x01200 to 0x013FF.
Channel 2 Data FIFO	0x01022	R	Data FIFO output port. This FIFO channel is also mapped to the address range 0x01400 to 0x015FF.
Channel 3 Data FIFO	0x01024	R	Data FIFO output port. This FIFO channel is also mapped to the address range 0x01600 to 0x017FF.
Channel 4 Data FIFO	0x01026	R	Data FIFO output port. This FIFO channel is also mapped to the address range 0x01800 to 0x019FF.
Number of records in Channel 1 FIFO	0x01028	R	
Number of records in Channel 2 FIFO	0x0102A	R	
Number of records in Channel 3 FIFO	0x0102C	R	
Number of records in Channel 4 FIFO	0x0102E	R	
FIFO Status Register	0x01030	R	See Table III
Alternate Integrator Control Register 1	0x01032	R/W	See Table IV
Alternate Integrator Control Register 2	0x01034	R/W	See Table V
Channel 1 Average Register	0x01038	R	
Channel 2 Average Register	0x0103A	R	
Channel 3 Average Register	0x0103C	R	
Channel 4 Average Register	0x0103E	R	
Test DAC Setting Register	0x01048	R/W	Setting for the Test DAC. Full scale output is 0x7FFF.

TABLE II: Command Register

Bit	**Description**
0	Error signal J2
1	Use Alternate Integrator Control Signal Register 1. This is a VME controlled register used to manipulate the individual integrator Hold, Select, and Reset switches integral to the TI/ Burr Brown ACF2101 dual integrators.
2	Disable Baseline Subtraction when logic high.
3	Unassigned
5..4	Test Vector Select Bits
11..6	Unassigned
12	Dip Switch 5
13	Dip Switch 6
14	Dip Switch 7
15	Unassigned

TABLE III: FIFO Status Register

Bit	**Description**
0	FIFO Channel 1 Full
1	FIFO Channel 1 Empty
2	FIFO Channel 2 Full
3	FIFO Channel 2 Empty
4	FIFO Channel 3 Full
5	FIFO Channel 3 Empty
6	FIFO Channel 4 Full
7	FIFO Channel 4 Empty
15..8	Unassigned

TABLE IV: Alternate Integrator Control Register 1
(This register is enables only if Bit 1 of the Command Register is High)

Bit	**Description**
0	Hold off the output of Channel 1 and 2 side B
1	Select Channel 1 and 2 side B for digitization
2	Reset the B side integrators on Channel 1 and 2
3	Hold off the output of Channel 3 and 4 side B
4	Select Channel 3 and 4 side B for digitization
5	Reset the B side integrators on Channel 3 and 4
6	Hold off the output of Channel 1 and 2 side A
7	Select Channel 1 and 2 side A for digitization
8	Reset the A side integrators on Channel 1 and 2
9	Hold off the output of Channel 3 and 4 side A
10	Select Channel 3 and 4 side A for digitization
11	Reset the A side integrators on Channel 3 and 4
15..12	Unassigned

TABLE V: Alternate Integrator Control Register 2
(This register is enables only if either DIP switch SW7 is High or bit 14 of the Command Register is High)

Bit	**Description**
0	Switch the integrating capacitor to 500pF in Channel 1
1	Switch additional 16kΩ resistor in series with input for Channel 1
2	Switch the integrating capacitor to 500pF in Channel 2
3	Switch additional 16kΩ resistor in series with input for Channel 2
4	Switch the integrating capacitor to 500pF in Channel 3
5	Switch additional 16kΩ resistor in series with input for Channel 3
6	Switch the integrating capacitor to 500pF in Channel 4
7	Switch additional 16kΩ resistor in series with input for Channel 4
8	Switch the Test Input on for Channel 1
9	Switch the Test Input on for Channel 2
10	Switch the Test Input on for Channel 3
11	Switch the Test Input on for Channel 4
12	Switch Front Panel Signal into integrator for Channel 1
13	Switch Front Panel Signal into integrator for Channel 2
14	Switch Front Panel Signal into integrator for Channel 3
15	Switch Front Panel Signal into integrator for Channel 4