User's Guide

Revision History

Date	Description
6/19/2014	Version 1.0
9/22/2014	Version 1.1 with updated sections 2.3.4, 5, 6.3, and Appendices

http://openpet.lbl.gov

Table of Contents

Table of Figures **4**

Abbreviations **6**

1 System Overview **7**

1.1 Definitions *7*

1.1.1 Support Crate 8

1.1.2 Detector Unit 8

1.1.3 Coincidence Unit 9

1.2 System Configuration *9*

1.2.1 Small System 10

1.2.2 Standard System 10

1.2.3 Large System 11

1.3 Timing & Timing Signals *12*

1.3.1 System Clock 12

1.3.2 Time Slice Boundary 13

1.3.3 Time Slice 13

1.4 Firmware and Software Structures *13*

1.4.1 Large System 13

1.4.2 Small System 16

2 Getting Started – *Small System Oscilloscope Mode* **18**

2.1 Getting the Hardware *18*

2.2 Assembling the Hardware *18*

2.3 Downloading the Software and Firmware *20*

2.3.1 Installing QuickUSB 20

2.3.2 Installing Altera Tools 23

2.3.3 Installing USB-Blaster Driver 23

2.3.4 Installing OpenPET Firmware & Software 25

2.3.4.1 Programming OpenPET flash Images 26

2.4 Running a Small System in Oscilloscope Mode *28*

2.4.1 Commands 28

2.4.2 Data Acquisition 31

2.4.3 Example System Setup & Data Acquisition 31

2.5 Data Analysis *33*

2.5.1 Data Format 33

2.5.1.1 Single Detector Board 33

2.5.1.2 Multiple Detector Boards 37

2.6 Getting Help *39*

3 Detector Board **40**

3.1 Bus IO *40*

3.2 16-Channel Detector Board *43*

3.2.1 Analog Signal Conditioning 45

3.2.2 Timing Signal 45

3.2.3 ADC 45

3.2.4 TDC 45

3.2.5 Detector Memory 45

3.2.6 Analog Input Connections 45

4 Support Board **47**

4.1 Slave FPGAs *48*

4.2 Master FPGA & Support Microprocessor *48*
4.3 Support Memory *48*
4.4 Clock Conditioning *49*
4.5 Connectors, Slots 0-7 ..... *49*
4.6 Slots 8–11 ..... *49*
4.6.1 Coincidence Interface Board (Slot 8) 49
4.6.2 Host PC Interface (Slot 9) 49
4.6.3 User IO (Slot 10) 49
4.6.4 Debugging (Slot 11) 50

5 Commands 51

6 OpenPET Control and Analysis Tools (OpenPET CAT) 61
6.1 Installing ROOT *61*
6.2 Using OpenPET CAT *61*
6.2.1 Configuration Files 61
6.2.2 OpenPET CAT Library and Macros 65
6.3 OpenPET CAT Graphic User Interface *66*
6.3.1 Introduction 66
6.3.2 Configure System 66
6.3.3 Acquire Data 69
6.3.4 Display Waveform 70

7 Acknowledgements 73

8 Index 73

9 Appendices 74
9.1 Appendix 1: 16-Channel Detector Board Default Values *74*
9.2 Appendix 2: Troubleshooting Diagnostics *75*
9.2.1 Test Digital Communication Chain 75
9.2.2 Test Analog with Internal Trigger 77
9.2.3 Test Trigger 78

Table of Figures

Figure 1: Schematic of the OpenPET system architecture. 7
Figure 2: Support Crate (left) and Detector Unit (right). A Detector Unit is a Support Crate with up to 8 Detector Boards (in Slots 0-7). For a Small System, Slot 8 is usually empty, although a Coincidence Interface Board can (optionally) be plugged into it. For Standard and Large Systems, a Coincidence Interface Board must be plugged into Slot 8. 8
Figure 3: Coincidence Unit for a Standard System (left) and a Large System (right). For both Standard and Large Systems, a Coincidence Unit is a Support Crate with up to 8 Multiplexer Boards (in slots 0-7). In a Large System, the Multiplexer Boards function as multiplexers. 9
Figure 4: Configuration of a Small OpenPET System. 10
Figure 5: Configuration of a Standard OpenPET System. Each of slots 0-7 in the CU contains a Multiplexer Board-1 that services a single DU. 11
Figure 6: Configuration of a Large OpenPET System. Each of slots 0-7 in the CU contains a Multiplexer Board-8 that operates as a multiplexer, allowing that slot to service between 1 and 8 DUs. 12
Figure 7: System level timing signals. 13
Figure 8: Tree network topology of an OpenPET Large System. A Coincidence Unit Controller (CUC) that supports 8 Multiplexer Boards is housed in a crate with a power supply. The assembled unit is called a Coincidence Unit. A Detector Unit Controller (DUC) that supports 8 Detector Boards is also housed in a crate with a power supply. The assembled unit is called a Detector Unit. One Coincidence Unit fans out to up to 64 Detector Units. The hardware of the CUC and DUC are the same (Support Board). The differences are the firmware and software configurations. 15
Figure 9: Firmware and software structure for a Large System. 16
Figure 10: The tree network topology for a Small System. A CDUC performs the functions of both the CUC and DUC. 17
Figure 11: (a) Empty Support Crate viewed from the front. (b) Support Board viewed from the front. .. 19
Figure 12: (a) Close up of the small gap (~4 mm) between the Support Crate and the edge of the Support Board when assembled properly. (b) Close up of the power cables attached to the back of the Support Board. (c) Close up of the jumper board plugged into the back of the Support Board. 19
Figure 13: The assembled OpenPET Support Crate viewed from the (a) front and (b) back. 20
Figure 14: QuickUSB module mounted onto the (a) Host PC Interface Board or (b) Support Board..... 20
Figure 15: Bitwise Systems QuickUSB Library v2.15.1 InstallShield Wizard: a) welcome window, (b) destination folder window, (c) installing the library window, and (d) installing the driver window... 21
Figure 16: QuickUSB Programmer used to program the QuickUSB module with the correct firmware: QuickUSB QUSB2 Module v2.15.1 (Block Handshake). 22
Figure 17: QuickUsb Diagnostics used to confirm the QuickUSB module with the correct firmware: QuickUSB QUSB2 Module v2.15.1 (Block Handshake). 22
Figure 18: Altera Tools installation windows: (a) download center window, (b) Quartus II Web Edition window, (c) Akamai NetSession Interface window, (d) User Account Control confirmation window, (e) Quartus II Web Edition Setup Wizard window and (f) Altera Nios2 Command Shell window... 24
Figure 19 OpenPET Directory Tree (***OpenPET_ROOT***) 25
Figure 20 Windows Environment Variable through control panel 26
Figure 21 Setting the PATH environment variable 26
Figure 22: (a) Detector Board with jumpers installed on pins 1 and 2 for J1, J2 and J3. (b) USB-Blaster plugged into the JTAG connector on the back of the Support Board. A custom jumper board is also shown. 27
Figure 23 Support Board flash programming script 27
Figure 24 Detector Board flash programming script 28
Figure 25: OpenPET address formats. Bits shown in white are not used. 29

Figure 26: TestData1.dat example data file generated from above command sequence, with 32 samples per channel. See Section 2.5 for more details on the data format and Section 9.2 on diagnostic testing. ... 32
Figure 27: Definition of the Oscilloscope Mode data format for an ADC+TDC data train, where the maximum N is 254. ... 34
Figure 28: Format of the 32-bit Starting word 1 for an Oscilloscope Mode data train. ... 34
Figure 29: Format of the 32-bit Starting word 2 for an Oscilloscope Mode data train. ... 35
Figure 30: Format of the 32-bit raw ADC data for an Oscilloscope data train. ... 36
Figure 31: Format of the 32-bit raw TDC data for an Oscilloscope data train. ... 36
Figure 32: Format of the 32-bit Ending word for an Oscilloscope Mode data train. ... 37
Figure 33: Example transmission of Oscilloscope Mode data trains from the CDUC to the Host PC. ... 38
Figure 34: Diagram of the BUS IO. ... 41
Figure 35: Connections between the Support Board and the Detector Board. ... 42
Figure 36: Block diagram of the 16-channel Detector Board. ... 43
Figure 37: Block diagram of the front-end circuitries of one channel of the 16-channel Detector Board. ... 44
Figure 38: Photograph of the 16-channel Detector Board. ... 44
Figure 39: Pin assignment for the Analog Input Connector to the 16-channel Detector Board. ... 46
Figure 40: Schematic of the Support Board. ... 47
Figure 41: OpenPET command/response address formats. Bits shown in white are not used. ... 51
Figure 42: Example system configuration for a small system. ... 62
Figure 43: Example Multiplexer Board configuration file. ... 63
Figure 44: Example Detector Unit configuration file. ... 63
Figure 45: Example Detector Board configuration file. ... 64
Figure 46: Initial startup windows. ... 66
Figure 47: Sections of the Configuration window. ... 67
Figure 48: Opening a system configuration file. ... 67
Figure 49: Selecting MB and DU addresses. ... 68
Figure 50: Detector Board configuration file within the Configuration window. ... 68
Figure 51: Acquire Data window. ... 69
Figure 52: Initial Display Waveform window. ... 70
Figure 53: Display Waveform. ... 71
Figure 54: Example highlighting number of skipped events. ... 72
Figure 55: The event index (x-axis) as a function of the length of a data train in bytes (y-axis). Data were acquired for 5 seconds using 240 raw ADC samples per channel and test communication data format. ... 76
Figure 56: Waveforms for internally generated counter data, shown for 600th data train and all 16 DB channels with 240 ADC samples/channel. For each channel, the ADC sample index (x-axis) is plotted as a function of the ADC value (y-axis). ... 77
Figure 57: Waveforms for all 16 DB channels, when a sinewave was inputted into only channel 0 and data were acquired with 240 raw ADC samples/channel, test analog data format, and internal clock trigger. For each channel, the sample number index (x-axis) is plotted as a function of amplitude (y-axis). ... 78

Abbreviations

CDUC: Coincidence Detector Unit Controller
CI: Coincidence Interface Board
C/R: Commands and Responses
CU: Coincidence Unit
CUC: Coincidence Unit Controller
DB: Detector Board
DU: Detector Unit
DUC: Detector Unit Controller
EPCS: Enhanced Programming Configuration Serial device
FIFO: First-in, First-out Data Buffer
FPGA: Field-Programmable Gate Array
MB: Multiplexer Board
PLL: Phase-Locked Loop
SB: Support Board
TDC: Time-to-Digital Converter

1 System Overview

This document describes the OpenPET electronics architecture. The purpose of the OpenPET electronics is to provide a system that can be used by a large variety of users, primarily people who are developing prototype nuclear medical imaging systems. These electronics must be extremely flexible, as the type of detector, camera geometry, definition of event words, and algorithm for creating the event word given the detector outputs will vary from camera to camera. This implies that users must be able to modify the electronics easily, which further implies that they have easy access to documentation, including the schematics and documents needed to fabricate the circuit boards (Gerber files, bill of materials, etc.) and source code (for both firmware and software). They also need support, in the form of instructions, user manuals, and a knowledge base, and they want fabricated circuit boards to be readily available. Thus, the OpenPET system includes hardware, firmware, and software. It is scalable enough to provide solutions ranging from a "test bench" for a small number of detector modules to a complete camera, and it is "open source" to both maximize flexibility and minimize redundant development.

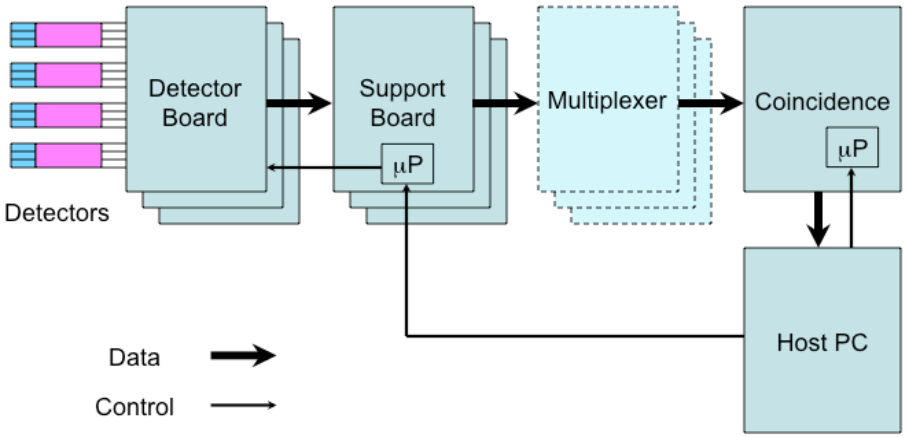


Figure 1: Schematic of the OpenPET system architecture.

The system architecture is shown in Figure 1.There are four types of custom electronics boards in the system: the Detector Board (DB), the Support Board (SB), the Multiplexer Board (MB), and the Coincidence Board. The Coincidence Board is identical to the Support Board, but has different FPGA firmware loaded into it. The general data flow is that analog signals from detector modules provide the inputs to the Detector Board. This board processes the analog signals to create a Singles Event Word, which is a digital representation of this single gamma ray interaction. These Singles Event Words are passed to the Support Board, whose main function is to multiplex the Singles Event Words from multiple Detector Boards. The Multiplexer Board is optional—it can provide a further layer of multiplexing for Singles Event Words, if desired. The multiplexed Single Event Words are then passed to the Coincidence Board, which searches through the Singles Event Words for pairs that are in time coincidence and forms a Coincidence Event Word when it does so. These Coincidence Event Words are then passed to the Host PC. Optionally, the Coincidence Board can act as a multiplexer and pass unaltered Singles Event Words to the Host PC. Control Signals originate from the Host PC, are passed to microprocessors that are on the Coincidence Board and Support Board, and are forwarded from there.

1.1 Definitions

The OpenPET components are housed in an assembly whose form factor is the same as a 12-slot VME 6U crate. A Support Board essentially replaces the backplane of the VME crate and all the other boards plug into it. The plug-in boards have the same form factor as a VME 6U board, except that the position of the connectors is offset (compared to true VME boards) to prevent OpenPET boards from being plugged into standard VME systems and vice versa.

1.1.1 Support Crate

A Support Crate (Figure 2) is conceptually similar to a VME crate (with controller), namely an intelligent support structure that “functional” boards can be plugged into. It consists of a mechanical frame with 12 plug-in slots, a Support Board (that has a considerable amount of programmable processing power and also acts as a backplane), power supplies, cooling fans, and appropriate boards plugged into slots 9–11. Slots 0–8 are vacant. Slot 9 holds a Host PC Interface Board, which is used to communicate with the Host PC; this board is required. Slot 10 holds a User IO Board, which allows users to interface to external components such as EKG signals and motor controllers; this board is optional. Slot 11 holds a Debugging Board, which has interfaces to logic analyzers, a number of diagnostic LEDs, an external clock input, and a JTAG connector; this board is optional. Some ancillary components (such as DRAM memory and a QuickUSB board) are also necessary for a functioning Support Crate. By programming the Support Board with appropriate (but different) firmware, the Support Crate becomes part of either a Detector Unit or a Coincidence Unit.

1.1.2 Detector Unit

A Detector Unit (DU), as shown in Figure 2 consists of a Support Crate with between one and eight Detector Boards plugged into slots 0–7. Each Detector Board can process up to 32 analog input signals. A Detector Unit can therefore process up to 256 analog signals, which corresponds to 64 conventional block detector modules (with 4 analog outputs per module). In a Small System, Slot 8 is empty (if data is transferred to the Host PC through USB or Ethernet via the Host PC Interface Board plugged into Slot 9). In a Standard or Large System, a Coincidence Interface Board must be plugged into Slot 8 of the Detector Unit. There are two versions of the Coincidence Interface Board: Coincidence Interface Board-1 (CI-1) for the Standard System and Coincidence Interface Board-8 (CI-8) for the Large System. At present, the Coincidence Interface Board-8 has not been designed or specified. These boards transfer event data and bidirectional control data between the Detector Unit and a Coincidence Unit.

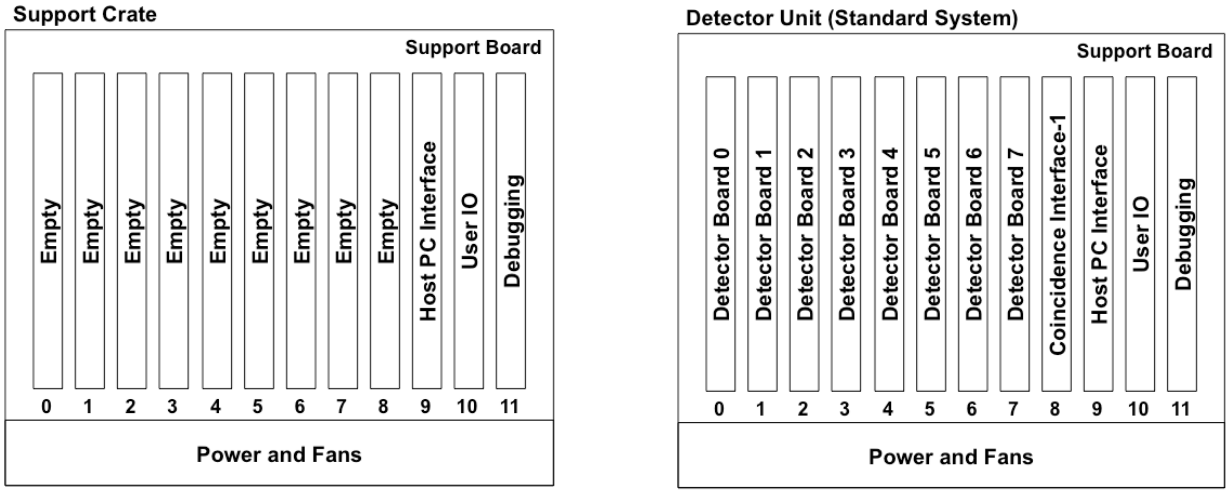


Figure 2: Support Crate (left) and Detector Unit (right). A Detector Unit is a Support Crate with up to 8 Detector Boards (in Slots 0-7). For a Small System, Slot 8 is usually empty, although a Coincidence Interface Board can (optionally) be plugged into it. For Standard and Large Systems, a Coincidence Interface Board must be plugged into Slot 8.

In a Small System (see Section 1.2.1), the Support Board in the Detector Unit is programmed to multiplex outputs from the Detector Boards, process coincident events, and pass the coincident events to the Host PC. It can also be programmed to multiplex singles events and pass them to the Host PC. In a Standard or Large System (see Sections 1.2.2 and 1.2.3), the Support Board in the Detector Unit is programmed to multiplex singles events from the Detector Boards and forward them to a Coincidence Unit.

1.1.3 Coincidence Unit

In Small Systems the coincidence processing is performed on the Detector Unit's Support Board. In Standard and Large Systems the coincidence processing is performed in a Coincidence Unit (CU) as shown in Figure 3. The Coincidence Unit for a Standard System consists of a Support Crate with between one and eight Multiplexer Boards plugged into slots 0–7. The Support Board is loaded with firmware to perform the coincidence processing. Each Multiplexer Board communicates with one Detector Unit via the Coincidence Interface Board using a cable. Similar to the Coincidence Interface Board, there are two versions of the Multiplexer Board: Multiplexer Board-1 (MB-1) for the Standard System and Multiplexer Board-8 (MB-8) for the Large System. At present, the Multiplexer Board-8 has not been designed or specified. The Coincidence Unit's Support Board is programmed to do the coincidence processing and pass the coincident events to the Host PC, although it can also function as a multiplexer and forward singles events. Data is transferred to the Host PC either through USB or Ethernet via the Host PC Interface Board plugged into Slot 9.

In the Coincidence Unit for the Standard System, each MB-1 connects with only one Detector Unit via a single cable, allowing up to 64 Detector Boards (or 512 block detector modules) in the system. In the Coincidence Unit for a Large System, the MB-8s plugged into slots 0–7 connect via cables (one cable per Detector Unit) with up to 8 Detector Units, allowing up to 512 Detector Boards (or 4096 block detector modules) in the system. The MB-8s are programmed to serve as multiplexers for events coming from up to 8 Detector Units. Due to the nature of multiplexing, this allows a larger number of channels to be serviced, but does not increase the maximum total event rate (singles or coincidence).

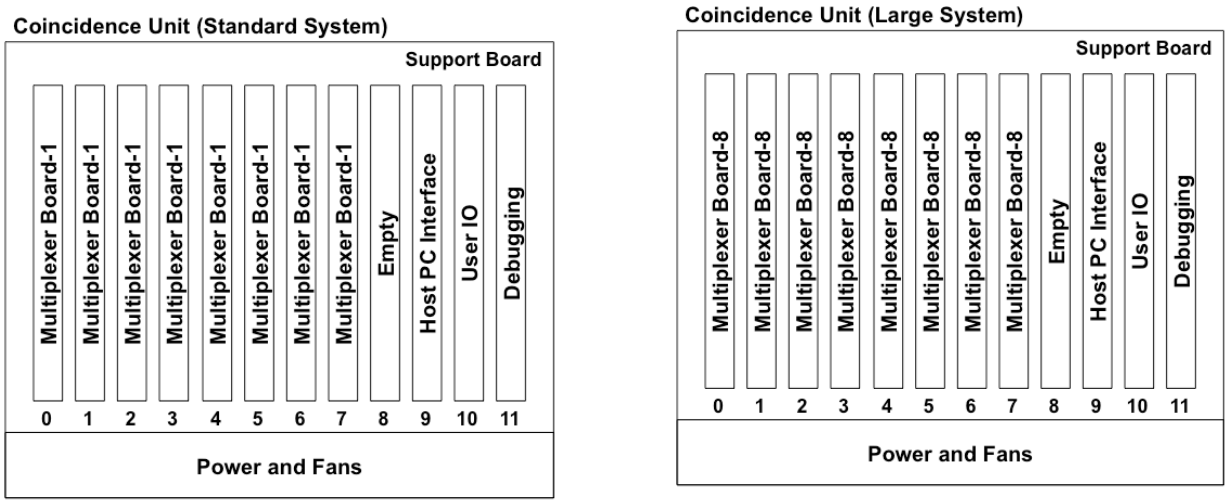


Figure 3: Coincidence Unit for a Standard System (left) and a Large System (right). For both Standard and Large Systems, a Coincidence Unit is a Support Crate with up to 8 Multiplexer Boards (in slots 0-7). In a Large System, the Multiplexer Boards function as multiplexers.

1.2 System Configuration

OpenPET can be configured either as a Small System, a Standard System, or a Large System, with the difference largely due to the number of analog signals that can be read out. To determine which system components you need, the first step is to determine how many DBs are necessary. Each DB can process up to 16 or 32 analog signals (depending on the DB used, mixing of different types of DB is not supported), so the minimum number of DBs necessary is the number of analog signals divided by 16 or 32. While information can be shared between DBs, processing is far easier if all the signals from the same detector module are on the same DB. Thus, if your detector module has 5 analog outputs and you use a 32-channel DB, it is easiest to have each DB process 30 analog signals (*i.e.,* from six detector modules) and not use the other two channels on each Detector Board. Thus, the initial estimate for the number of DBs needed is the number of analog signals divided by the number of analog signals you will have each DB process. This estimate may be modified due to the camera topology, as described in the following subsections. Once you have determined this initial estimate for the

number of Detector Boards, you can determine whether you will need a Small, Standard, or Large OpenPET System.

1.2.1 Small System

If the total number of Detector Boards is 8 or fewer, you can use a Small System. This consists of a single Detector Unit (defined in Section 1.1.2) connected to a Host PC (Figure 4). A DU can have anywhere between 1 and 8 DBs plugged into it. The initial estimate of the number of DBs in your system may need to be increased to support the camera topology. In the Small System, the default coincidence processing algorithm searches for coincidences between singles events that originate on different DBs, but it doesn't allow coincidences between singles events that originate on the same DB. Thus, more DBs may be necessary to allow all the desired coincidences, as none of the modules in a DB can be in coincidence with each other.

As an example, consider a four-headed PET system, where each head consists of a 3x3 array of detector modules and each detector module has five analog outputs. The OpenPET system would be configured as follows. Each 32-channel DB would service 6 detector modules, using 30 analog channels and leaving two channels on each DB unused. As the system consists of 36 detector modules (four heads of nine modules each), the initial estimate for the number of DBs is six. To see whether the appropriate coincidences can be accommodated, we first try to distribute the detector modules as follows. DB 0 services six modules from head 0, DB 1 services three modules from head 0 and three from head 1, DB 2 services six modules from head 1, DB 3 services six modules from head 2, DB 4 services three modules from head 2 and three from head 3, and DB 5 services six modules from head 3. Unfortunately, this will not work, as two Detector Boards (numbers 1 and 4) service modules from two different heads, which means that the system will not look for all possible coincidences between detector modules that are in different heads. No amount of redistributing the modules among the DBs will satisfy these criteria either. Thus, the only way the coincidence criteria can be satisfied (using the default coincidence processing software) is to use two DBs per head, for a total of eight Detector Boards. While the coincidence processing software can be rewritten to allow coincidences between two singles events that originate in the same DB, this is likely to take significantly longer and cost more than purchasing two additional Detector Boards.

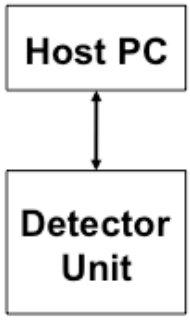


Figure 4: Configuration of a Small OpenPET System.

1.2.2 Standard System

If the total number of DBs is between 9 and 64, you can use a Standard System. This consists of between 2 and 8 DUs (defined in Section 1.1.2) and a single CU (defined in Section 1.1.3) connected to a Host PC (Figure 5). The DUs are connected to the CU via the Coincidence Interface Board CI-1 in each of the DUs and the Multiplexer Board MB-1 in the CU. Determining the number of DUs and DBs per DU needed follows the same principles as described in the Small System section. Each DU should only contain modules that will not be in coincidence with each other, as the default coincidence processing software does not allow coincidences between detectors that originate from the same DU. The DU should then contain the minimum number of DBs necessary to service all the required detector modules.

Each DU services a maximum of eight DBs, and each 32-channel DB services a maximum of 32 analog inputs (note that a conventional PET block detector has four analog outputs, one for each photomultiplier tube). Thus, the Standard System can support a maximum of 2048 analog inputs (32 analog channels per DB, 8 DBs per DU, and 8 DUs per CU), which corresponds to 512 block detector modules.

As an example, consider a cylindrical PET camera, where each detector module has five analog outputs and covers a 5 cm x 5 cm area. There are 44 detector modules per ring (roughly 70 cm diameter) and 5 rings (25 cm axial coverage). The OpenPET system would be configured as follows. The system consists of 220 detector modules. Each 32-channel DB would service 6 detector modules, using 30 analog channels and the remaining two channels on each DB would be unused. As there are a maximum of eight DBs per DU, a DU can service a maximum of 48 of these detector modules. If we divide the 220 modules by 48 modules per DU, we find that the system requires 4.583 DUs. Since DUs are quantized, it really needs 5 DUs, with each DU servicing 44 detector modules. In order to make sure that the correct coincident pairs will be collected, the modules in each DU should be selected so that each DU services a “pie slice” that spans ~72° azimuthally and the full 25 cm axial thickness.

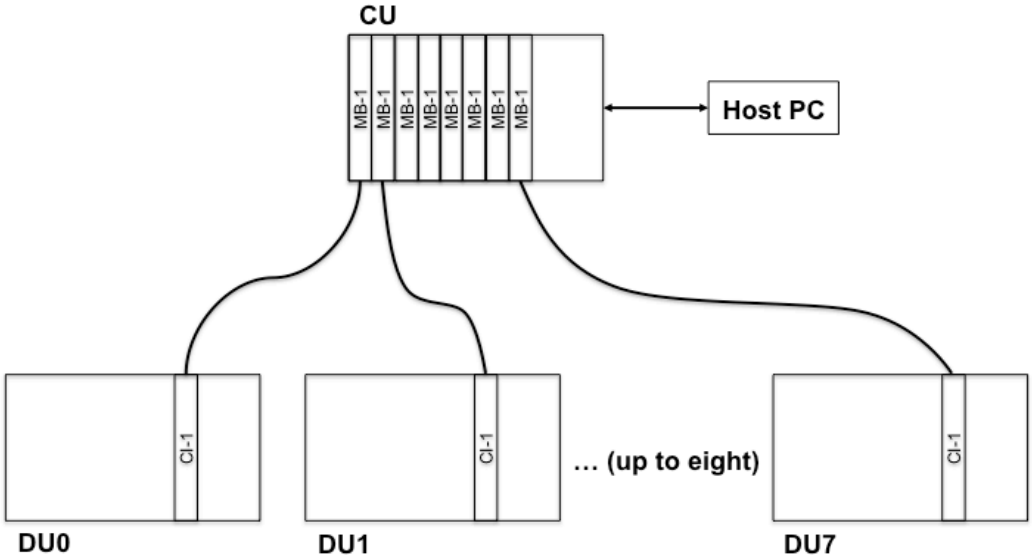


Figure 5: Configuration of a Standard OpenPET System. Each of slots 0-7 in the CU contains a Multiplexer Board-1 that services a single DU.

1.2.3 Large System

If the total number of DBs is between 65 and 512, you must use a Large System. This consists of between 9 and 64 DUs (defined in Section 1.1.2) and a single CU (defined in Section 1.1.3) connected to a Host PC (Figure 6). The DUs are connected to the CU via the CI-8 in each of the DUs and the MB-8 in the CU. Determining the number of DUs and DBs per DU needed follows the same principles as described in the Standard System section. The difference between the Standard and Large Systems is that the Multiplexer Board-8 that plugs into slots 0–7 of the CU contains active circuitries (*i.e.,* FPGA, etc) that are configured as multiplexers. This allows each of the eight slots in the CU to service up to eight DUs (in a Standard System, each CU slot services a single DU). Thus, the Large System can support a maximum of 16,384 analog inputs (32 analog channels per DB, 8 DBs per DU, and 64 DUs per CU), which corresponds to 4,096 block detector modules. Again, the group of DUs processed by one slot in the CU should only contain modules that will not be in coincidence with each other, as the default coincidence processing software does not allow coincidences between detectors serviced by the same CU slot. The DU should then contain the minimum number of DBs necessary to service all the required detector modules.

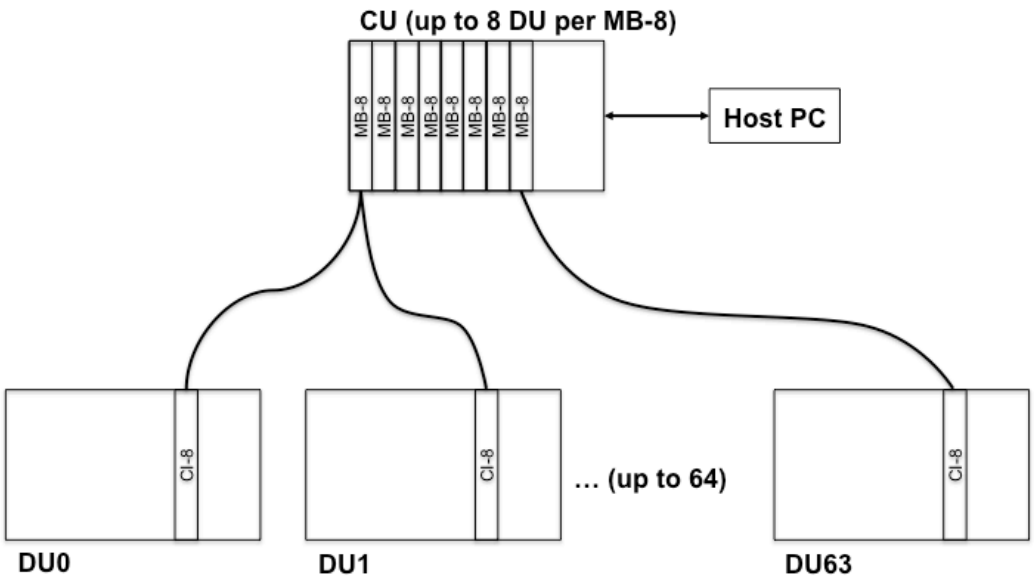


Figure 6: Configuration of a Large OpenPET System. Each of slots 0-7 in the CU contains a Multiplexer Board-8 that operates as a multiplexer, allowing that slot to service between 1 and 8 DUs.

1.3 Timing & Timing Signals

The system level timing signals are shown in Figure 7. There are two timing signals—the System Clock, which is an 80 MHz clock signal, and the Time Slice Boundary, which defines the beginning of a Time Slice. The firmware will support both "short" and "long" event words. In "short" mode, the Time Slice Boundary is generated every eight System Clock cycles, while in "long" mode it is generated every sixteen System Clock cycles. The choice creates a tradeoff—in "short" mode the dead time is a factor of two shorter but the number of bits per event word is also a factor of two smaller.

The general concept is that the system divides time into small, fixed length time slices (100–200 ns or 8–16 clocks). All individual operations must occur within a single Time Slice, which implies that only Single Event Words that occur in the same Time Slice can be combined to form a coincident event. Since it can take significantly longer than a single Time Slice to fully process a single event, the system is pipelined so that the processing is divided into smaller steps that each can be completed in a single Time Slice. During one Time Slice, each of the boards that output Singles Event Words (namely the Detector Boards and Coincidence Interface Boards) can pass four Singles Event Words. Thus, the maximum singles rate seen at the output of the Multiplexer Boards is 32 Singles Event Words (four for each of the eight Multiplexer Boards) per Time Slice, or approximately 320 million Singles Event Words per second. Similarly, the Coincidence Unit can theoretically identify 448 Coincident Events per Time Slice (16 for each of the 28 Detector Unit—Detector Unit combinations), which corresponds to 4.48 billion Coincidence Event Words per second. In practice, the maximum event rate is limited by the transfer rate between the Coincidence Unit and the Host PC, which is considerably slower.

1.3.1 System Clock

The System Clock is an 80 MHz clock. In general, it is generated on the Support Board in the Coincidence Unit (although it can be generated on the Support Board in a Detector Unit such as in the Small System), and then buffered through the rest of the system. Propagation delays will introduce skewing, therefore each FPGA that outputs data will also output a copy of the System Clock that is synchronized with its output data signals. In general, each board in the system regenerates the clock using a phase-locked loop (PLL) in order to maintain signal quality and to minimize phase drift.

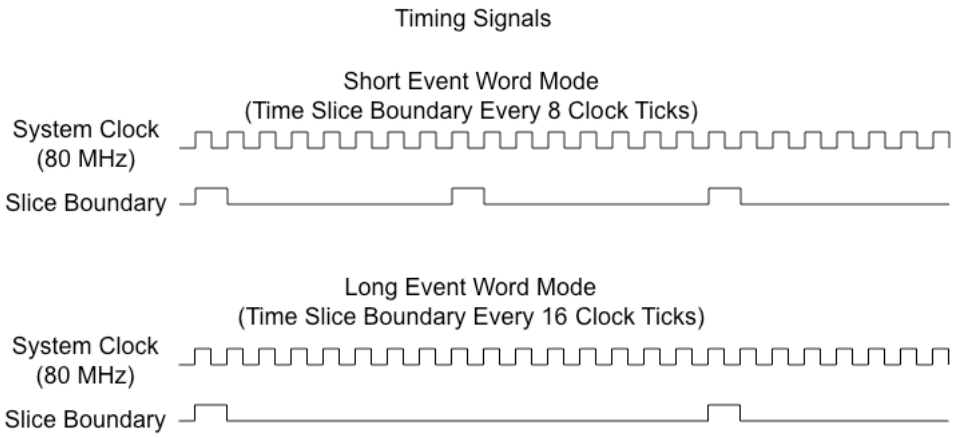


Figure 7: System level timing signals.

1.3.2 Time Slice Boundary

The rising edge of the Time Slice Boundary defines the beginning of a Time Slice. The width of the pulse is one System Clock cycle, and the period is eight System Clock cycles . In general, it is generated on the Support Board in the Coincidence Unit (although it can be generated on the Support Board in a Detector Unit such as in the Small System), and then buffered through the rest of the system. Propagation delays will introduce skewing; therefore each FPGA that outputs data will also output a copy of the Time Slice Boundary that is synchronized with its output data signals.

1.3.3 Time Slice

The system divides time into small, fixed length Time Slices (100–200 ns or 8–16 clocks). All individual data processing operations must occur within a single Time Slice, which implies that only Single Event Words that occur in the same Time Slice can be combined to form a coincident event. While it can take significantly longer than one Time Slice to fully process a single event, the system is pipelined so that the processing is divided into smaller operations that each can be completed in a single Time Slice. It takes one Time Slice to transfer a Singles Event Word.

1.4 Firmware and Software Structures

The OpenPET firmware and software structures are based on a computer network tree topology. The configuration strategy needs to fulfill the following two basic requirements:

(1) Compatibility with different types of detector modules (e.g. single analog channel addressing, single crystal addressing for a conventional block detector, etc.);
(2) Compatibility with different sized systems (e.g., Small, Standard and Large Systems).

In addition, the addressing strategy needs to be implementable, flexible and reliable.

1.4.1 Large System

As shown in Figure 8, the basic characteristics of the OpenPET tree topology for a Large System (Section 1.2.3) are:

(1) The host computer is the central 'root' node (the top level of the hierarchy);
(2) The host computer is connected to a Coincidence Unit Controller (CUC) node that is one level lower in the hierarchy (i.e., the second level) with a point-to-point link.
(3) The CUC node is connected to up to 8 Multiplexer Board (MB) nodes that are one level lower in the hierarchy (i.e., the third level) with point-to-point links.
(4) The MB node is connected to up to 8 Detector Unit Controller (DUC) nodes that are one level lower in the hierarchy (i.e., the fourth level) with point-to-point links.
(5) The DUC node is connected to up to 8 Detector Board (DB) nodes that are one level lower in the hierarchy (i.e., the fifth level) with point-to-point links.
(6) The DB node supports 16 or 32 analog channels which can either be addressed individually or addressed in groups (e.g. 4 channels for one conventional block detector)

The firmware and software structure for a Large System is shown in Figure 9.There are four types of software code and seven types of firmware code run in a complete Large System:

Software:

(1) Software that runs in the Host computer
Function: top level system configuration and calibration, data acquisition and analysis.
(2) Software that runs in the CUC NIOS II μ-processor;
Function: CUC SB board monitoring and management, coincidence pair configuration;
(3) Software that runs in the MB (hardware to be determined);
Function: MB board monitoring and management, multiplexer configuration;
(4) Software that runs in the DUC NIOS II μ-processor;
Function: DUC SB board monitoring and management, DB board configuration, calibration, monitoring and management.

Firmware:

(1) Firmware for CUC main FPGA (Altera Cyclone III EP3C40F780)
Function: CUC SB board monitoring and management, coincidence pair configuration;
(2) Firmware for CUC IO FPGA 1 and 2 (two identical FPGAs, Altera Cyclone III EP3C40F780)
Function: CUC command/status flow and high-speed dataflow router;
(3) Firmware for MB (hardware to be determined)
Function: MB monitoring and management, multiplexer configuration, MB command/status flow;
(4) Firmware for DUC main FPGA (Altera Cyclone III EP3C40F780)
Function: DUC SB board monitoring and management, DB board configuration, calibration, monitor and management.
(5) Firmware for DUC IO FPGA 1 and 2 (two identical FPGAs, Altera Cyclone III EP3C40F780)
Function: DUC command/status flow and high-speed dataflow router;
(6) Firmware for DB FPGA
Function: ADC control, energy calculation and correction, crystal decoding, energy threshold, TDC, time correction and etc;

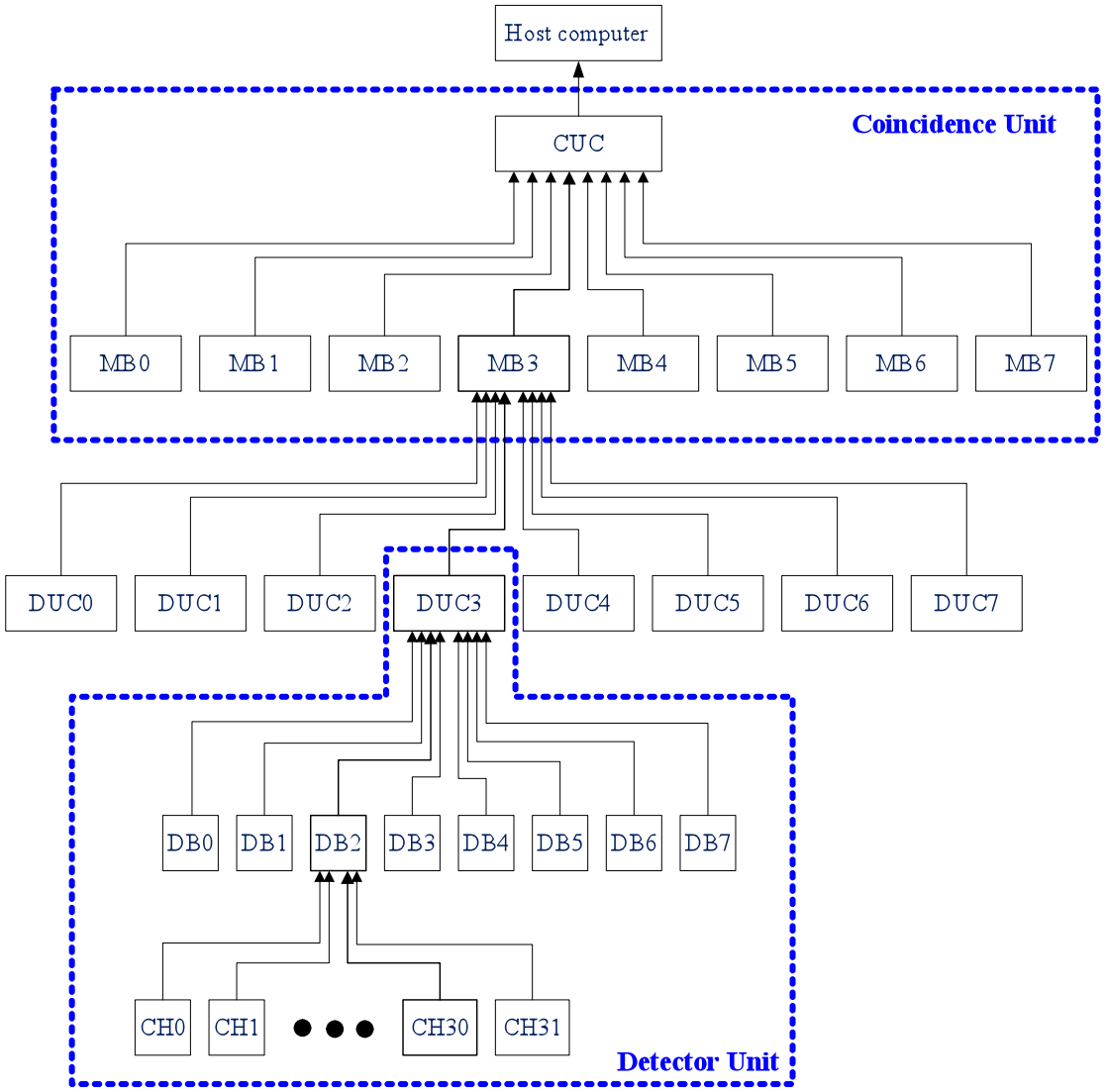


Figure 8: Tree network topology of an OpenPET Large System. A Coincidence Unit Controller (CUC) that supports 8 Multiplexer Boards is housed in a crate with a power supply. The assembled unit is called a Coincidence Unit. A Detector Unit Controller (DUC) that supports 8 Detector Boards is also housed in a crate with a power supply. The assembled unit is called a Detector Unit. One Coincidence Unit fans out to up to 64 Detector Units. The hardware of the CUC and DUC are the same (Support Board). The differences are the firmware and software configurations.

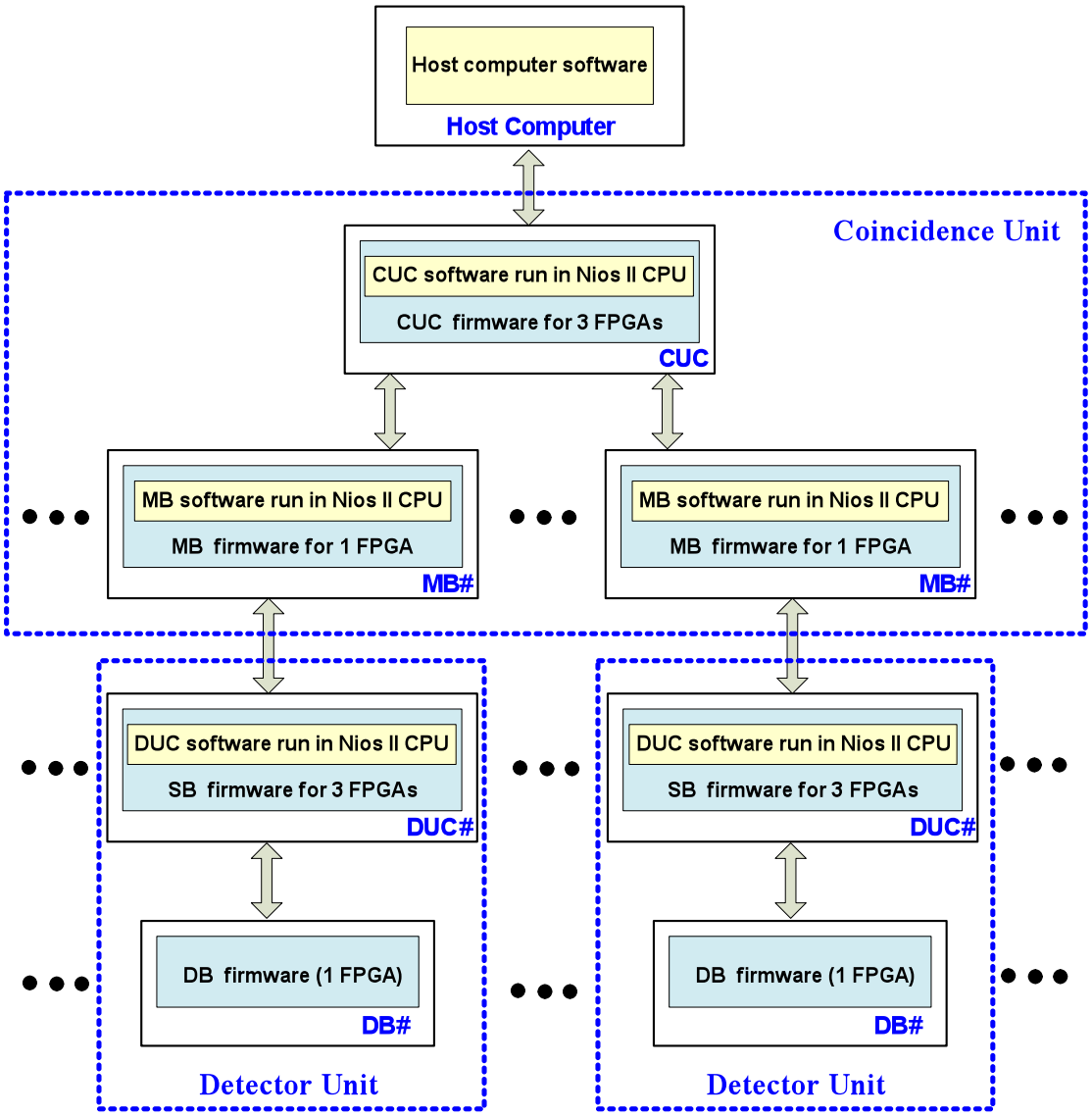


Figure 9: Firmware and software structure for a Large System.

1.4.2 Small System

As described earlier (Section 1.2.1), the OpenPET system can also be configured as a Small System. In a Small System, a support board is configured as a Coincidence Detector Unit Controller (CDUC), which interfaces with the detector boards and performs coincidence functions. Basically the CDUC performs the functions of both the CUC and DUC. The initial firmware and software for the first release has been developed for a Small System. The configuration for a Small System is shown in Figure 10.

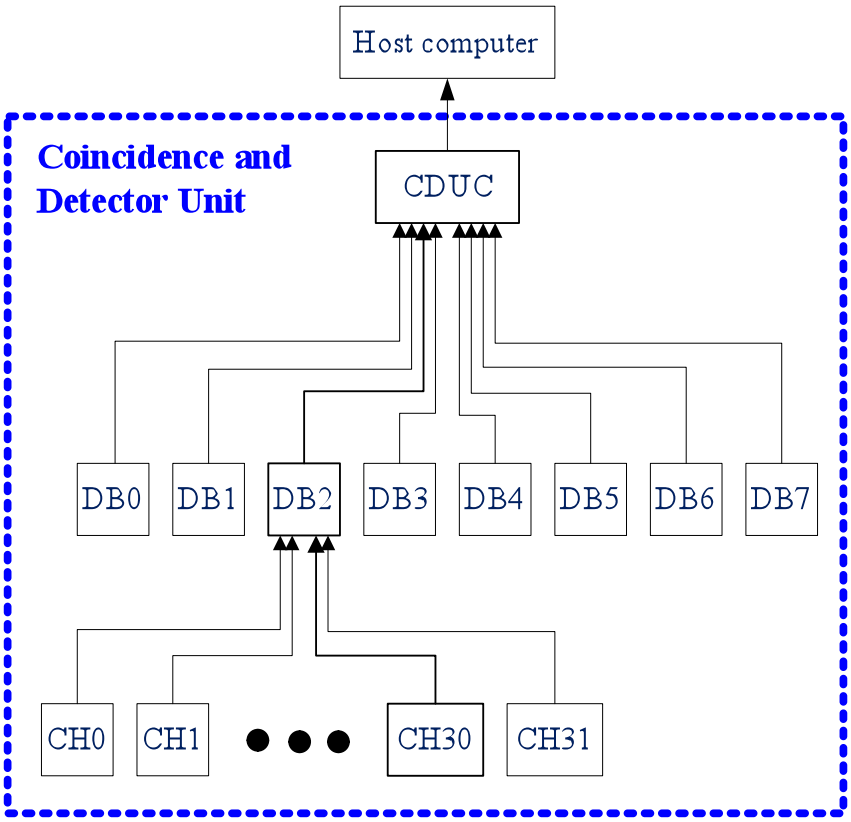


Figure 10: The tree network topology for a Small System. A CDUC performs the functions of both the CUC and DUC.

The firmware and software structure for a Small System is similar to that shown in Figure 9. There are two types of software code and three types of firmware code run in a complete Small System:

Software:

(1) Software that runs in the Host computer (the same as the in a Large system)
Function: top level system configuration and calibration, data acquisition and analysis.

(2) Software that runs in the CDUC NIOS II μ-processor;
Function: CDUC SB board monitoring and management, coincidence pair configuration, DB board configuration, calibration, monitoring and management;

Firmware:

(1) Firmware for CDUC main FPGA (Altera Cyclone III EP3C40F780)
Function: CDUC SB board monitoring and management, coincidence pair configuration, DB board configuration, calibration, monitoring and management.

(2) Firmware for CDUC IO FPGA 1 and 2 (two identical FPGAs, Altera Cyclone III EP3C40F780)
Function: CDUC command/status flow and high-speed dataflow router;

(3) Firmware for DB FPGA
Function: ADC control, energy calculation and correction, crystal decoding, energy threshold, TDC, time correction and etc;

2 Getting Started – *Small System Oscilloscope Mode*

2.1 Getting the Hardware

The OpenPET system uses a custom Support Crate that is similar to a VME crate (see Section 1.1.1). Support Crates should be purchased through Elma: Support Crate Chassis, part number 12V12XXX78N2VCGX-LBL, http://www.elma.com/en/us/.

The OpenPET PC boards can be purchased from Terasic through their OpenPET website at http://www.openpet.terasic.com. For a small system in the first release, each Detector Unit (see Section 1.1.2) requires the following OpenPET boards:1 Support Board

- 1-8 16-Channel Detector Boards

You also need the following additional components:

- 1 host PC with a Windows 7 operating system.
- 1 QuickUSB module: Bitwise Systems, part number QUSB2, http://www.bitwisesys.com/qusb2-p/qusb2.htm.
- 1 standard USB cable
- 1 USB-Blaster Cable: Terasic, Digi-Key part number P0302-ND, http://www.digikey.com/product-highlights/us/en/terasic-usb-blaster-cable/3718.

The QuickUSB module is a small PC board that contains circuitry to provide high-speed USB 2.0 capability. It is plugged into either the Host PC Interface Board or the Support Board. The USB-Blaster cable interfaces between a USB port on the host PC to the Altera main FPGA on the Support Board, so configuration data can be sent from the PC to the FPGAs. More detailed instructions are provided in the following sections.

2.2 Assembling the Hardware

Once you receive the parts for an OpenPET support crate, some minor assembly is required. Figure 11 shows the two main components: the empty Support Crate and the Support Board. In addition, you will receive a small custom jumper board (see Figure 12c). You will also need a standard power cable for the crate, M2.5 x 12 mm Phillips head screws, M4 x 5 mm Phillips head screws, and appropriate screw drivers.

The first assembly step is to attach the Support Board to the crate. From the back side of the crate, align the screw holes on the Support Board to those on the Support Crate and secure it using M2.5 x 12 mm Phillips head screws. We recommend using at least 3 screws on the top, middle and bottom rows. It is also useful to place a piece of paper across the fans during assembly so any dropped screws don't fall into them. When the board is properly aligned, there should be about a 4 mm gap between the right edge of the Support Board and the right side of the Support Crate (Figure 12a).

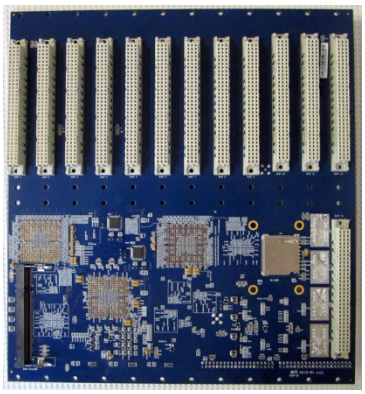
(a) (b) 

Figure 11: (a) Empty Support Crate viewed from the front. (b) Support Board viewed from the front.

Once the Support Board is secured on the crate, you need to attach the power cables. The power cables and Support Board connectors are labeled. From bottom to top, the power cables are -5 V (orange cable), ground (black cables), +3.3 V (purple cables) and +5 V (red cable), as shown in Figure 12b. Secure these power cables on to their respective Support Board connectors using M4 x 5 mm Phillips head screws. You may need to feed each screw through the power cable lug nut before attaching it to the Support Board, since the lug nuts fit tightly. In addition, there is a small bundle of cables that can be used for monitoring but these cables are not needed.

Finally you need to plug the custom small jumper board into the back of the Support Board. It plugs in to the connector just to the right of the Jtag connector (see Figure 12c). Specifically, board plugs into the left column of pins on the connector (which is marked main, 58, 62, 65, 68). This jumper board identifies that the Jtag should communicate with the main FPGA on the Support Board.

The fully assembled Support Crate Is shown in Figure 13.

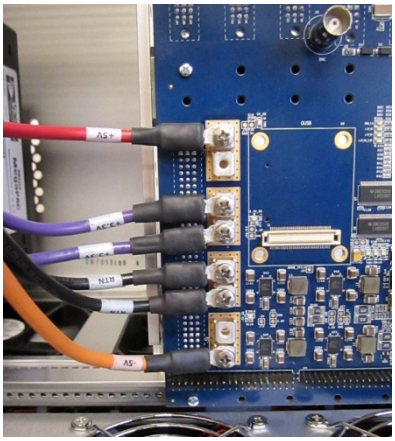
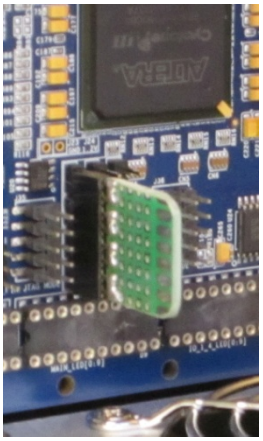
(a) (b)  (c) 

Figure 12: (a) Close up of the small gap (~4 mm) between the Support Crate and the edge of the Support Board when assembled properly. (b) Close up of the power cables attached to the back of the Support Board. (c) Close up of the jumper board plugged into the back of the Support Board.

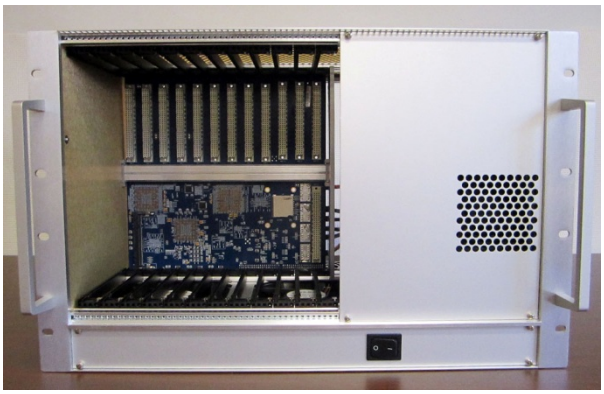
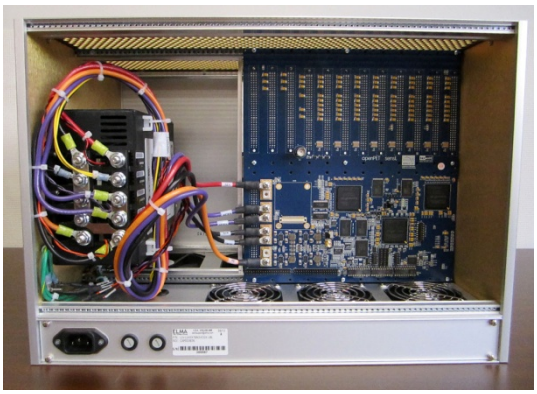
(a)  (b) 

Figure 13: The assembled OpenPET Support Crate viewed from the (a) front and (b) back.

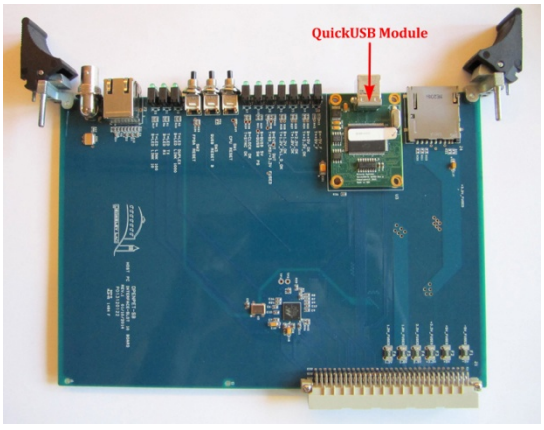
2.3 Downloading the Software and Firmware

2.3.1 Installing QuickUSB

The OpenPET system requires the use of a QuickUSB module to provide high-speed USB 2.0 capability. You can purchase this online through Bitwise systems at http://www.bitwisesys.com/qusb2-p/qusb2.htm (product code QUSB2). In addition to the hardware module, this includes the QuickUSB library with a driver, interface DLL, and example programs for Linux, MacOSX and Windows. In addition, you will need a USB cable. However, you do not need to buy the QuickUSB Adapter Board as Bitwise implies.

First, you need to mount the QuickUSB module onto either the Host PC Interface Board in the U3 connector or onto the Support Board in the U6 connector (Figure 14). The system is designed to work with the QuickUSB module mounted in either (but not both) of these locations.

Note: During the first release, the Host PC Interface Board is not supported yet. So you need to mount the QuickUSB module onto the Support Board (Figure 14**b).**

(a) 

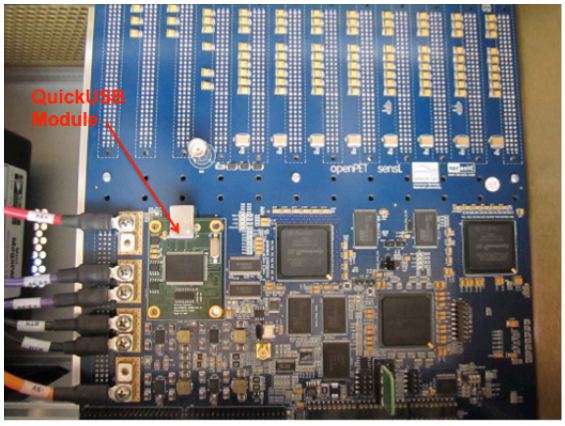
(b) 

Figure 14: QuickUSB module mounted onto the (a) Host PC Interface Board or (b) Support Board.

Next, you need to install the QuickUSB driver onto your host PC. Browse and select the setup executable for the QuickUSB software (e.g., Desktop → QuickUSB_Installation → Windows → setup.exe). This will launch the InstallShield Wizard in a new window (Figure 15a). Step through the instructions by clicking the Next button at the

bottom, agreeing to the software license agreement terms and defining the folder where the QuickUSB software should be installed (Figure 15b). When the wizard is ready to begin installation of the library, click Install (Figure 15c). The installation takes a few minutes and a status bar shows the progress. An additional window will then pop up that walks you through the installation of the QuickUSB device drivers (Figure 15d). The InstallShield Wizard will indicate when the full installation is completed.

If you have further questions or problems with the installation, please refer to the Bitwise QuickUSB User Guide for details (e.g., http://www.bitwisesys.com/v/public/media/QuickUSB_User_Guide_v2.15.2.pdf).

Once you have installed the QuickUSB module, you need to program it with the correct firmware using the QuickUSB Programmer (see Figure 16). After you have successfully installed the correct firmware, you can confirm your installation by running the QuickUsb Diagnostics (see Figure Figure 17). The OpenPET system requires the firmware model "QuickUSB QUSB2 Module v2.15.1 (Block Handshake)" to configure the QuickUSB module for the correct data transfer mode. For further details on this firmware model and configuration, please refer to the QuickUSB User Guide at http://www.bitwisesys.com/v/public/media/quickusb_user_guide.pdf.

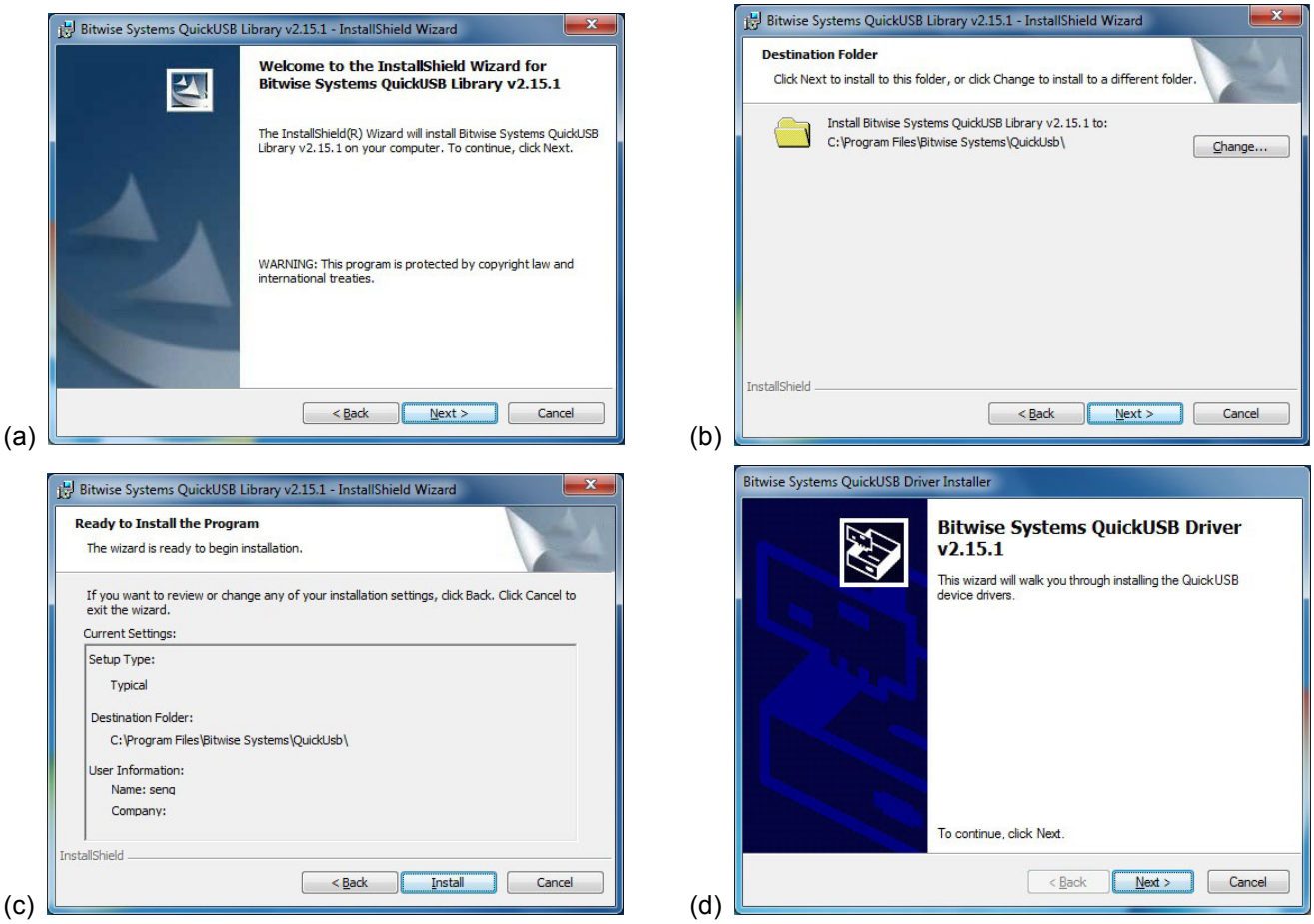


Figure 15: Bitwise Systems QuickUSB Library v2.15.1 InstallShield Wizard: a) welcome window, (b) destination folder window, (c) installing the library window, and (d) installing the driver window.

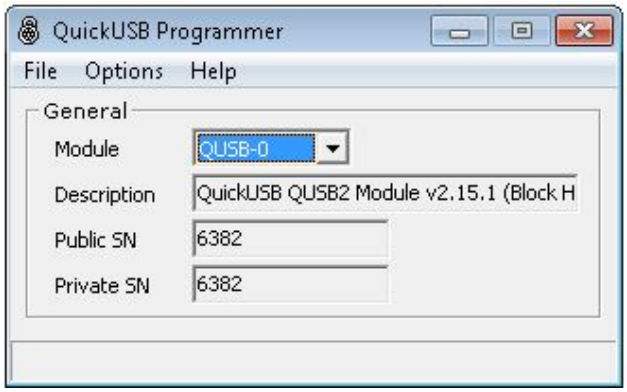


Figure 16: QuickUSB Programmer used to program the QuickUSB module with the correct firmware: QuickUSB QUSB2 Module v2.15.1 (Block Handshake).

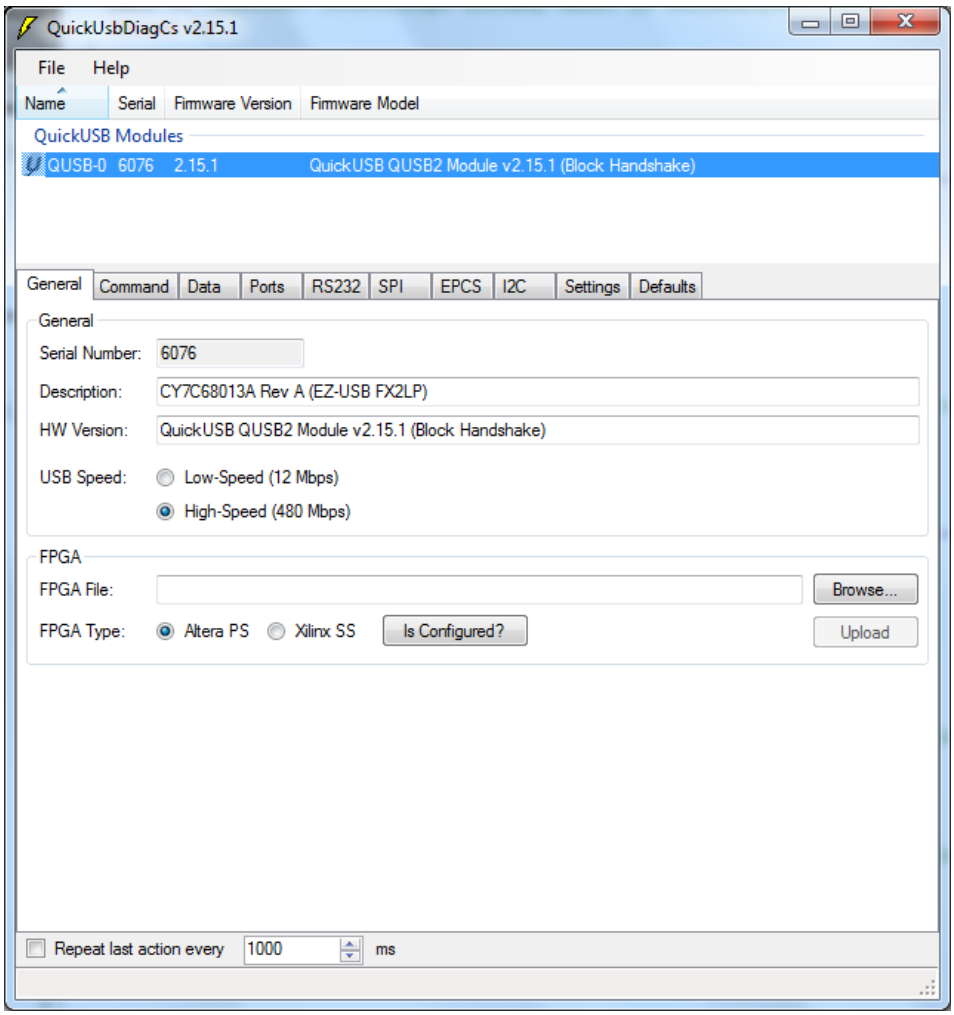


Figure 17: QuickUsb Diagnostics used to confirm the QuickUSB module with the correct firmware: QuickUSB QUSB2 Module v2.15.1 (Block Handshake).

2.3.2 Installing Altera Tools

Even if you only plan to use the off the shelf OpenPET software and firmware, the OpenPET system requires the use of Altera design tools in order to load the appropriate firmware into the Support Board. These tools can be downloaded for free from the Altera Download Center website (e.g., https://www.altera.com/download/sw/dnl-sw-index.jsp, Figure 18a). From the Download Center, select to download the Quartus II Web Edition that is free and doesn't require a license.

On the Quartus II Web Edition download page Figure 18b, select release: **13.1** select the Windows operating system and the Akamai DLM Download Manager. Under the Individual Files tab, select the Quartus II Software and the Cyclone III, Cyclone IV device support. You can select additional options (such as the ModelSim-Altera Edition simulation tools), but they are not required to run OpenPET and they will lengthen the download time.

At this time, only Altera Quartus **13.1** is supported.

You are then required to log in with your username and password if you already have a myAltera account. If not, then create an account using your email address and complete the account registration information. Once your myAltera account has been created, you will be directed to your myAltera Home page. Select the Download Center link on the top right side of the page.

Once you have returned to the Download Center (Figure 18a), select the Quartus II Web Edition and specify the Windows operating system, DLM Download Manager, Quartus II Software, and the Cyclone III, Cyclone IV device support (if they aren't already specified). A separate Akamai NetSession Interface window will then pop up (Figure 18c) in which you should click Download on the installer and then Run to proceed with the software installation. You will have to agree to the End User License terms. A popup warning message will then appear in which you have to confirm that you want to open the executable file (e.g., QuartusSetupWeb-13.10.163.exe) from the Internet. You will then have to confirm that you want to allow this program to make changes to your computer (Figure 18d).

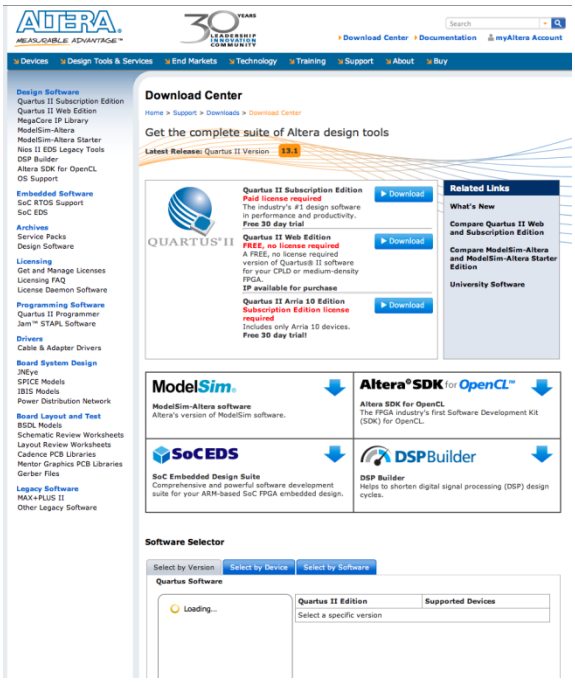
Finally the Quartus II Web Edition Setup Wizard window will appear and step you through the installation (Figure 18e). You will have to agree to a License Agreement again as well as specify the installation directory (e.g., C:\altera\13.1). Then you should make sure that the components selected for installation are correct (e.g., Quartus II Software and Cyclone III/IV) and that you have the available disk space specified in the Summary. Once you start the installation, its progress will be shown by a status bar. Once the Setup has finished installing the Altera tools, you can create shortcuts on your Desktop and launch Quartus II (Figure 18f).

If you have further questions or problems with the installation, please refer to the Altera support website (http://www.altera.com/support/spt-index.html) for user information. For specific instructions on how to use these Altera tools with the OpenPET system, see Section 2.3.4.

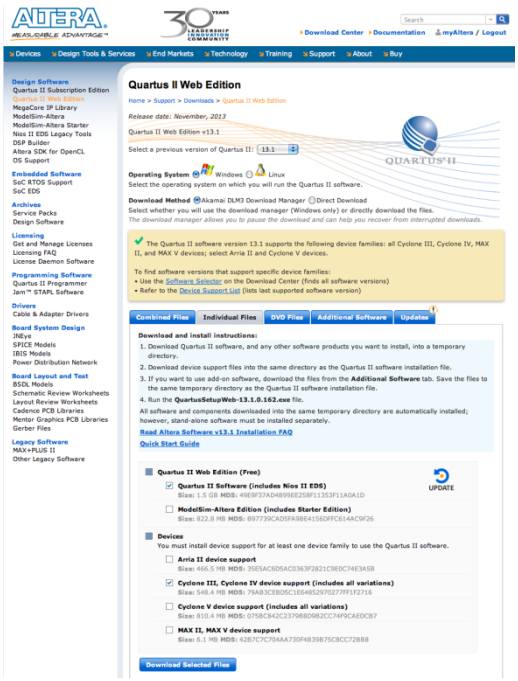
2.3.3 Installing USB-Blaster Driver

The USB-Blaster cable interfaces between a USB port on your host PC and the Altera main FPGA on the Support Board. It allows configuration data to be sent from the host PC to the FPGAs.

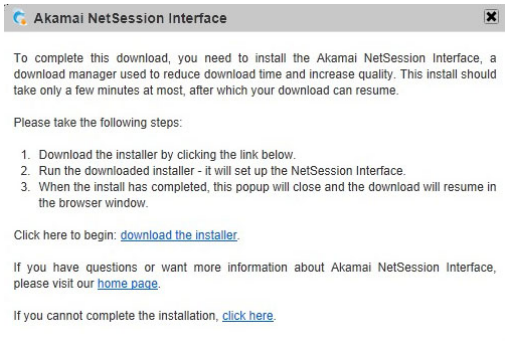
You must install the Altera USB-Blaster or USB-Blaster II driver before you can use the program devices with the Quartus II software. These drivers are automatically copied to the drivers folder within the Altera folder during the installation described in the previous section. However, it needs to be installed at first use. You will be prompted to install the driver the first time the USB-Blaster cable is plugged in. Whether you need to install the USB-Blaster or USB-Blaster II driver depends on your cable. Please see the step-by-step Altera instructions for this driver installation at http://www.altera.com/download/drivers/usb-blaster/dri-usb-blaster-vista.html.



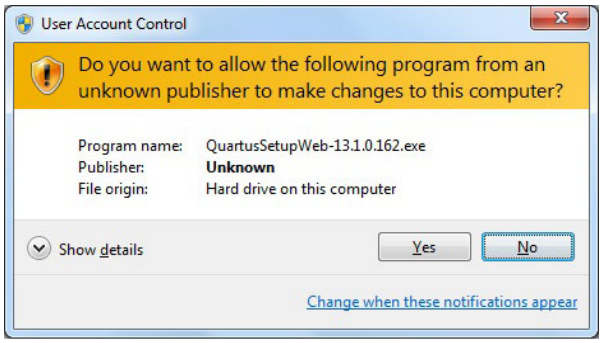
(a)



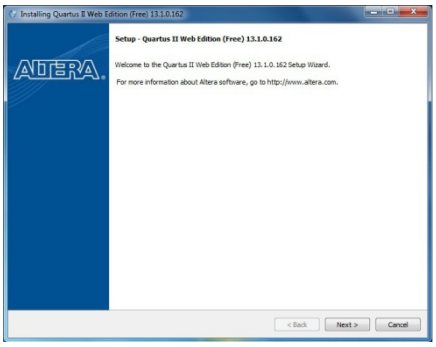
(b)



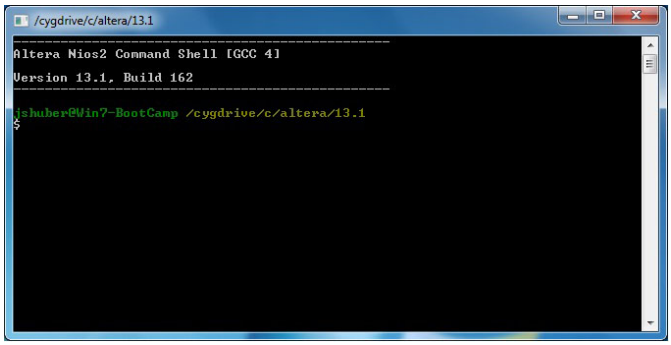
(c)



(d)



(e)



(f)

Figure 18: Altera Tools installation windows: (a) download center window, (b) Quartus II Web Edition window, (c) Akamai NetSession Interface window, (d) User Account Control confirmation window, (e) Quartus II Web Edition Setup Wizard window and (f) Altera Nios2 Command Shell window.

2.3.4 Installing OpenPET Firmware & Software

Download onto your host PC the latest version of the OpenPET firmware and software "Binary" zip from the OpenPET website at http://openpet.lbl.gov/downloads/firmware-software/. You will need to register as an OpenPET user and login before you can access this page. Once you unzip this file (e.g., OpenPET_v1.0.zip), you should see the ***OpenPET_ROOT*** directory, as shown in Figure 19.

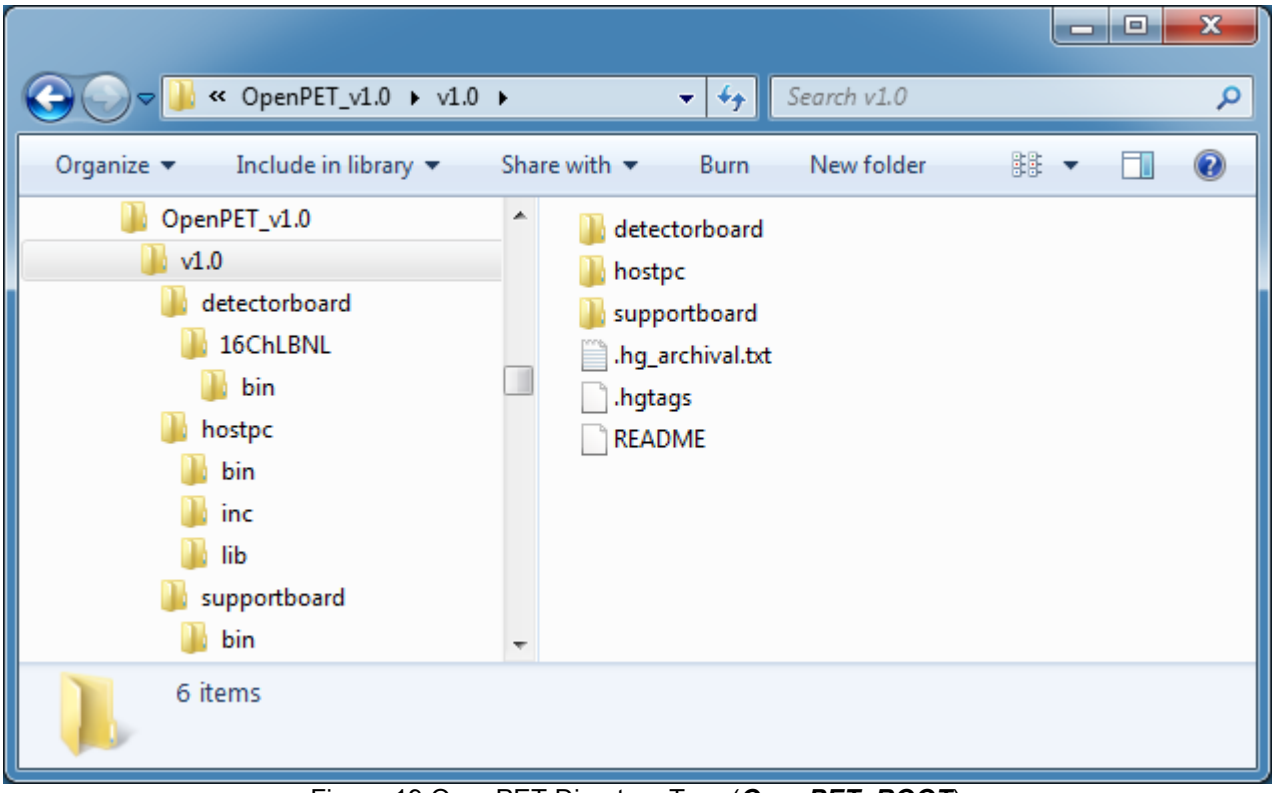


Figure 19 OpenPET Directory Tree (***OpenPET_ROOT***)

The directory '***OpenPET_ROOT****/supportboard*' contains the Support Board's firmware, as well as an additional script to program the board's flash.

The directory '***OpenPET_ROOT****/detectorboard*' contains the LBNL 16-channel Detector Board's firmware and a script to program the flash. Future releases will also include other Detector Boards.

The directory '***OpenPET_ROOT****/hostpc*' contains three subdirectories: inc, lib and bin. The 'inc' and 'lib' subdirectories contain the header and library files for the OpenPET QuickUSB interface. The 'bin' subdirectory contains the two executables needed to interface with the QuickUSB modules. Please add this directory to your MS Windows PATH environment variable. (You will have to do this every time a new OpenPET binary package is released.) Open a Windows Control Panel and type 'env' in the search box in the upper right corner as shown in Figure 20, then click on 'Edit environment variables for your account'. When the new popup window appears, go to user variables, double click on the variable 'PATH', and append the full path to variable value (see Figure 21).

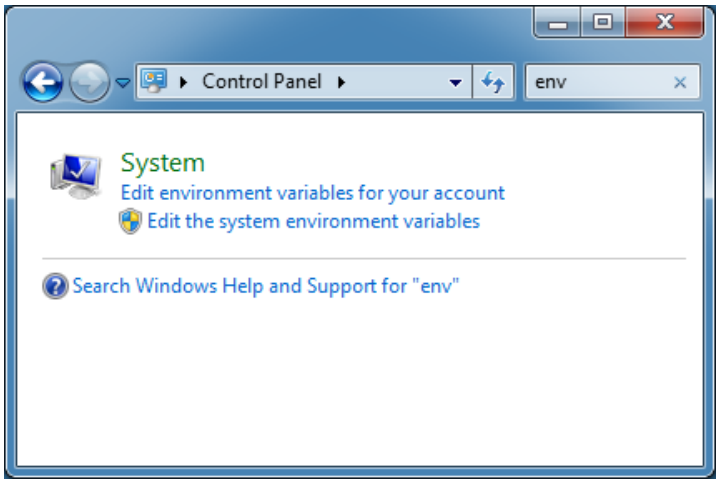


Figure 20 Windows Environment Variable through control panel

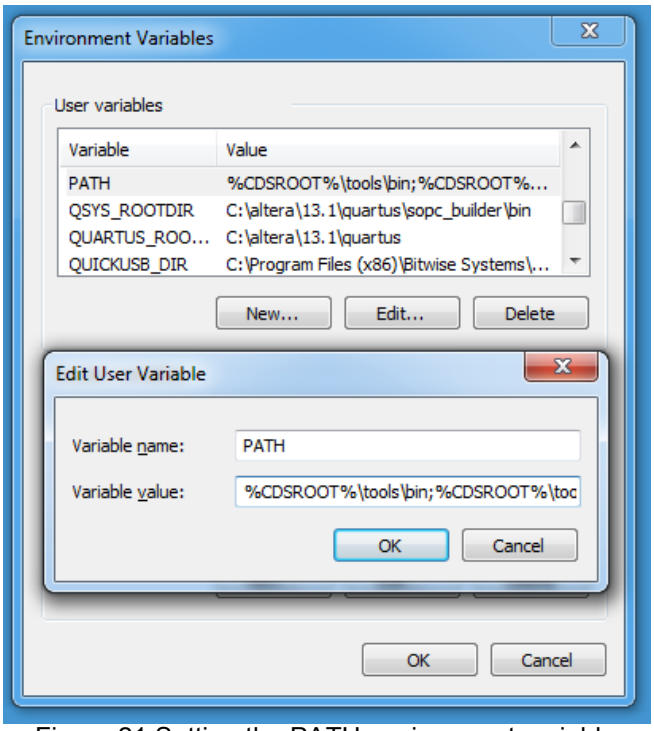


Figure 21 Setting the PATH environment variable

2.3.4.1 Programming OpenPET flash Images

Before we program the flash images on the Support Board, we have to setup some jumpers. First make sure the Support Crate is powered off and disconnected. Place jumpers on each Detector Board, connecting pins 1 and 2 on J1, J2 and J3, so you can download firmware to the Detector Board (Figure 22 (a)) using the instructions below. (Note: code can also be downloaded via the JTAG connector, which is still enabled with theses jumpers in place.)

Connect the USB-Blaster hardware from the back of the Support Board to your host PC (USB port), as shown in Figure 22 (b); pin 1 (marked as red) should face down when connecting to the JTAG connector.

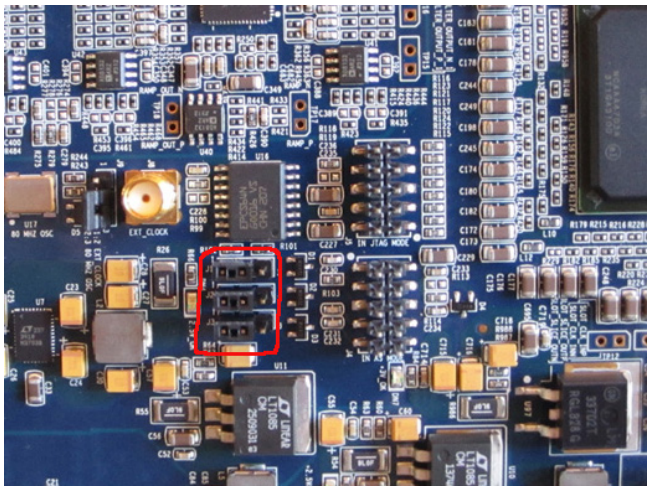
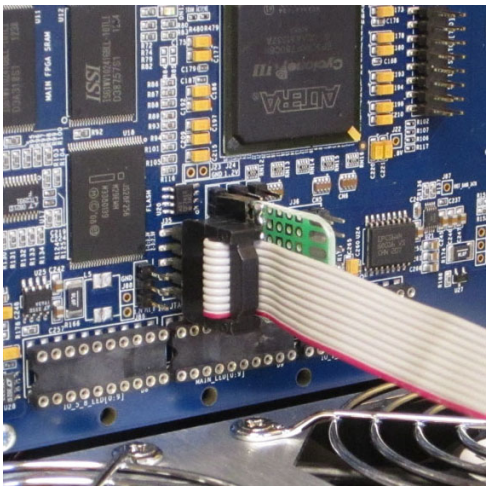
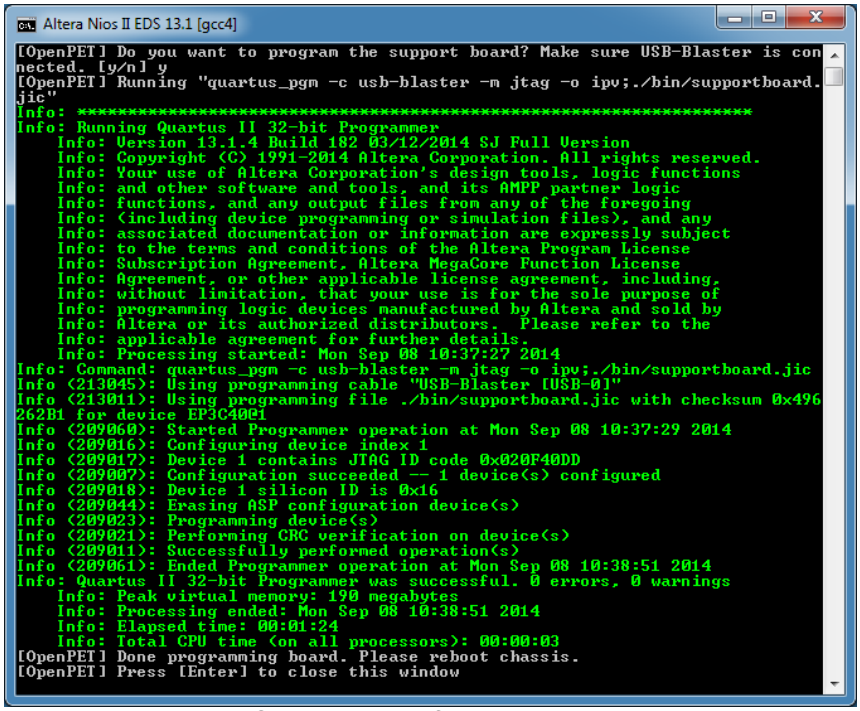
(a)  (b) 

Figure 22: (a) Detector Board with jumpers installed on pins 1 and 2 for J1, J2 and J3. (b) USB-Blaster plugged into the JTAG connector on the back of the Support Board. A custom jumper board is also shown.

After plugging in the cable, connect and turn the power on to your Support Crate. Now you are ready to download the OpenPET software and firmware to your OpenPET hardware in two steps:

- In the first step, you download the OpenPET software and firmware to the 3 FPGAs on the Support Board. Go to ***OpenPET_ROOT****/supportboard* and launch **flashboard.bat** (Type = Windows Batch file). Please follow the prompts on the command line window. The result is shown in Figure 23. Once this is done, **it is important that you reboot the crate by turning it off then on again**.



```
Altera Nios II EDS 13.1 [gcc4]
[OpenPET] Do you want to program the support board? Make sure USB-Blaster is con
nected. [y/n] y
[OpenPET] Running "quartus_pgm -c usb-blaster -m jtag -o ipv;./bin/supportboard.
jic"
Info: ***********************************************************************
Info: Running Quartus II 32-bit Programmer
    Info: Version 13.1.4 Build 182 03/12/2014 SJ Full Version
    Info: Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
    Info: Your use of Altera Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Altera Program License
    Info: Subscription Agreement, Altera MegaCore Function License
    Info: Agreement, or other applicable license agreement, including,
    Info: without limitation, that your use is for the sole purpose of
    Info: programming logic devices manufactured by Altera and sold by
    Info: Altera or its authorized distributors.  Please refer to the
    Info: applicable agreement for further details.
    Info: Processing started: Mon Sep 08 10:37:27 2014
Info: Command: quartus_pgm -c usb-blaster -m jtag -o ipv;./bin/supportboard.jic
Info (213045): Using programming cable "USB-Blaster [USB-0]"
Info (213011): Using programming file ./bin/supportboard.jic with checksum 0x496
262B1 for device EP3C40Q1
Info (209060): Started Programmer operation at Mon Sep 08 10:37:29 2014
Info (209016): Configuring device index 1
Info (209017): Device 1 contains JTAG ID code 0x020F40DD
Info (209007): Configuration succeeded -- 1 device(s) configured
Info (209018): Device 1 silicon ID is 0x16
Info (209044): Erasing ASP configuration device(s)
Info (209023): Programming device(s)
Info (209021): Performing CRC verification on device(s)
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Mon Sep 08 10:38:51 2014
Info: Quartus II 32-bit Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 190 megabytes
    Info: Processing ended: Mon Sep 08 10:38:51 2014
    Info: Elapsed time: 00:01:24
    Info: Total CPU time (on all processors): 00:00:03
[OpenPET] Done programming board. Please reboot chassis.
[OpenPET] Press [Enter] to close this window
```

Figure 23 Support Board flash programming script

- In the second step, you download the OpenPET firmware to the Detector Board FPGA. Go to ***OpenPET_ROOT**/16ChLBNL/detectorboard* and launch **flashboard.bat** (Type = Windows Batch file). Please follow the prompts on the command line window. The result is shown in Figure 24. **When done, it is important that you reboot the crate by turning it off then on again**.

Caution: You cannot program the detector board flash before programming the Support Board.

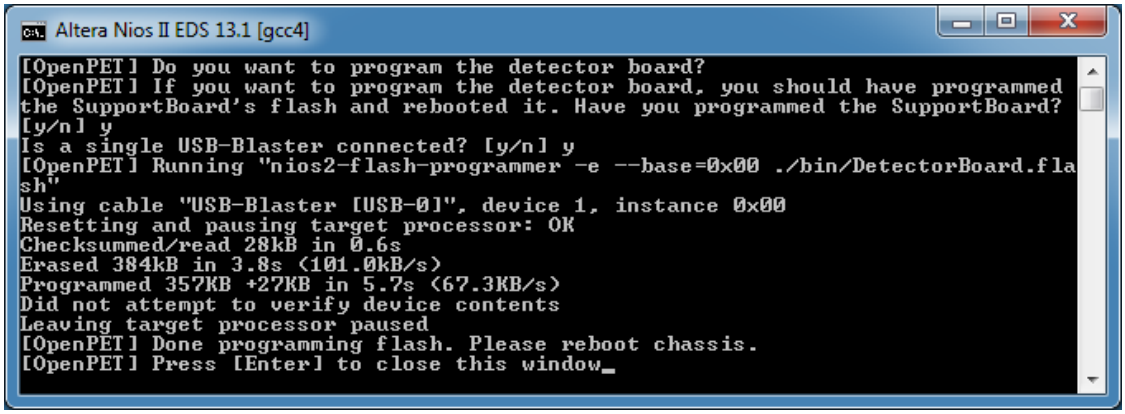


Figure 24 Detector Board flash programming script

2.4 Running a Small System in Oscilloscope Mode

There are two basic executables that can be used to setup and run an OpenPET Small System using USB to communicate with the host PC:

- **opet_cmd_usb.exe** command ID, target address, command payload, timeout, USB module number
- **opet_acq_usb.exe** DAQ time, filename, USB module number.

Prior to running the system with these executables, you need to have the Small System hardware configured, powered on and loaded with the correct OpenPET firmware and software (see Sections 2.2 and 2.3). You also need to plug in a QuickUSB cable to connect the host PC (USB port) with either the Host PC Interface Board (plug into USB port on the front panel) or Support Board (plug into QuickUSB module directly on SB in back of crate). These executables are issued from a standard terminal window from your working directory folder.

2.4.1 Commands

Every OpenPET system command also has an associated response. Both commands and responses (C/R) have the format shown in Figure 25. There are four segments of information required: command/response id (16 bits), source address (16 bits), target address (16 bits), and command/response payload (32 bits).

The command/response id specifies the function of the command, using a 16-bit number with the most significant bit (C/R flag) as 0 for a command and 1 for a response. The source address is a 16-bit number that defines where the command/response originates. Commands originate in the host PC, which has source address 0xF000. The target address is a 16-bit number that identifies where the command/response should be received and processed. Typically commands for a small system have a target address of 0x4000 for the CDUC. The C/R data payload is a 32-bit number that specifies additional information for each command. See Section 5 for further details.

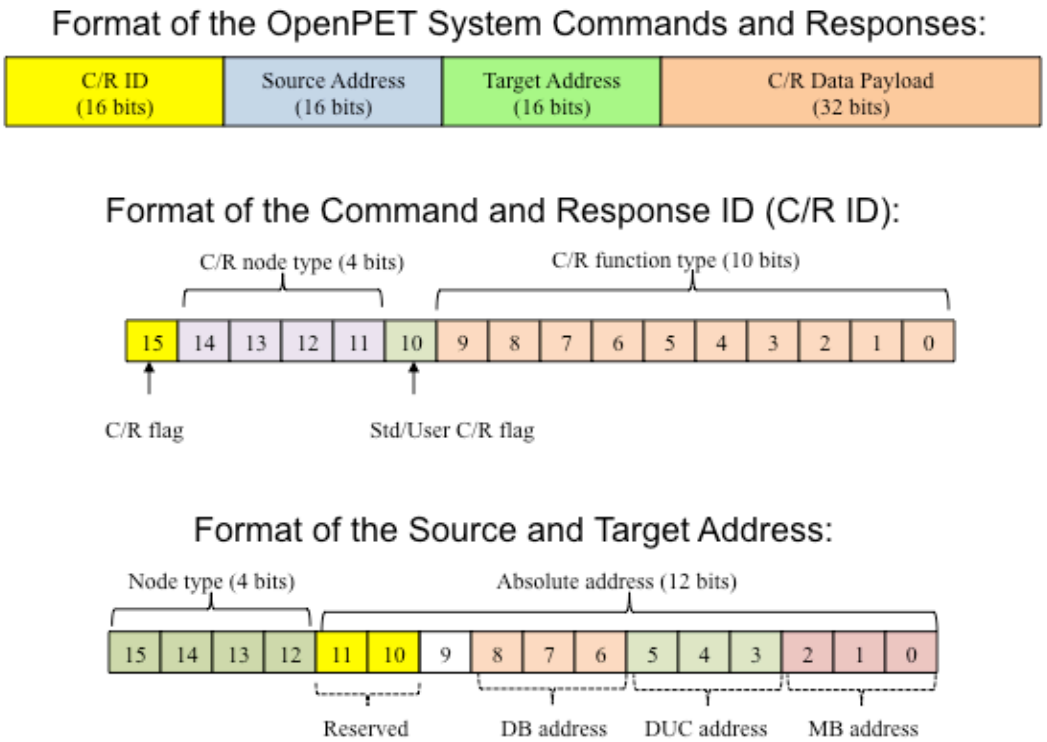


Figure 25: OpenPET address formats. Bits shown in white are not used.

The executable opet_cmd_usb.exe is used to control the system. In particular, it is used to send a command to target hardware that provides a response. The command executable has several arguments:

opet_cmd_usb command ID, target address, command payload, timeout, USB module number

The opet_cmd executable arguments are summarized below:

- **Command ID:** 16 bits
  - (See Table 1)
- **Target address:** 16 bits
  - Coincidence Detector Unit Controller: 0x4000
- **Command payload:** 32 bits
  - (See examples in Section 5)
- **Timeout:** units of milliseconds, default 200, integer range -30,000 to 30,000
- **USB module number:** default 0, integer range 0-7

Tables 1 and 2 show a list of the current OpenPET commands, including their command id and a brief description of their function. See Section 5 for more details, including many command examples.

Command ID (Response ID)	Command Name	Command Function
0x2200 (0xa200)	Boot-up Detector Boards	Loads the FPGA configuration to all Detector Boards.
0x2201 (0xa201)	Set system data mode	Configures the system data mode. At present, only Oscilloscope data mode is implemented.
0x2202 (0xa202)	Read system data mode	Reads the system data mode.
0x2203 (0xa203)	Set Oscilloscope data settings	Configures the Oscilloscope mode data settings. Can specify data format and number of ADC samples per channel.
0x2204 (0xa204)	Read Oscilloscope data settings	Reads the Oscilloscope mode data settings.
0x2205 (0xa205)	Set trigger mask (channels 0-15)	Configures the trigger mask. Can enable/disable trigger for channels 0-15 for a single DB or all DBs.
0x2206 (0xa206)	Set trigger mask (channels 16-31)	Configures the trigger mask. Can enable/disable trigger for channels 16-31 for a single DB or all DBs.
0x2207 (0xa207)	Read trigger mask (channels 0-15)	Reads the trigger mask for channels 0-15 for a single DB.
0x2208 (0xa208)	Read trigger mask (channels 16-31)	Reads the trigger mask for channels 16-31 for a single DB.
0x2209 (0xa209)	Clear event FIFO	Clears the event FIFO to remove data from previous events.

Table 1: Summary of the OpenPET commands for universal tasks.

Command ID (Response ID)	Command Name	Command Function
0x2400 (0xa400)	Send sawtooth test pulse	Sends a sawtooth test pulse signal to all DACs on a single Detector Board. Can specify which DB and the number of sawtooth waves.
0x2401 (0xa401)	Initialize DAC	Initializes DACs registers with default values. Can specify a single DB or all DBs.
0x2402 (0xa402)	Set DAC voltage	Sends a single voltage signal to DACs on a single Detector Board. Can specify which DB, DAC type, DAC channels and voltage value.
0x2403 (0xa403)	Initialize ADC	Initializes ADCs registers with default values. Can specify a single DB or all DBs.
0x2404 (0xa404)	Set ADC gain	Set ADC gain in a single Detector Board. Can specify gain, which DB, and a single channel or all channels.

Table 2: Summary of the OpenPET commands for DB/hardware specific tasks.

Instead of using this command executable, users can use an optional control and analysis tools package based on the ROOT framework, called OpenPET Control and Analysis Tools (OpenPET CAT). See Section 6 for further details on how to install and use OpenPET CAT.

2.4.2 Data Acquisition

The opet_acq_usb.exe executable is used to acquire data, after the system is correctly initialized. The acquisition executable has a few arguments:

opet_acq_usb DAQ time, filename, USB module number.

The specifications of these arguments are outlined below:

- DAQ time: acquisition time in seconds.
- Filename: filename for acquired data.
- USB module number: default=0, integer range 0-7.

An example acquisition executable is opet_acq 30, File1.dat, 0 that collects data from the CDUC through the QuickUSB block 0 for 30 seconds and saves the data to file File1.dat.

Instead of using this data acquisition executable, users can use an optional control and analysis tools package based on the ROOT framework, called OpenPET Control and Analysis Tools (OpenPET CAT). See Section 6 for further details on how to install and use OpenPET CAT.

2.4.3 Example System Setup & Data Acquisition

The sections above describe the two OpenPET executables needed to setup the system and acquire data. However, the use of these executables and their commands are order sensitive. So this section outlines an example of how to use the executables to initialize the system and take test data, in order to help determine whether your system is working properly. More information on diagnostic testing is also available in Section 9.2.

- We list here an example string of commands – see Section 5 for more details. All of the commands below use a USB module number of 0. The recommended response time out for each command is listed below (200 ms – 3000 ms). Figure 26 shows an example data file generated from the last command (opet_acq 30, TestData1.dat, 0); see Section 2.5 for more details.

- opet_cmd_usb 0x2200, 0x4000, 0x00000000, 3000, 0
  Boot-up DBs: loads the FPGA firmware with default values to all connected Detector Boards.
  Correct response: 0xa200, 0x4000, 0x0

- opet_cmd_usb 0x2403, 0x4000, 0x00000000, 1000, 0
  Initializes all the ADC registers with default values for the Detector Board in slot 0.
  Correct response: 0xa403, 0x4000, 0x30000000

- opet_cmd_usb 0x2404, 0x4000, 0x00000106, 1000, 0
  Sets the ADC gain to 6 dB for all ADCs on the Detector Board in slot 0.
  Correct response: 0xa404, 0x4000, 0x30000106

- opet_cmd_usb 0x2401, 0x4000, 0x00000000, 200, 0
  Initializes all the DAC registers with default values for the Detector Board in slot 0.
  Correct response: 0xa401, 0x4000, 0x30000000

- opet_cmd_usb 0x2402, 0x4000, 0x00602150, 200, 0
  Sets all DACs to +2.150 V for all channels on the Detector Board in slot 0.
  Correct response: 0xa402, 0x4000, 0x30602150

- opet_cmd_usb 0x2205, 0x4000, 0x0000FFFF, 200, 0
  Enables the trigger for channels 0-15 for the Detector Board in slot 0.

- opet_cmd_usb 0x2201, 0x4000, 0x00000000, 200, 0
  Configures the system data mode to Oscilloscope mode for all connected Detector Boards.
  Correct response: 0xa202, 0x4000, 0x00000000

- opet_cmd_usb 0x2203, 0x4000, 0x02000020, 200, 0
  Configures the CDUC and all connected Detector Boards with the Oscilloscope mode settings of test communication data format and 32 raw ADC samples per channel.
  Correct response: 0xa203, 0x4000, 0x02000020

- opet_cmd_usb 0x2209, 0x4000, 0x00F0F0F0, 200, 0
  Clears the event FIFO in the CDUC and all connected Detector Boards to remove data from previous events.
  Correct Response: 0xa209, 0x4000, 0x00F0F0F0

- opet_acq 30, TestData1.dat, 0
  Acquires data from the CDUC through the USB module 0 for 30 seconds and saves the data to file TestData1.dat.

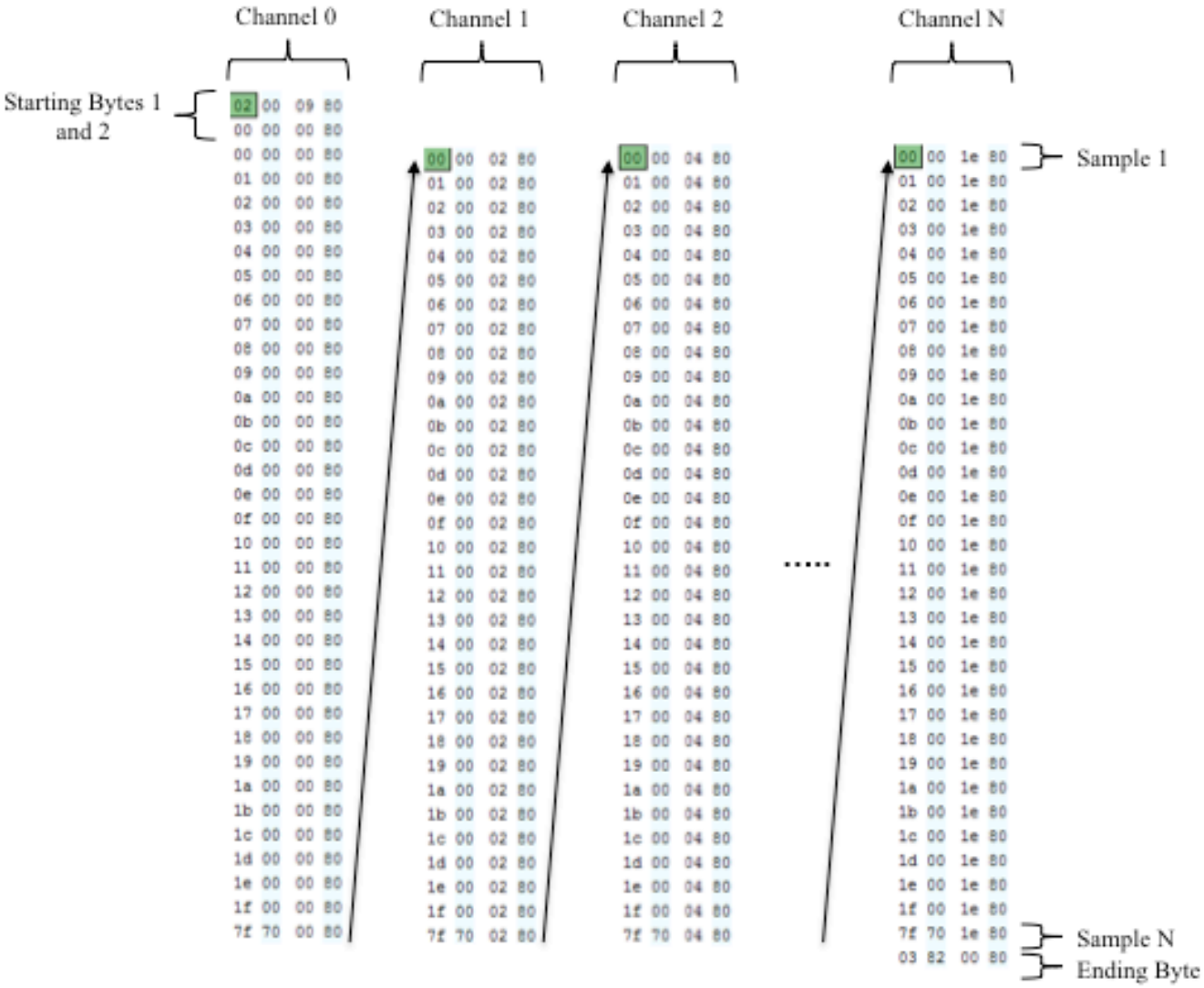


Figure 26: TestData1.dat example data file generated from above command sequence, with 32 samples per channel. See Section 2.5 for more details on the data format and Section 9.2 on diagnostic testing.

2.5 Data Analysis

The list mode data will be written to a binary file so the user can analyze it offline. The format for this list mode data is specified in the following sections.

Users can also use an optional analysis tools package based on the ROOT framework, called OpenPET Control and Analysis Tools (OpenPET CAT). See Section 6 for further details on how to install and use OpenPET CAT.

2.5.1 Data Format

A Coincidence Detector Unit Controller has eight FIFO data buffers, one for each Detector Board. The CDUC checks these eight FIFOs periodically to see if any have been filled. Once the event data are buffered in a FIFO, the CDUC reads the entire event data train from the FIFO and sends it to the host PC.

When a trigger signal is detected by a Detector Board, it collects raw event data and sends them to the CDUC. If the CDUC FIFO for the Detector Board is empty, the raw event data are buffered in the FIFO. Otherwise, the event data are dropped.

2.5.1.1 Single Detector Board

In Oscilloscope mode, the definition of the data format is shown in Figure 27. The data train consists of two 32-bit Starting words, a series of 32-bit raw data, and a 32-bit Ending word. The detailed format for the starting words, raw data and ending words are specified in Figure 28 through Figure 32.

The start and end of an event data train for a Detector Board are identified using a starting flag bit and ending flag bit, respectively. Starting word 1 sets the starting flag bit (bit 16) to “1”, and the following words in the data train have this starting flag bit and the ending flag bit (bit 15) set to “0”. The Ending word then sets the ending flag bit to “1”, signifying that data buffering is complete.

The raw data words are written in sequential order for a single Detector Board. For the Oscilloscope mode ADC plus TDC data train, this means that the raw ADC data are written for the first sample of Channel 0, followed by the second sample and so on until Sample N, and then followed by the raw TDC data for Channel 0. This is repeated sequentially for all 16 Detector Board channels.

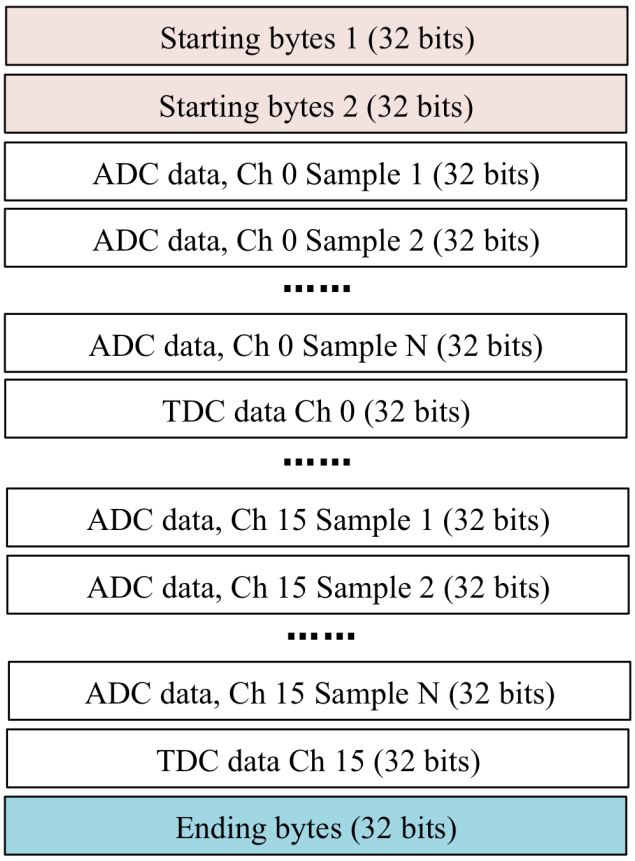


Figure 27: Definition of the Oscilloscope Mode data format for an ADC+TDC data train, where the maximum N is 254.

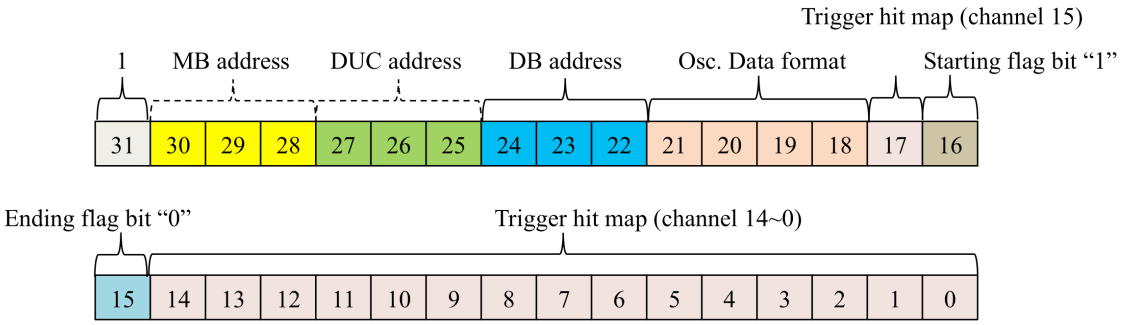


Figure 28: Format of the 32-bit Starting word 1 for an Oscilloscope Mode data train.

The format of Starting word 1 is specified in detail in Figure 28. The 32 bits in the first Starting word are defined below:

- **Bit 31**: Data valid bit. This bit has to be set to "1". A parent node rejects any event data with a data valid bit set to "0" from its offspring nodes.
- **Bit 30 to 28**: 3 Multiplexer Board address bits. The 3-bit MB addresses are assigned and added to the 32-bit event data in the CUC. In a system without a Multiplexer Board (a Small or Standard system), these 3 bits can be redefined as payload bits by the users.

- **Bit 27 to 25**: 3 DUC address bits. The 3-bit DUC addresses are assigned and added to the 32-bit event data in the MB (for a Large System) or CUC (for a Standard System). In a Small System, these 3 bits can also be redefined as payload bits by the users.
- **Bit 24 to 22**: 3 DB address bits. The 3-bit DB addresses are assigned and added to the 32-bit event data in the DUC or CDUC. These 3 bits CANNOT be redefined.
- **Bit 21 to 18**: 4-bit Oscilloscope mode data format. This 4-bit address specifies the data format for Oscilloscope mode:
  - **0000: ADC plus TDC data format**
  - 0001: ADC data format
  - 0010: Test communication data format
  - 0011: Test analog data format
  - 0100-0111: Reserved
  - 1000-1110: User defined data format
  - 1111: Reserved
- **Bit 17**: Trigger hit map (channel 15). This bit indicates whether or not channel 15 triggered.
- **Bit 16**: Starting flag bit. This bit is set to “1” in the Starting word 1 (and “0” for the following words in the data train).
- **Bit 15:** Ending flag bit. This bit is set to “0” in the Starting words.
- **Bits 14 to 0**: Trigger hit map (channels 0-14). This 15-bit address indicates whether or not channels 0-14 in the Detector Board triggered.

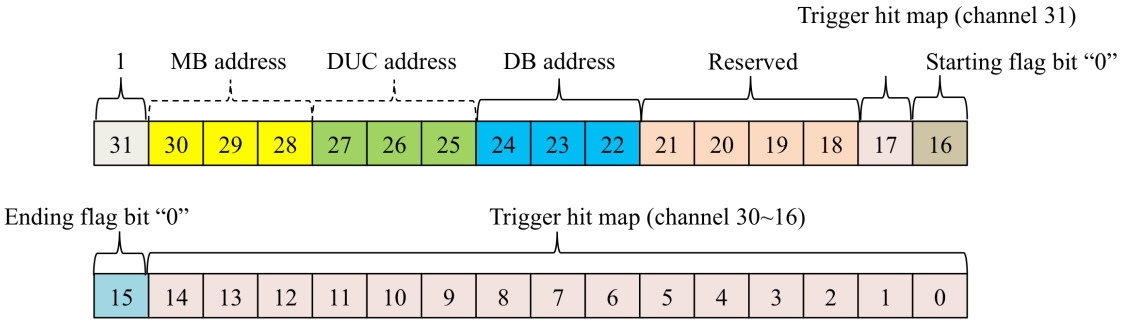


Figure 29: Format of the 32-bit Starting word 2 for an Oscilloscope Mode data train.

The format of Starting word 2 is specified in detail in Figure 29. This Starting word can be ignored when using a 16-channel Detector Board. The 32 bits in the second Starting word are defined below:

- **Bit 31**: Data valid bit. This bit has to be set to “1”. A parent node rejects any event data with a data valid bit set to “0” from its offspring nodes.
- **Bit 30 to 28**: 3 Multiplexer Board address bits. The 3-bit MB addresses are assigned and added to the 32-bit event data in the CUC. In a system without a Multiplexer Board (a Small or Standard system), these 3 bits can be redefined as payload bits by the users.
- **Bit 27 to 25**: 3 DUC address bits. The 3-bit DUC addresses are assigned and added to the 32-bit event data in the MB (for a Large System) or CUC (for a Standard System). In a Small System, these 3 bits can also be redefined as payload bits by the users.
- **Bit 24 to 22**: 3 DB address bits. The 3-bit DB addresses are assigned and added to the 32-bit event data in the DUC or CDUC. These 3 bits CANNOT be redefined.
- **Bit 21 to 18**: Reserved.
- **Bit 17**: Trigger hit map (channel 31). This bit indicates whether or not channel 31 triggered.
- **Bit 16**: Starting flag bit. This bit is set to “0” in the Starting word 2.
- **Bit 15:** Ending flag bit. This bit is set to “0” in the Starting words.

- **Bits 14 to 0**: Trigger hit map (channels 16-30). This 15-bit address indicates whether or not channels 16-30 in the Detector Board triggered.

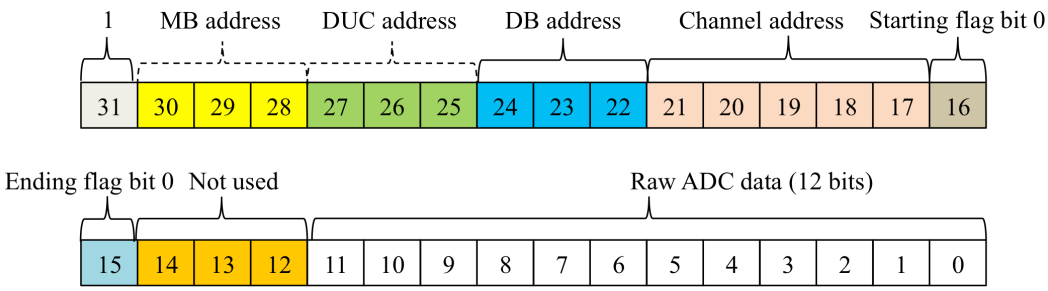


Figure 30: Format of the 32-bit raw ADC data for an Oscilloscope data train.

The raw data format is specified in Figure 30 for ADC data, where the 32-bits in an ADC data word is defined below:

- **Bit 31**: Data valid bit. This bit has to be set to "1". A parent node rejects any event data with a data valid bit set to "0" from its offspring nodes.
- **Bit 30 to 28**: 3 Multiplexer Board address bits. The 3-bit MB addresses are assigned and added to the 32-bit event data in the CUC. In a system without a Multiplexer Board (a Small or Standard system), these 3 bits can be redefined as payload bits by the users.
- **Bit 27 to 25**: 3 DUC address bits. The 3-bit DUC addresses are assigned and added to the 32-bit event data in the MB (for a Large System) or CUC (for a Standard System). In a Small System, these 3 bits can also be redefined as payload bits by the users.
- **Bit 24 to 22**: 3 DB address bits. The 3-bit DB addresses are assigned and added to the 32-bit event data in the DUC or CDUC. These 3 bits CANNOT be redefined.
- **Bit 21 to 17**: 5 channel address bits. The 5-bit channel addresses are assigned in the Detector Board when a single event is detected.
- **Bit 16**: Starting flag bit. This bit is set to "0" for raw data words.
- **Bit 15**: Ending flag bit. This bit is set to "0" for raw data words.
- **Bits 14 to 12**: Not used.
- **Bits 11 to 0**: 12-bit raw ADC data.

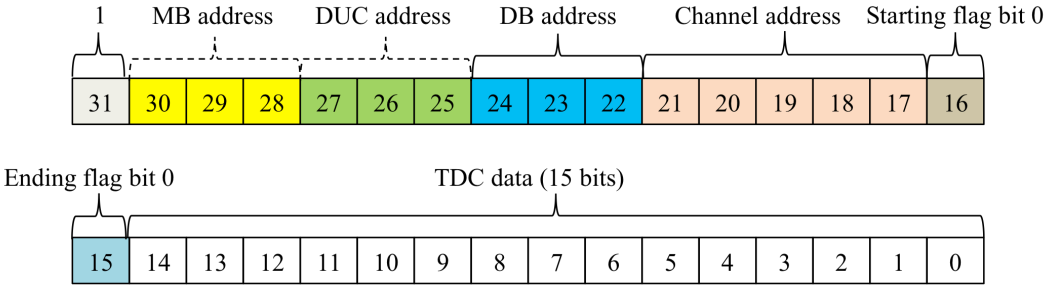


Figure 31: Format of the 32-bit raw TDC data for an Oscilloscope data train.

The raw TDC data format is specified in Figure 31 for the ADC + TDC data train (i.e., Oscilloscope mode data format address = 0000 in Starting word 1), where the 32-bits in a TDC data word is defined below:

- **Bit 31**: Data valid bit. This bit has to be set to "1". A parent node rejects any event data with a data valid bit set to "0" from its offspring nodes.

- **Bit 30 to 28**: 3 Multiplexer Board address bits. The 3-bit MB addresses are assigned and added to the 32-bit event data in the CUC. In a system without a Multiplexer Board (a Small or Standard system), these 3 bits can be redefined as payload bits by the users.
- **Bit 27 to 25**: 3 DUC address bits. The 3-bit DUC addresses are assigned and added to the 32-bit event data in the MB (for a Large System) or CUC (for a Standard System). In a Small System, these 3 bits can also be redefined as payload bits by the users.
- **Bit 24 to 22**: 3 DB address bits. The 3-bit DB addresses are assigned and added to the 32-bit event data in the DUC or CDUC. These 3 bits CANNOT be redefined.
- **Bit 21 to 17**: 5 channel address bits. 5-bit channel addresses are assigned in the Detector Board when a single event is detected.
- **Bit 16**: Starting flag bit. This bit is set to “0” for raw data words.
- **Bit 15**: Ending flag bit. This bit is set to “0” for raw data words.
- **Bits 14 to 0**: 15-bit raw TDC data.

The raw data words are written in sequential order for a single Detector Board. For the ADC plus TDC data train, this means that the raw ADC data are written for the first sample of Channel 0, followed by the second sample and so on until Sample N, and then followed by the raw TDC data for Channel 0. This is repeated sequentially for all 16 Detector Board channels (see Figure 27).

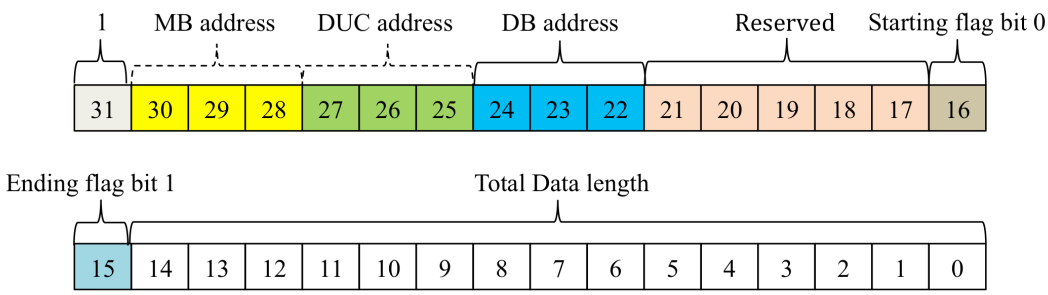


Figure 32: Format of the 32-bit Ending word for an Oscilloscope Mode data train.

The Ending word format is specified in Figure 32, where the 32-bits are defined below:

- **Bit 31**: Data valid bit. This bit has to be set to “1”. A parent node rejects any event data with a data valid bit set to “0” from its offspring nodes.
- **Bit 30 to 28**: 3 Multiplexer Board address bits. The 3-bit MB addresses are assigned and added to the 32-bit event data in the CUC. In a system without a Multiplexer Board (a Small or Standard system), these 3 bits can be redefined as payload bits by the users.
- **Bit 27 to 25**: 3 DUC address bits. The 3-bit DUC addresses are assigned and added to the 32-bit event data in the MB (for a Large System) or CUC (for a Standard System). In a Small System, these 3 bits can also be redefined as payload bits by the users.
- **Bit 24 to 22**: 3 DB address bits. The 3-bit DB addresses are assigned and added to the 32-bit event data in the DUC or CDUC. These 3 bits CANNOT be redefined.
- **Bit 21 to 17**: Reserved.
- **Bit 16**: Starting flag bit. This bit is set to “0” for Ending words.
- **Bit 15**: Ending flag bit. This bit is set to “1” for Ending words.
- **Bits 14 to 0**: 15-bit total data length. The 15-bit total data length address can be used for data validation (e.g., to verify that data bits were not erroneously dropped).

2.5.1.2 Multiple Detector Boards

Section 2.5.1.1 describes the data format when detectors from a single Detector Board are triggered.

When a system has multiple Detector Boards, each DB will receive trigger signals and buffer event data in its respective FIFO (when the FIFO is empty and available). Once a complete data train from a Detector Board is buffered in its FIFO, it is ready for the CDUC to read the entire event data and transfer it to the host PC. The CDUC will periodically check the eight FIFOs to see if any have been filled, transmitting the Detector Board data trains in the order they are buffered.

Since the Detector Board data trains are transferred to the host PC according to their buffering order, you cannot assume that the Detector Boards are read out sequentially in time. In order to look for coincident events, you need to check the TDC data of nearby events.

For example, assume Detector Boards 0, 1 and 5 receive trigger signals and start filling their respective FIFOs in the order listed. Detector Board 0 fills its FIFO first and its data train is transferred to the Host PC. Meanwhile, Detector Board 5 fills its FIFO before Detector Board 1, so the CDUC transfers the data train from Detector Board 5 before sending the data train from Detector Board 1 (Figure 33). Hence, the Detector Boards are not read out in order with incrementing TDC. In order to identify coincident events, you must check the TDC data of the 7 nearest neighboring events.

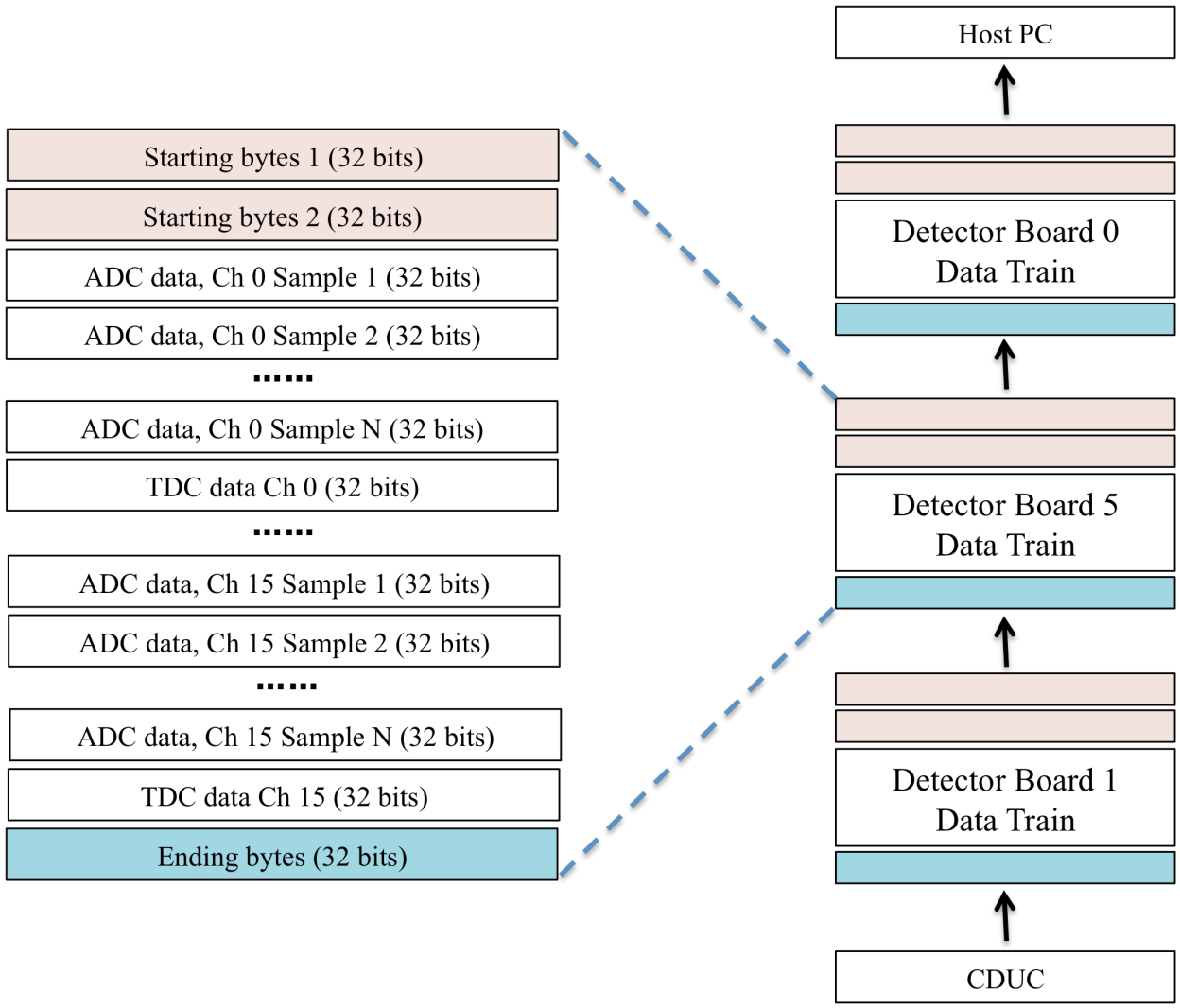


Figure 33: Example transmission of Oscilloscope Mode data trains from the CDUC to the Host PC.

2.6 Getting Help

The goal of OpenPET is to create an active community of general users and developers, so that we can all pool our resources and expertise. In order to help build this OpenPET support network, we have created an opt-in email list for people that want to keep informed of OpenPET news and updates. You can add your email to this OpenPET email list by sending a message to sympa@lists.lbl.gov with the subject line "subscribe openpet-users@lbl.gov".

For general questions, please see the documentation available on the OpenPET website, including presentations, publications and user guides (http://openpet.lbl.gov/documentation/general-documents). You can also check the website's Frequently Asked Questions at http://openpet.lbl.gov/users/faq/.

If you have specific questions, you can read and post questions using the OpenPET forums on our website at http://openpet.lbl.gov/forums/. Please check past discussion threads before creating a new one.

3 Detector Board

The purpose of the Detector Board is to accept analog inputs from the detectors and convert them into Singles Event Words. Generally speaking, this requires identifying the energy, interaction position, and arrival time associated with a single gamma ray interaction, and as many corrections as possible should be applied before the Singles Event Word is generated. This board must also have the ability to produce "singles events" that have Alternate Event Formats, which are necessary for debugging, calibration, etc.

Each DB will contain an FPGA and an associated memory (SRAM). The DB FPGA will perform the necessary computation to convert the detector signals into a Singles Event Word. Firmware is loaded into the DB FPGA by the Support Board via the standard OpenPET Bus IO between SB and DB, as described in Section 3.1.

While the details of the signal processing depend strongly on the details of the detector module, the following is an example describing the processing performed for a block detector module with four analog outputs. Event processing is initiated by the OR of the low to high transitions from each of the Timing Signals. The total amount of signal observed by each of the four PMTs (A, B, C, & D) is computed by summing the output of each ADC for an appropriate period of time (typically 2–3 times the decay time of the scintillator). If necessary, pulse pile-up correction is also applied. These four signals are then summed to get a raw estimate of the energy (E=A+B+C+D), and the appropriate Anger logic estimators are computed (X=(A+B)/E, Y=(A+C)/E). Note that a fast division algorithm can be implemented using look-up tables in the Detector Memory. The X and Y values are used to address a crystal map table that resides in the Detector Memory, and so assign a crystal of interaction via a look-up table. The raw energy E and the crystal of interaction are again used to address another look-up table in the Detector Memory, and so determine whether the event satisfies the energy window criteria. Each Timing Signal is input to a TDC that is implemented in the Detector FPGA, and so measures the (raw) position of the arrival time of each signal within a Time Slice, and a timing estimator computed from these digitized arrival times. The crystal of interaction and raw arrival time are used to address a look-up table stored in Detector Memory and create a corrected arrival time. Thus, the position (crystal of interaction), energy, and arrival time have all been computed. If the event satisfies the energy window criteria, these data are formatted to create a Singles Event Word, and then passed to Support Board through the Bus IO block.

Note that different programs can be loaded into the DB FPGA to perform tasks other than event processing, such as debugging, testing, and calibration tasks.

Several versions of the DB will be designed, mostly differing in how the analog inputs are being processed by the front-end circuitries and in the number of analog input channels per DB. The details of each of the design are described in the subsections below.

3.1 Bus IO

A diagram of the Bus IO is shown in Figure 34. The top block essentially provides control signals. A copy of the System Clock and Time Slice Boundary signals are sent from the Support Board and are used to clock data from the Support Board to the Detector Board. Because there is propagation delay within the Detector Board, another copy of the System Clock and Time Slice Boundary that is produced by the Detector Board is used to clock data from the Detector Board to the Support Board. Lines controlled by the Support Board are used to program the DB FPGA on the Detector Board. There are lines that can simultaneously pass four Singles Event Words from the Detector Board to the Support Board in a combination serial / parallel protocol. For each Singles Event Word, one bit of data is passed on each of four parallel lines during each System Clock cycle, with the entire Singles Event Word being passed during a single Time Slice. Finally, eight additional "spare" lines can be used to pass bi-directional information between boards.

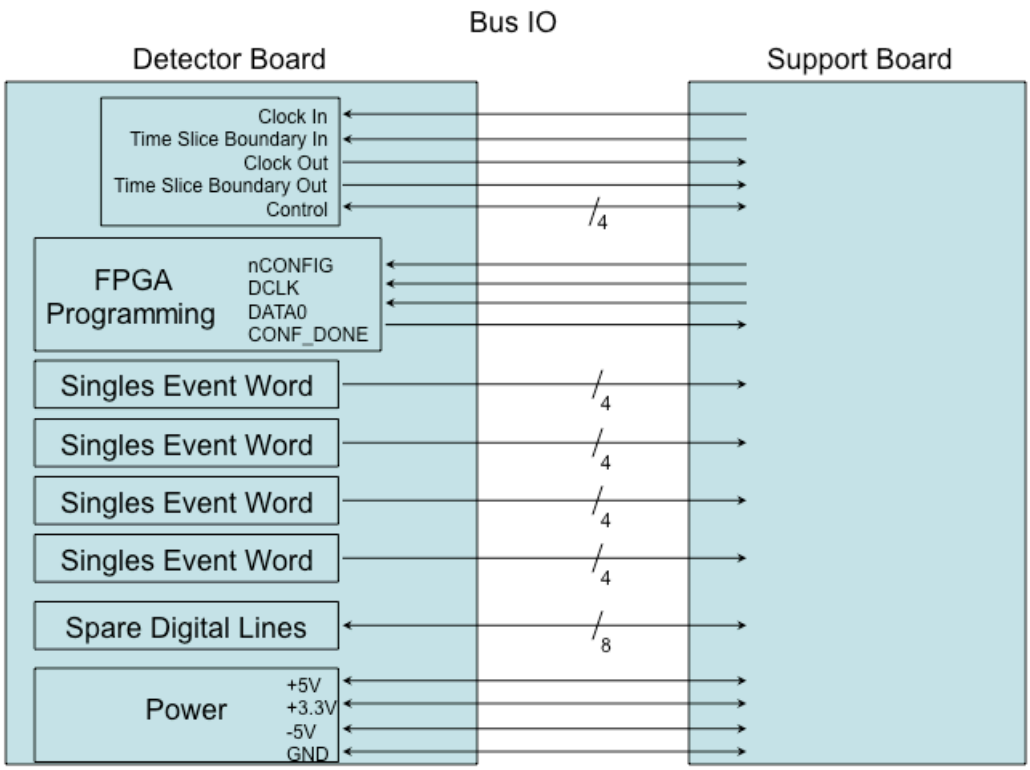


Figure 34: Diagram of the BUS IO.

Figure 35 and the discussion below detail the connections between the Support Board and the Detector Board.

Singles Event Data
There will be 16 LVDS differential pairs that are used to transfer digital Singles Event Words from the Detector Board (DB) to the Support Board (SB).

Clock
There will be 4 LVDS differential pairs that are used to transfer digital timing signals between the Detector Board (DB) and the Support Board (SB). A Clock In and Time Slice Boundary In will be sent from the SB to the DB, and data going from the SB to the DB will be synchronous with these clock signals. Similarly Clock Out and Time Slice Boundary Out will be sent from the DB to the SB, and data going from the DB to the SB will be synchronous with these clock signals.

Control
There will be 4 LVTTL single ended lines used to pass control data from the SB to the DB. They will use a variant of the SPI (Serial Peripheral Interface) Bus protocol, which consists of a Clock line, a Data In line, a Data Out line and a Board Enable line.

FPGA Programming
There will be 4 single ended LVTTL lines used to program the FPGA using the serial protocol. These signals will be provided by the SB.

Spare Digital
There will be 8 LVDS differential pairs between the FPGAs on the SB and the DB. Their purpose is not defined, as the intent is to provide users flexibility in passing information between the two boards.

Power and Ground

The SB will supply power to the DB. It will supply +5V, +3.3V, –5V, ground, and a detector bias voltage (–100V to +100V). With the exception of the detector bias voltage, it is assumed that this is "digital quality" power—that the DB will use these as inputs to on-board regulators to create analog quality power, as well as whatever other digital voltages are necessary. It is also assumed that the +3.3V will be used to supply power for the digital components, while the +5V and –5V will be used to supply power to the analog components.

Connector

We will use the same 96-pin connector as is used in VME modules. On the DB, this is a right angle male connector for a 0.0625" thick board, Vector Electronics RE96MSR-062 (Digi-Key part # V1235-ND). Note that the pin assignment is not compatible with the VME standard, and so the OpenPET electronics CANNOT be plugged into VME crates.

	A	B	C
1	D0+	GND	D1+
2	D0-	3.3V	D1-
3	D2+	GND	D3+
4	D2-	+5V	D3-
5	D4+	GND	D5+
6	D4-	-5V	D5-
7	D6+	GND	D7+
8	D6-	3.3V	D7-
9	D8+	GND	D9+
10	D8-	+5V	D9-
11	D10+	GND	D11+
12	D10-	-5V	D11-
13	D12+	GND	D13+
14	D12-	3.3V	D13-
15	D14+	GND	D15+
16	D14-	+5V	D15-
17	GND	-5V	GND
18	CLK_IN+	3.3V	CLK_OUT+
19	CLK_IN-	+5V	CLK_OUT-
20	GND	-5V	GND
21	SLICE_IN+	3.3V	SLICE_OUT+
22	SLICE_IN-	GND	SLICE_OUT-
23	GND	NC	GND
24	SPARE0+	SPARE1+	SPARE2+
25	SPARE0-	SPARE1-	SPARE2-
26	SPARE3+	SPARE4+	SPARE5+
27	SPARE3-	SPARE4-	SPARE5-
28	SPARE6+	SPARE7+	CTRL_CS
29	SPARE6-	SPARE7-	CTRL_CLK
30	CTRL_DO	NC	CTRL_DI
31	DCLK	DETECTOR BIAS	DATA0
32	nCONFIG	3.3V	CONF_DONE

Color	Group Description
	LVDS Data between DB FPGA & SB FPGA
	Clock & Slice IN/OUT
	Undefined pins between DB FPGA & SB FPGA
	Slow control SPI interface signals
	DB FPGA serial programming pins
	No connection
	Power & GND
	Detector Bias Voltage (100 V max.)

Connector is
96 pin VME Connector

On DB
Vector Electronics RE96MSR-062
(Digi-Key part # V1235-ND)

On SB
Vector Electronics RE96FSP
(Digi-Key part # V1240-ND)

Figure 35: Connections between the Support Board and the Detector Board.

3.2 16-Channel Detector Board

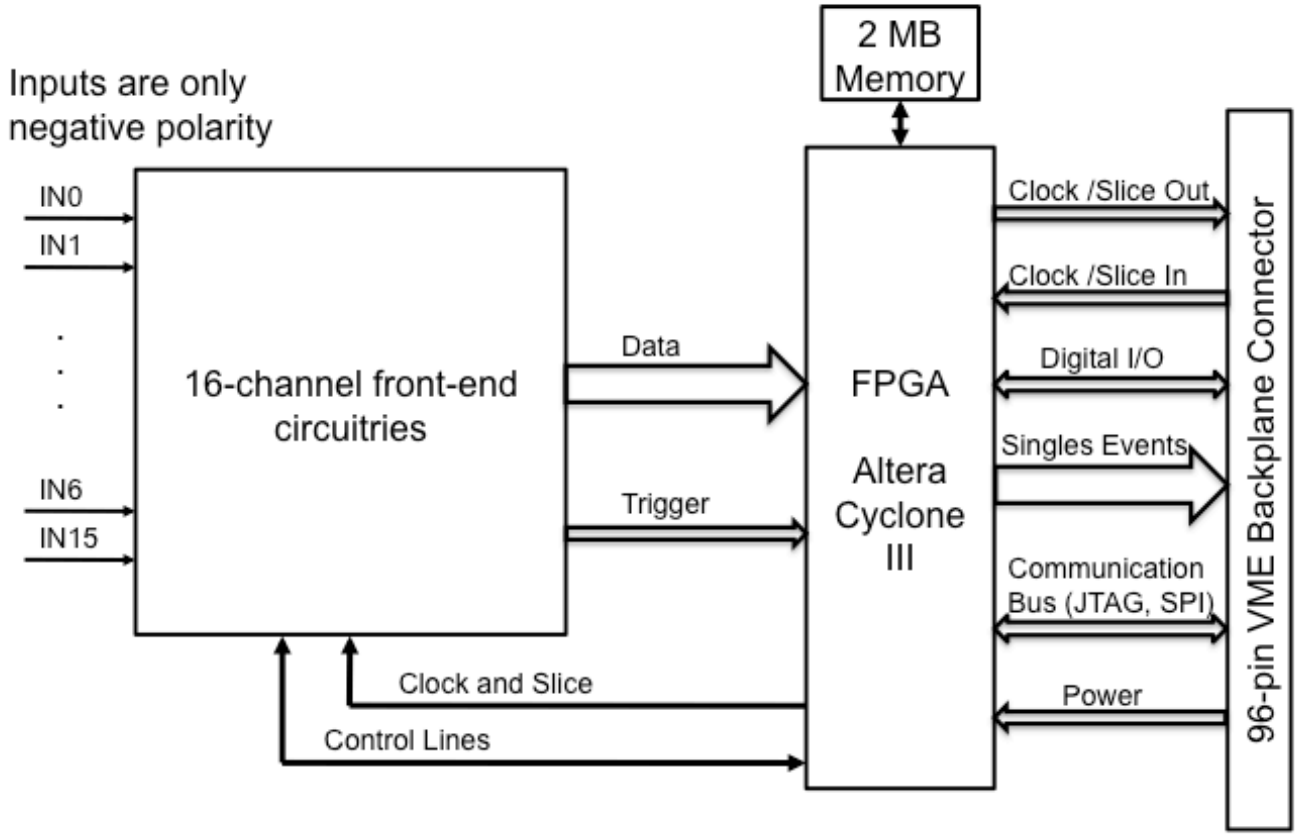


Figure 36: Block diagram of the 16-channel Detector Board.

A schematic of the 16-channel Detector Board is shown in Figure 36 and a photograph is shown in Figure 38. This DB accepts up to 16 analog input signals, each of which is processed independently. The processing circuit for one channel, as shown in Figure 37, consists of two processing chains, where the input signal is split into a timing chain and an energy chain. The energy chain amplifies the input signal and then splits the amplified signal into an anti-aliasing filter and a comparator. The filtered output is sent to an ADC with programmable digital gain that digitizes the analog signal with a sampling rate of 40 – 65 MHz. The comparator provides a trigger signal to start event processing; the trigger threshold is controlled by the energy DAC. The timing chain amplifies the input signal with a high-gain high bandwidth amplifier followed by an ultra fast discriminator that converts the analog signal into a digital timing signal whose leading edge is synchronized to the interaction time. The ADC values and the Timing Signal are sent to the DB FPGA.

Inside the DB FPGA, a TDC generates a time stamp indicating the arrival time of the Timing Signal relative to the Time Slice Boundary. The DB FPGA and its memory also analyzes the ADC data from this channel and (potentially) combines it with information from other channels to compute the energy deposit, the interaction position, and the event time. Appropriate calibration correction factors are also stored in the memory and applied to the data.

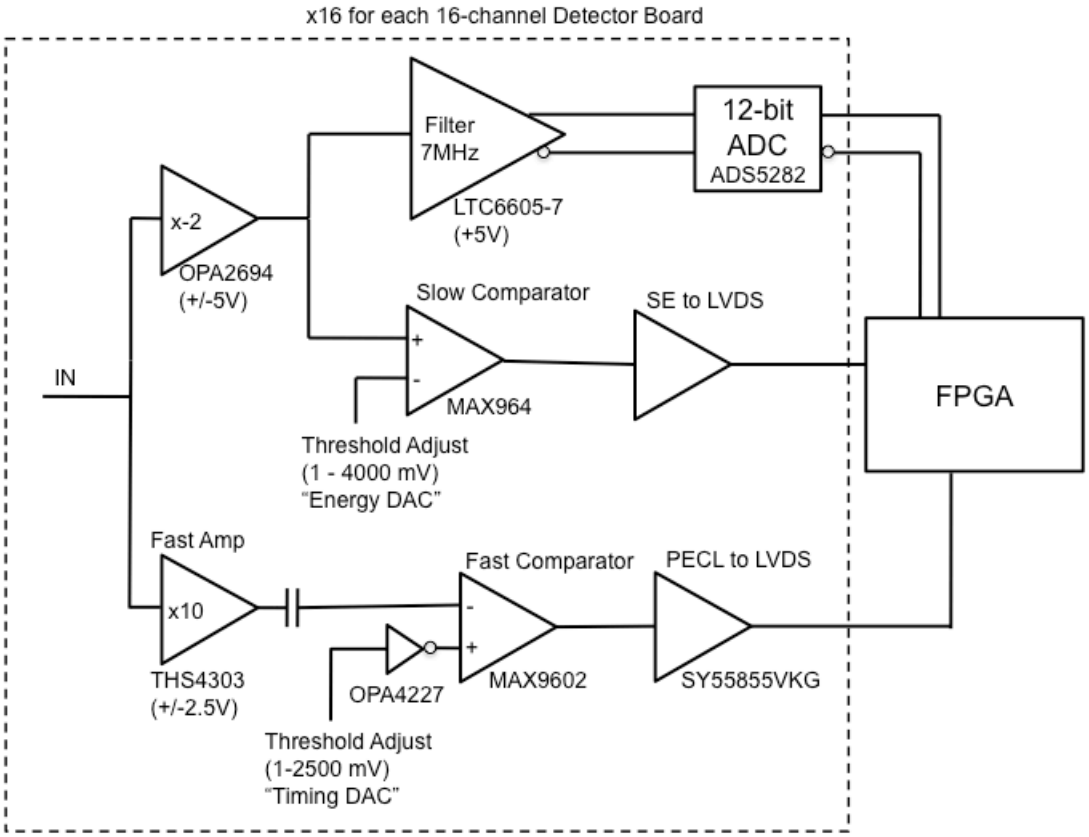


Figure 37: Block diagram of the front-end circuitries of one channel of the 16-channel Detector Board.

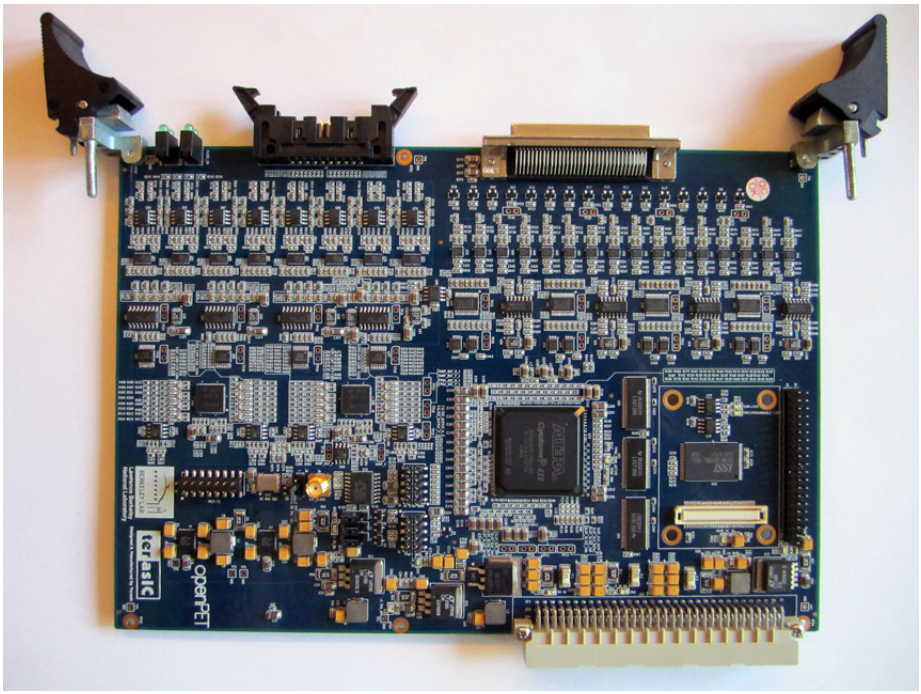


Figure 38: Photograph of the 16-channel Detector Board.

3.2.1 Analog Signal Conditioning

Each input signal is required to be negative polarity. The signal is terminated, and then split into two processing chains: timing chain and energy chain. The input stage accepts voltages between -0.8 V to 0 V and has input diodes to protect against over and under voltage. The range of the input voltages is limited in part by the dynamic range of the ADC. The input voltage can be attenuated to fall within the acceptable range by changing the attenuation resistor values where the signal is split into the timing chain and energy chain. Only the signal on the energy chain needs to be attenuated.

3.2.2 Timing Signal

This circuit is a high-speed leading edge discriminator with a threshold that is controlled via a timing DAC. The timing edge is a high to low transition.

3.2.3 ADC

Each analog input signal is digitized by a 12-bit ADC with digital programmable gain that digitizes the analog signal at a sampling rate ranging from 40 MHz to 65 MHz.

3.2.4 TDC

The TDC digitizes the arrival time of the Timing Signal with respect to the Time Slice Boundary. Thus, its minimum and maximum counts correspond to the beginning and end of a single Time Slice respectively. The TDC is synchronized to the System Clock, which is used to create the higher order bits of the TDC. However, the granularity of the System Clock is too coarse for PET, and so the TDC must generate a minimum of four additional low order bits of timing data. Thus, the least bit of the TDC will be 0.8 ns or smaller. Although it may not be implemented initially, the long-range plan is to have a TDC with 50 ps fwhm timing resolution, which should be more than sufficient for time-of-flight PET.

3.2.5 Detector Memory

2 MBytes of SRAM memory is attached to and controlled by the DB FPGA. This control also includes loading the contents of the memory.

3.2.6 Analog Input Connections

Figure 39 and the discussion below detail the analog input connections to the 16-channel Detector Board. Note that these inputs can be from a user-supplied analog conditioning circuit board (*e.g.*, a preamplifier board), and so the connector contains pins used to provide power to and communicate with one of these (optional) circuit boards.

Analog Inputs
There are 16 analog input channels on the 16-channel DB. Each is a single-ended negative polarity input signal with a negative-going leading edge. Input voltage levels should be between –0.8 V and 0 V.

Power and Ground

The 16-channel DB is capable of supplying power to the analog conditioning board. It supplies +5V, –5V, and ground. It is assumed that this is "digital quality" power — that the analog conditioning board will use these as inputs to on-board regulators to create analog quality power, as well as whatever other digital voltages are necessary. The maximum current for each of the voltage supplies is 1 A.

Connector

We use a 68-pin D-sub connector with 0.050" Pitch x 0.100 Row to Row. The part number is AMP-Part-5787169-7 (Digi-Key part # A33512-ND).

PIN#	DESCRIPTION	DESCRIPTION	PIN#
1	+5 V	–5 V	**2**
3	GND	IN 16	**4**
5	GND	IN 15	**6**
7	GND	IN 14	**8**
9	GND	IN 13	**10**
11	GND	IN 12	**12**
13	GND	IN 11	**14**
15	GND	IN 10	**16**
17	GND	IN 09	**18**
19	GND	IN 08	**20**
21	GND	IN 07	**22**
23	GND	IN 06	**24**
25	GND	IN 05	**26**
27	GND	IN 04	**28**
29	GND	IN 03	**30**
31	GND	IN 02	**32**
33	GND	IN 01	**34**
35	+5V	–5 V	**36**
37	GND	GND	**38**
39	GND	GND	**40**
41	GND	GND	**42**
43	GND	GND	**44**
45	GND	GND	**46**
47	GND	GND	**48**
49	GND	GND	**50**
51	GND	GND	**52**
53	GND	GND	**54**
55	GND	GND	**56**
57	GND	GND	**58**
59	GND	GND	**60**
61	GND	GND	**62**
63	GND	GND	**64**
65	GND	GND	**66**
67	GND	GND	**68**

Color	Group Description
	Power & GND
	Analog Input to DB

Connector on DB is
68 pin D-Sub Connector
AMP-Part-5787169-7
0.050" Pitch x 0.100 Row to Row
Digi-Key Part # A33512-ND

Figure 39: Pin assignment for the Analog Input Connector to the 16-channel Detector Board.

4 Support Board

The main purpose of the Support Board is to accept Singles Event Words from multiple Detector Boards, multiplex them, and pass these Singles Event Words to the Coincidence Interface Board. In addition, it provides the control and power for the Detector Boards, and is the interface with the Host PC. It can also be configured to act as a low-performance version of a Coincidence Board, and so identify coincident events and pass them to the Host PC.

The Support Board, shown schematically in Figure 40, services up to 8 Detector Boards. The standard OpenPET Bus IO circuit, as described in Section 3.1, connects the Support Board to each Detector Board. Three FPGAs on the Support Board (possibly with the help of Support Memory) multiplex the Singles Event Words and pass them through the slot 8 Bus IO block to the Coincidence Interface Board. The event multiplexing and forwarding is shared among three FPGAs (one Master FPGA and two Slave FPGAs) due to limited pin count. Several other blocks, such as a clock-conditioning block that ensures the fidelity of the System Clock, logic analyzer connectors, two RS-232 ports, and diagnostic LEDs, are not shown in Figure 40.

High-level commands are sent via USB (or alternatively, through Gigabit Ethernet) from the Host PC to a Support Microprocessor, which interprets and executes these commands. This execution may involve controlling the Detector Board, such as by loading a program into the DB FPGA on the DB, or may involve higher level functions, such as performing a calibration by instructing the DB to produce calibration data, analyzing the forthcoming calibration events, computing calibration parameters, and loading these parameters into the Detector Memory on the DB.

The SB is also capable of identifying coincident pairs of Singles Event Words, format them into Coincidence Event Format, and pass them to the Support Microprocessor, which then passes them to the Host PC. Thus it can act as a full-featured PET data acquisition system, albeit with a limited number of input channels and output event rate capability.

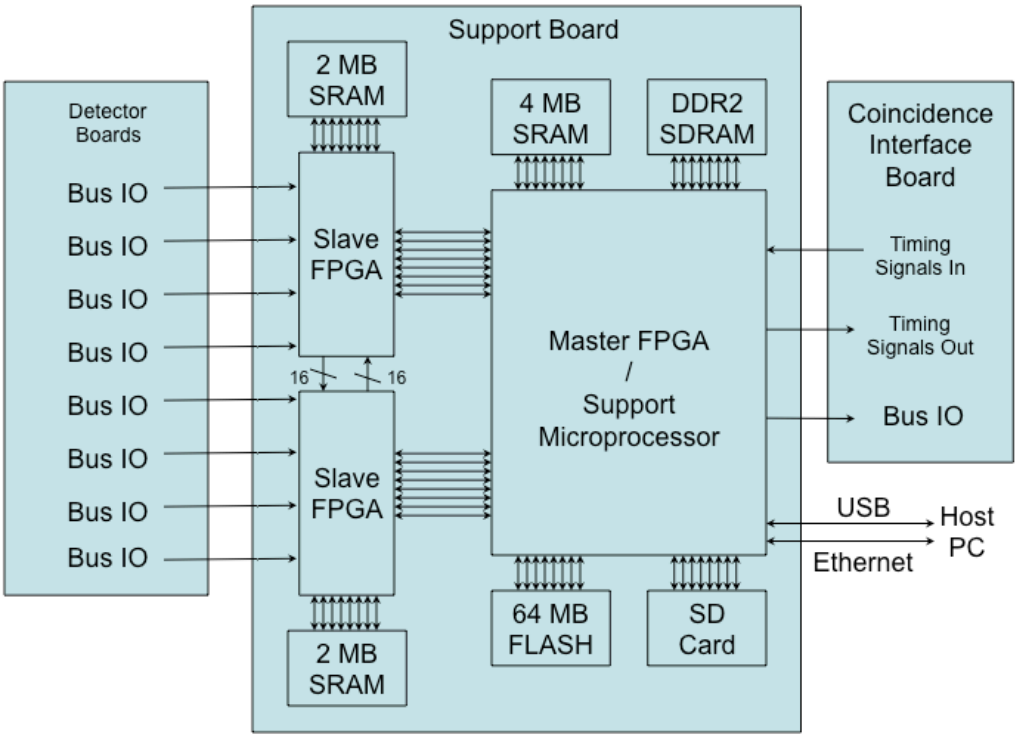


Figure 40: Schematic of the Support Board.

4.1 Slave FPGAs

There are two Slave FPGAs on the SB. Each one of these acts primarily as a multiplexer for Singles Events, each taking up to 16 individual Singles Events that it can receive in a single Time Frame and passing up to 4 of them to the Master FPGA. Obviously, there is some possibility for data loss, and the multiplexing algorithm is designed to ensure that this loss is unbiased. Each Slave FPGA also serves as a fan-in / fan-out for communication between the Support Microprocessor and the individual Detector Boards, and the two Slave FPGAs can communicate with each other.

The digital signals between each Slave FPGA and the Master FPGA are identical to the Bus IO signals between the Slave FPGA and a single Detector Board: lines for the Clock and Time Slice Boundary (both directions), FPGA programming lines, 16 event data lines, and eight user-definable data lines. There are also 32 user-definable digital lines between the two Slave FPGAs, 16 sending data in each direction.

4.2 Master FPGA & Support Microprocessor

A single physical FPGA performs the logical functions of both the Master FPGA and the Support Microprocessor. Its FPGA-like functions are mostly limited to passing events from the Slave FPGAs to the Coincidence Interface Board (providing multiplexing, if necessary).

Some of the logic blocks in the Master FPGA shown in Figure 40 can be programmed (using Nios II) to act as the Support Microprocessor. Nios II is used to program some of the logic blocks in this FPGA to be identical to microprocessor hardware, which then runs executable files programmed in C. This microprocessor is connected via USB (or Gigabit Ethernet) to the Host PC, from which it receives high-level commands, and then interprets and executes these commands. It is responsible for loading all the programs into the Slave FPGAs and DB FPGA, as well as the contents of all the Support Memory and Detector Memory, and all of the other registers that are on the DB and SB. It monitors the event stream and can insert diagnostic information (such as event rates) into the event stream or provide this information directly to the Host PC. Whenever possible, calibration routines are also performed on the Support Microprocessor.

4.3 Support Memory

There are multiple forms of memory on the Support Board. The SRAM is used by the “FPGA-like” functions. This memory provides its output within 1 clock cycle of being addressed, and so is both reasonably fast and deterministic, which greatly simplifies incorporating it within FPGA algorithms. However, SRAM capacity is fairly small. Thus, the Master FPGA is connected to 4 MB of SRAM and each of the Slaves FPGAs is connected to 2 MB of SRAM, for a total of 8 MB of SRAM on each Support Board. This memory is typically used to store look-up tables that apply real time calibration and corrections to the event data.

As the Master FPGA also emulates a microprocessor, “disk storage” and “RAM memory” are also necessary for it to function effectively. The “RAM memory” is provided via up to 1 GB of RAM that can be plugged into a DDR2 SDRAM connector (identical to that typically found in laptop computers). The “disk storage” is provided by a SD Card (identical to that found in digital cameras) that is plugged into a SD Card slot. Our vision is that it will contain all the FPGA firmware needed for Detector FPGAs, the contents of all the Support SRAM Memory and Detector Memory, and all of the other registers that are on the Detector Board and Support Board. For systems where the convenience of using “disk storage” that can be easily removed is not desired, the same function can be provided by a 64 MB FLASH memory chip that is also connected to the Master FPGA. This information can also be stored in the on-board EPCS memory, which is where it resides in the initial release.

4.4 Clock Conditioning

The Clock Conditioning block consists of a PLL (phase-locked loop) that regenerates the System Clock signal from the Support Board in a Coincidence Unit and passes it to the Support Board FPGAs in a Detector Unit and then to the Detector Boards. The block also includes space for a clock IC, which is used to provide the System Clock when the system is being used without a Coincidence Unit (*i.e.,* when the Support Microprocessor passes events directly to the Host PC).

4.5 Connectors, Slots 0-7

The connector that is soldered onto the SB in order to connect to a single DB is a 96 pin female connector, press fit for a 0.125" thick board, Vector Electronics V1240-ND (Digi-Key part # RE96FSP).

4.6 Slots 8–11

The rightmost four slots (slot numbers 8 through 11) will each contain a board with a specific purpose, but in general are used to facilitate connection to and communication between various parts of the system.

4.6.1 Coincidence Interface Board (Slot 8)

The purpose of the board in slot 8 is to communicate with the Coincidence Unit. The formats of the signals that are passed between the Detector Unit and the Coincidence Unit via the Coincidence Interface Board are identical to those being passed between the Detector Board and the Support Board on slots 0-7.

There will be two versions of the Coincidence Interface Board, one called Coincidence Interface Board CI-1 to interface with the Multiplexer Board MB-1 in the Standard System and another called Coincidence Interface Board CI-8 to interface with the Multiplexer Board MB-8 in the Large System. CI-1 is a passive board with no active components—just traces connecting the front panel and rear connectors. This front panel connector will then connect to a cable that brings these signals to MB-1 in the Coincidence Unit. On the other hand, CI-8 will have active components (e.g., FPGA, etc.) to interface with MB-8. These boards are only necessary if the system contains a Coincidence Unit.

4.6.2 Host PC Interface (Slot 9)

The purpose of this board is to provide the interface to the Host PC. The front panel contains an Ethernet connector, a USB connector, an SD Card connector, three reset switches, 20 LEDs, and a Detector Bias Voltage input (BNC, maximum 100 V, positive or negative polarity). The board itself holds a Gigabit Ethernet transceiver chip, as well as a connector that a QuickUSB card must be plugged into in order for the USB communication to function. In the initial release, only communication through the QuickUSB connector is supported.

4.6.3 User IO (Slot 10)

The purpose of this board is to provide User IO. It has an external clock input, two DB9 RS-232 connectors that are connected to the Main FPGA (which can be used to communicate to motor controllers, etc.) and 48 digital IO lines. An on-board jumper selects whether these 48 lines use 5 V or 3.3 V logic level. Each of the three FPGAs (the Main and the two Slaves) is connected to 16 IO lines. On board jumpers select the direction (input or output) of each IO line in groups of 4 (*e.g.*, the direction Master FPGA IO lines 0-3 are set by a single jumper, and so must be the same).

4.6.4 Debugging (Slot 11)

The purpose for this board is Debugging. It contains a JTAG connector (that can be used to program the FPGAs directly), four Aligent 16902B connectors for logic analyzers (two connect to the Main FPGA, and one to each of the two Slave FPGAs), and 30 user-defined LEDs (10 connected to each of the 3 FPGAs).

5 Commands

In general, every OpenPET system command also has an associated response. Both commands and responses (C/R) have the format shown in Figure 41. There are four segments of information required: command/response id (16 bits), source address (16 bits), target address (16 bits), and command/response data payload (32 bits).

The command/response id specifies the command type, using a 16-bit number with the most significant bit (C/R flag) as 0 for a command and 1 for a response. For example, command id 0x2200 boots up the Detector Boards, and the corresponding response id is 0xa200. Within the command/response id, the 4-bit C/R node type specifies whom the command is for within the OpenPET tree topology (Section 1.4). For a small system, the C/R node type will always be 0100 to specify the CDUC node. (Note: the C/R node type in the C/R ID should be the same as the node type in the source/target address.) The standard/user C/R flag (bit 10) in the command/response id is reserved for future use, so commands can be grouped by functionality. The 10-bit C/R function type specifies the command task (where bit 9 is 0 for universal tasks and 1 for hardware-specific tasks). So all 16-bits together specify the command id or response id (see Tables 3 and 4).

The source address is a 16-bit number that defines where the command/response originates. Commands originate in the host PC, which has source address 0xF000.

The target address is a 16-bit number that identifies where the command/response should be received and processed. Typically commands for a small system have a target address of 0x4000 for the CDUC. However, some underlying commands target a Detector Board, which has a target address of 0x3000. (Note: the DB address of 0-7 corresponds to DB hardware slots 0-7; for example, the DB address for slot 3 is 30c0.)

The C/R data payload is a 32-bit number that specifies additional information for each command; see below for examples.

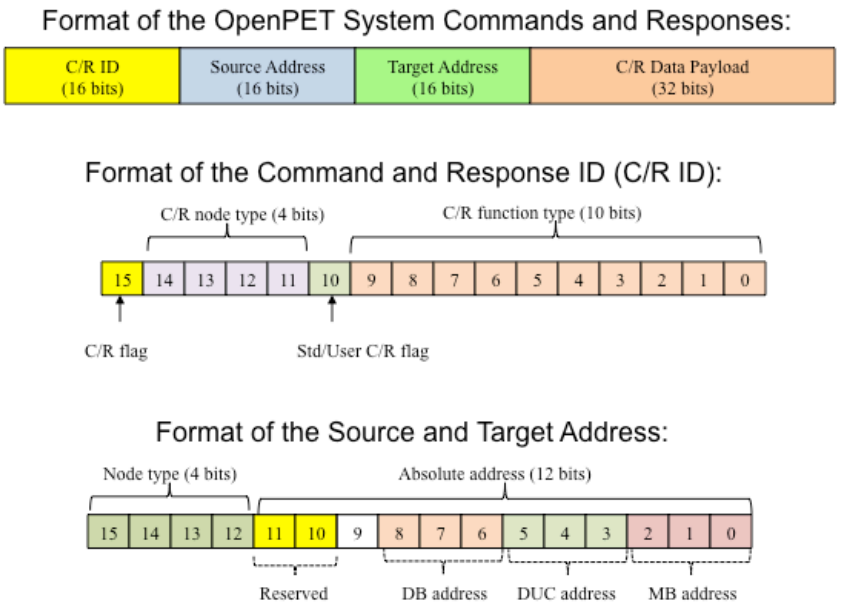


Figure 41: OpenPET command/response address formats. Bits shown in white are not used.

The executable opet_cmd_usb.exe is used to control the system. In particular, it is used to send a command to target hardware that provides a response. The command executable has several arguments:

opet_cmd_usb command ID, target address, command payload, timeout, USB module number

Tables 3 and 4 show a list of the current OpenPET commands, including their command id and a brief description of their function. As discussed above, the target address for user commands for a small system is 0x4000 to specify a CDUC with absolute address of 0. The command payload is specified for each command in the examples that follow. The timeout is measured in milliseconds within an allowed integer range of -30,000 to 30,000 and a default value of 200. A negative timeout indicates not to send the command, but to wait to read responses from the USB FIFO; the magnitude of the timeout value is not used for a negative timeout. A timeout of 0 indicates to send a command without reading a response. And a positive timeout indicates to send a command and wait to read a response until the specified timeout. Finally, the USB module number is an integer between 0-7 with the default value of 0. USB module numbering applies only to QuickUSB modules plugged into the host PC (not other peripheral USB devices), and the QuickUSB module numbers are based on the order in which they were plugged in.

In summary, the opet_cmd executable arguments are:

- **Command ID:** 16 bits
  - (See Table 2)
- **Target address:** 16 bits
  - Coincidence Detector Unit Controller: 0x4000
- **Command payload:** 32 bits
  - (See examples below)
- **Timeout:** units of milliseconds, default 200, integer range -30,000 to 30,000
- **USB module number:** default 0, integer range 0-7
- The timeout and USB module number are optional arguments, so they can be omitted. However, if USB module is provided, the timeout must also be provided.
- All bits specified as “not used” or “reserved” should be set to 0.

Command ID (Response ID)	Command Name	Command Function
0x2200 (0xa200)	Boot-up Detector Boards	Loads the FPGA configuration to all Detector Boards.
0x2201 (0xa201)	Set system data mode	Configures the system data mode. At present, only Oscilloscope data mode is implemented.
0x2202 (0xa202)	Read system data mode	Reads the system data mode.
0x2203 (0xa203)	Set Oscilloscope data settings	Configures the Oscilloscope mode data settings. Can specify data format and number of ADC samples per channel.
0x2204 (0xa204)	Read Oscilloscope data settings	Reads the Oscilloscope mode data settings.
0x2205 (0xa205)	Set trigger mask (channels 0-15)	Configures the trigger mask. Can enable/disable trigger for channels 0-15 for a single DB or all DBs.
0x2206 (0xa206)	Set trigger mask (channels 16-31)	Configures the trigger mask. Can enable/disable trigger for channels 16-31 for a single DB or all DBs.
0x2207 (0xa207)	Read trigger mask (channels 0-15)	Reads the trigger mask for channels 0-15 for a single DB.
0x2208 (0xa208)	Read trigger mask (channels 16-31)	Reads the trigger mask for channels 16-31 for a single DB.
0x2209 (0xa209)	Clear event FIFO	Clears the event FIFO to remove data from previous events.

Table 3: Summary of the OpenPET commands for universal tasks.

Command ID = 0x2200 **Load FPGA firmware into all Detector Boards**

Payload:

- Bits 31-1: Not used
- Bit 0: Selects which FPGA settings to load to DBs (0 =default, 1 = previous settings)

Note:

- This command will load the default/previous FPGA settings into all the Detector Boards

Example: opet_cmd_usb 0x2200, 0x4000, 0x00000000, 3000, 0

- Bits 31-1: Unused bits should be set to 0
- Bit 0: 0 indicates to load default settings

→ Loads the FPGA firmware into all Detector Boards of the CDUC using default values, with a 3000 ms timeout for a response.

Correct response: 0xa200, 0x4000, 0x0
(Response payload = command payload)

Command ID = 0x2201 **Configures system data mode**

Payload:

- Bits 31-8: Not used
- Bits 7-0: System data mode (0x00 for Oscilloscope mode, currently the only system data mode that has been implemented and the default)

Example 1: opet_cmd_usb 0x2201, 0x4000, 0x00000000, 200, 0

- Bits 31-8: Unused bits should be set to 0
- Bits 7-0: 0x00 selects Oscilloscope mode

→ Configures the system data mode to Oscilloscope mode for the CDUC and all connected Detector Boards, with a 200 ms timeout for a response.

Correct response: 0xa201, 0x4000, 0x00000000
(Response payload = command payload)

Command ID = 0x2202 **Reads system data mode** (e.g., 0x00000000 for Oscilloscope)

Payload:

- Bits 31-0: Not used

Example 1: Setup: opet_cmd_usb 0x2201, 0x4000, 0x00000000, 200, 0
Read: opet_cmd_usb 0x2202, 0x4000, 0x00000000, 200, 0

- Bits 31-0: Unused bits should be set to 0

→ Reads the system data mode of the CDUC and all connected Detector Boards, with a 200 ms timeout for a response.

Note:

- Before this command is run, command 0x2201 must be run to configure the system data mode

Correct response: 0xa202, 0x4000, 0x00000000
(Response payload bits 7-0 = CDUC data mode value)
Error: response payload bit 8 = 1 indicates that the system data mode values from the DBs do not match the CDUC value

Command ID = 0x2203 **Configures Oscilloscope mode data settings**

Payload:

- Bits 31-28: Reserved.
- Bits 27-24: Oscilloscope mode data format address:
  - **0000: ADC plus TDC data format (default)**
  - 0001: ADC data format
  - 0010: Test communication data format*
  - 0011: Test analog data format**
  - 0100-0111: Reserved
  - 1000-1110: User defined data format
  - 1111: Reserved (for invalid Oscilloscope mode data format flag)
- Bits 23-8: Reserved.
- Bits 7-0: Number of raw ADC samples per channel (spaced 25 ns apart, maximum number of 254).

Note:

- * **Test communication data format** = ADC+TDC data format with ADC values = counter values that increment from 0 to N-1, where N is the value set in bits 7-0. For example, ADC data Sample 1 (for all channels) = "0", …, ADC data Sample N (for all channels) = "N-1".

- ** **Test analog data format** = ADC+TDC data format with ADC values from analog input signal that is triggered with an internal clock.

Example 1: opet_cmd_usb 0x2203, 0x4000, 0x00000020, 200, 0

- Bits 27-24: 0x0 selects ADC plus TDC data format
- Bits 23-8: Reserved bits should be set to 0
- Bits 7-0: 0x20 selects 32 raw ADC samples per channel

→ Configures the CDUC and all connected Detector Boards with the Oscilloscope settings of ADC plus TDC data format and 32 raw ADC samples per channel, with a 200 ms timeout for a response.

Correct response: 0xa203, 0x4000, 0x00000020
(Response payload = command payload)
Error: response payload bits 27-24 = 0xF indicates an invalid Oscilloscope mode data format

Example 2: opet_cmd_usb 0x2203, 0x4000, 0x02000020, 200, 0

- Bits 27-24: 0x2 selects test communication data format
- Bits 23-8: Reserved bits should be set to 0
- Bits 7-0: 0x20 selects 32 raw ADC samples per channel

→ Configures the CDUC and all connected Detector Boards with the Oscilloscope settings of test communication data format and 32 raw ADC samples per channel, with a 200 ms
timeout for a response.

Correct response: 0xa203, 0x4000, 0x02000020
(Response payload = command payload)
Error: response payload bits 27-24 = 0xF indicates an invalid Oscilloscope mode data format

Command ID = 0x2204 **Reads Oscilloscope mode data settings**

Payload:

- Bits 31-0: Not used

Note:

- Before this command is run, command 0x2203 must be run to configure the Oscilloscope mode data settings

Example 1: Setup: opet_cmd_usb 0x2203, 0x4000, 0x00000020, 200, 0
Read: opet_cmd_usb 0x2204, 0x4000, 0x00000000, 200, 0

- Bits 31-0: Unused bits should be set to 0

→ Reads the Oscilloscope mode data settings of the CDUC and all connected Detector Boards, with a 200 ms timeout for a response.

Correct response: 0xa204, 0x4000, 0x00000020
(Response payload bits 27-24 = CDUC Oscilloscope mode data format value, bits 7-0 = CDUC raw ADC sample number per channel)
Error: response payload bit 15 = 1 indicates that the raw ADC samples per channel from the DBs do not match the CDUC value
Error: response payload bit 31 = 1 indicates that the data format values from the DBs do not match the CDUC value

Command ID = 0x2205 **Configures trigger mask for channels 0-15**

Payload:

- Bits 31-28: DB address (0-7 corresponding to slots 0-7; 8 for all DBs)
- Bits 27-16: Not used
- Bits 15-0: Enables/disables the trigger for channels 0-15

Note:

- The energy DAC determines whether or not a channel triggers (see Section 3.2).

Example 1: opet_cmd_usb 0x2205, 0x4000, 0x3000FFFF, 200, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-16: Unused bits should be set to 0
- Bits 15-0: 0xFFFF enables the trigger for channels 0-15

→ Enables the trigger for channels 0-15 for the Detector Board in slot 3, with a 200 ms timeout for a response.

Correct response: 0xa205, 0x4000, 0x3000FFFF
(Response payload = command payload)
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2206 **Configures trigger mask for channels 16-31**

Payload:

- Bits 31-28: DB address (0-7 corresponding to slots 0-7; 8 for all DBs)
- Bits 27-16: Not used
- Bits 15-0: Enables/disables the trigger for channels 16-31

Note:

- The energy DAC determines whether or not a channel triggers (see Section 3.2).

Example 1: opet_cmd_usb 0x2206, 0x4000, 0x30000000, 200, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-16: Unused bits should be set to 0
- Bits 15-0: 0x0000 disables the trigger for channels 16-31

→ Disables the trigger for channels 16-31 for the Detector Board in slot 3, with a 100 ms timeout for a response.

Correct response: 0xa206, 0x4000, 0x30000000
(Response payload = command payload)
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2207 **Reads trigger mask for channels 0-15 for a single Detector Board**

Payload:

- Bits 31-28: DB address (0-7 corresponding to slots 0-7)
- Bits 27-0: Not used

Note:

- Before this command is run, command 0x2205 must be run to configure the trigger mask for channels 0-15.

Example 1: Setup: opet_cmd_usb 0x2205, 0x4000, 0x3000FFFF, 200, 0
Read: opet_cmd_usb 0x2207, 0x4000, 0x30000000, 200, 0

- Bits 31-28: 0x3 indicates that the read command is for the Detector Board in slot 3.
- Bits 27-0: Unused bits should be set to 0

→ Reads the trigger mask for channels 0-15 the Detector Board in slot 3, with a 200 ms timeout for a response.

Correct response: 0xa207, 0x4000, 0x3000FFFF
(Response payload bits 31-28 = DB address, bits 15-0 = trigger mask for channels 0-15)
Error: response payload bits 31-0 = 0xFFFFFFFF indicates that the trigger mask register in the DB doesn't match that in the CDUC
Error: response payload bits 31-0 = 0xF0000000 indicates an invalid DB address

Command ID = 0x2208 **Reads trigger mask for channels 16-31 for a single Detector Board**

Payload:

- Bits 31-28: DB address (0-7 corresponding to slots 0-7)
- Bits 27-0: Not used

Note:

- Before this command is run, command 0x2206 must be run to configure the trigger mask for channels 16-31.

Example 1: Setup: opet_cmd_usb 0x2206, 0x4000, 0x0000FFFF, 200, 0
Read: opet_cmd_usb 0x2208, 0x4000, 0x00000000, 200, 0

- Bits 31-28: 0x0 indicates that the read command is for the Detector Board in slot 0.
- Bits 27-0: Unused bits should be set to 0

→ Reads the trigger mask for channels 16-31 of the specified Detector Board, with a 200 ms timeout for a response.

Correct Response: 0xa208, 0x4000, 0x0000FFFF
(Response payload bits 31-28 = DB address, bits 15-0 = trigger mask for channels 16-31)
Error: response payload bits 31-0 = 0xFFFFFFFF indicates that the trigger mask register in the DB doesn't match that in the CDUC
Error: response payload bits 31-0 = 0xF0000000 indicates an invalid DB address

Command ID = 0x2209 **Clears event FIFO**

Payload:

- Bits 31-24: Not used
- Bits 23-16: Reset hold time in ms for DB FPGA (Recommend 240 ms)
- Bits 15-8: Reset hold time in ms for CDUC IO FPGA (Recommend 240 ms)
- Bits 7-0: Reset hold time in ms for CDUC Main FPGA (Recommend 240 ms)

Example 1: opet_cmd_usb 0x2209, 0x4000, 0x00F0F0F0, 200, 0

- Bits 31-24: Unused bits should be set to 0
- Bits 23-0: 0xF0F0F0 sets proper reset hold time of 240 ms for all FPGAs

→ Clears the event FIFO in the CDUC and all connected DBs (to remove data from previous events), with a 200 ms timeout for a response.

Correct Response: 0xa209, 0x4000, 0x00F0F0F0
(Response payload = command payload)

Example 2: opet_cmd_usb 0x2209, 0x4000, 0x00000000, 200, 0

- Bits 31-24: Unused bits should be set to 0
- Bits 23-0: 0x000000 sets reset hold time to 0 ms for all FPGA

→ Does NOT clear the event FIFO in the CDUC and DBs, since reset hold time for all FPGA set to 0 ms. This is not recommended.

Correct Response: 0xa209, 0x4000, 0x00000000
(Response payload = command payload)

Command ID (Response ID)	Command Name	Command Function
0x2400 (0xa400)	Send sawtooth test pulse	Sends a sawtooth test pulse signal to all DACs on a single Detector Board. Can specify which DB and the number of sawtooth waves.
0x2401 (0xa401)	Initialize DAC	Initializes DACs registers with default values. Can specify a single DB or all DBs.
0x2402 (0xa402)	Set DAC voltage	Sends a single voltage signal to DACs on a single Detector Board. Can specify which DB, DAC type, DAC channels and voltage value.
0x2403 (0xa403)	Initialize ADC	Initializes ADCs registers with default values. Can specify a single DB or all DBs.
0x2404 (0xa404)	Set ADC gain	Set ADC gain in a single Detector Board. Can specify gain, which DB, and a single channel or all channels.

Table 4: Summary of the OpenPET commands for DB/hardware specific tasks.

Command ID = 0x2400 **Send sawtooth test pulse to all DACs on a single Detector Board**

Payload:

- Bits 31-28: DB address (0-7, corresponding to slots 0-7)
- Bits 27-25: Reserved (for hardware version control)
- Bits 24-16: Not used
- Bits 15-0: Number of sawtooth waves. Each wave has a period of 1.8 s and a peak-to-peak amplitude of 4.08 V (Energy DACs) or 2.51 V (Timing or Reserved DACs).

Example: opet_cmd_usb 0x2400, 0x4000, 0x30000004, 200, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-16: Reserved and unused bits should be set to 0
- Bits 15-0: 0x0004 indicates to send a 4 sawtooth signal

→ Sends a single 4 sawtooth signal to all DACs on the Detector Board in slot 3 of the CDUC, with a 200 ms timeout for a response.

Correct response: 0xa400, 0x4000, 0x30000004
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2401 **Initialize all DAC registers with default values on a single DB or all DBs**

Payload:

- Bits 31-28: DB address (0-7 corresponding to slots 0-7; 8 for all DBs)
- Bits 27-25: Reserved (for hardware version control)
- Bits 24-0: Not used

Example: opet_cmd_usb 0x2401, 0x4000, 0x30000000, 200, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-0: Reserved and unused bits should be set to 0

→ Initializes all DAC registers with default values (see Appendix 1) for the Detector Board in slot 3 of the CDUC, with a 200 ms timeout for a response.

Correct response: 0xa401, 0x4000, 0x30000000
(Response payload = command payload)
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2402 **Set DAC voltages on a single Detector Board**

Payload:

- Bits 31-28: DB address (0-7, corresponding to slots 0-7)
- Bits 27-25 Reserved (for hardware version control)
- Bits 24-23: Not used
- Bits 22-21: DAC type (0=energy DAC, 1=timing DAC, 2= reserved DAC, 3=all DACs, all channels)
- Bit 20: All DAC channels within single DAC type (0=no, 1=yes)
- Bits 19-16: Individual DAC channel number (0-15))
- Bits 15-12: Voltage value (0-4, volts)
- Bits 11-8: Voltage value (0-9, 0.1 volts)
- Bits 7-4: Voltage value (0-9, 0.01 volts)
- Bits 3-0: Voltage value (0-9, 0.001 volts)

Note:

- Before this command is run, command 0x2401 must be run to initialize the DACs.
- This command will be ignored if the selected input voltage is invalid. The allowed range is 0 - 4.08 V for energy DACs and 0 - 2.51 V for timing or reserved DACs.
- The trigger DAC is usually set to a low threshold to obtain the best timing.
- The energy DAC determines whether readout is initiated, and so is usually set to a higher threshold to reduce noise triggers (see Section 3.2).

Example 1: opet_cmd_usb 0x2402, 0x4000, 0x30602150, 200, 0

- 0x30602150 (=0011 000 00 11 0 0000 0010 0001 0101 0000)
- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-23: Reserved and unused bits should be set to 0
- Bits 22-21: 0x3 indicates that the voltage will be set for all DACs, all channels
- Bits 20: Bit ignored for this case (set to 0)
- Bits 19-16: Bits ignored for this case (set to 0)
- Bits 15-0: Sets the DAC voltage value to 2.150 V

→ Sets all DACs and all channels on the Detector Board in slot 3 of the CDUC to +2.150 V, with a 200 ms timeout for a response.

Correct response: 0xa402, 0x4000, 0x30602150
(Response payload = command payload)
Error: response payload bits 15-0 = 0xFFFF indicates an invalid voltage setting
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Example 2: opet_cmd_usb 0x2402, 0x4000, 0x30032150, 200, 0

- 0x30032150 (=0011 000 00 00 0 0011 0010 0001 0101 0000)
- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-23: Reserved and unused bits should be set to 0
- Bits 22-21: 0x0 indicates that the DAC type is energy
- Bits 20: 0 indicates that an individual channel has been selected
- Bits 19-16: 0x3 indicates that the voltage will be set on DAC channel 4
- Bits 15-0: Sets the DAC voltage value to 2.150 V

→ Sets DAC energy channel 4 on the Detector Board in slot 3 of the CDUC to +2.150 V, with a 200 ms timeout for a response.

Correct response: 0xa402, 0x4000, 0x30032150
(Response payload = command payload)
Error: response payload bits 15-0 = 0xFFFF indicates an invalid voltage setting
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2403 **Initialize all ADC registers with default values on a single DB or all DBs**

Payload:

- Bit 31-28: DB address (0-7 corresponding to slots 0-7; 8 for all DBs)
- Bits 27-25: Reserved (for hardware version control)
- Bits 24-0: Not used

Example: opet_cmd_usb 0x2403, 0x4000, 0x30000000, 1000, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-0: Reserved and unused bits should be set to 0

→ Initializes all ADC registers with default values (see Appendix 1) for the Detector Board in slot 3 of the CDUC, with a 1000 ms timeout for a response.

Correct response: 0xa403, 0x4000, 0x30000000
(Response payload = command payload)
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Command ID = 0x2404 **Set ADC gain(s) on a single 16-channel Detector Board**

Payload:

- Bits 31-28: DB address (0-7, corresponding to slots 0-7)
- Bits 27-25: Reserved (for hardware version control)
- Bits 24-9: Not used
- Bits 8-4: Channel ID (0-15 configures channel 0-15, 16 configures all 16 channels)
- Bits 3-0: Gain (0-12 corresponds to 0-12 dB, with 12 being the highest gain)

Note:

- Before this command is run, command 0x2403 must be run to initialize the ADCs.

Example 1: opet_cmd_usb 0x2404, 0x4000, 0x30000106, 1000, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-9: Reserved and unused bits should be set to 0
- Bits 8-4: 0x10 indicates that the ADCs for all 16 channels will be configured

- Bits 3-0: 0x6 sets the ADC gain to 6 dB

→ Sets the ADC gain to 6 dB for all ADCs on the Detector Board in slot 3 of the CDUC, with a 1000 ms timeout for a response.

Correct response: 0xa404, 0x4000, 0x30000106
(Response payload = command payload)
Error: response payload bits 3-0 = 0xF indicates an invalid ADC gain
Error: response payload bits 8-4 = 0x1F indicates an invalid ADC channel
Error: response payload bits 31-28 = 0xF indicates an invalid DB address

Example 2: opet_cmd_usb 0x2404, 0x4000, 0x30000006, 1000, 0

- Bits 31-28: 0x3 indicates that the command is for the Detector Board in slot 3.
- Bits 27-9: Reserved and unused bits should be set to 0
- Bits 8-4: 0x00 indicates that the ADC for channel 0 will be configured
- Bits 3-0: 0x6 sets the ADC gain to 6 dB

→ Sets the ADC gain to 6 dB for ADC channel 0 on the Detector Board in slot 3 of the CDUC, with a 1000 ms timeout for a response.

Correct response: 0xa404, 0x4000, 0x30000006
(Response payload = command payload)
Error: response payload bits 31-28 = 0xF indicates an invalid DB address
Error: response payload bits 3-0 = 0xF indicates an invalid ADC gain

6 OpenPET Control and Analysis Tools (OpenPET CAT)

OpenPET Control and Analysis Tools (OpenPET CAT) is a data acquisition and analysis software for the OpenPET electronics based on the ROOT framework. OpenPET CAT utilizes the object-oriented design in providing basic utilities, control and analysis tools for the OpenPET electronics. Using OpenPET CAT requires installing the ROOT package (http://root.cern.ch). Using OpenPET CAT is optional; users can run the executables "opet_cmd_usb" to configure the system and "opet_acq_usb" to acquire data directly.

6.1 Installing ROOT

Go to http://root.cern.ch/drupal/content/downloading-root for instructions on downloading and installing ROOT. Although ROOT is supported on many platforms, OpenPET only supports the Windows version of ROOT using the Microsoft Visual C++ compiler. Current OpenPET CAT supports ROOT version 5.32/04 and Microsoft Visual C++ 2010. If you plan to develop and compile OpenPET CAT codes, you also need to install Microsoft Visual C++ 2010 Express, which can be downloaded at no-cost from http://www.visualstudio.com/downloads/download-visual-studio-vs, and cygwin.

6.2 Using OpenPET CAT

6.2.1 Configuration Files

Defining an OpenPET system starts with a system configuration file as shown in Figure 42.

The system configuration file defines:

- **Node Type**: "4" for CDUC.
- **Command Engine Type**: "0" for USB and "1" for Ethernet.
- **Command Engine Id**: "0"
- **Acquisition Engine Type**: "0" for USB and "1" for Ethernet.
- **Acquisition Engine Id**: "0"
- **Configuration Files Directory**: The directory path to the remaining configuration files.
- **MB Setup Files Prefix**: The file prefix to the Multiplexer Board configuration files.
- **MB Configuration**: The multiplexer board address (0 to 7).

Note: Any comments can be inserted between the "***end …" and the next "***…" tags.

```
# This file gives the hardware configuration parameters for
# the specified scanner.

***Node Type***
# 4=cduc
4
***end Node Type***

***Command Engine Type***
# 0=usb, 1=ethernet
0
***end Command Engine Type***

***Command Engine Id***
0
***end Command Engine Id***

***Acquisition Engine Type***
# 0=usb, 1=ethernet
0
***end Acquisition Engine Type***

***Acquisition Engine Id***
0
***end Acquisition Engine Id***

***Configuration Files Directory***
C:\Documents and Settings\seng\My Documents\OpenPET\software\small_system
***end Configuration Files Directory***

***MB Setup Files Prefix***
MB
***end MB Setup Files Prefix***

***MB Configuration***
# MB Address
0
***end MB Configuration***
```

Figure 42**:** Example system configuration for a small system.

Following the system configuration files, there are three more configuration files: 1) Multiplexer Board configuration file; 2) Detector Unit configuration file; and 3) Detector Board configuration file. For example, in Figure 42, there is one multiplexer board with address 0 (note that in the small system, there are no physical MB board, so a dummy MB configuration file with address 0 is used). The location and prefix of this file in specified in "***Configuration Files Directory***" and "***MB Setup Files Prefix***" respectively. In this example, a file " MB0.txt, as shown in Figure 43, would reside in the directory C:\Documents and Settings\seng\My Documents\OpenPET\software\small_system. MB0.txt shows that there is a Detector Unit with address 0 is connected to MB0 and the configuration file for this Detector Unit is MB0_DU0.txt as shown in Figure 44. MB0_DU0.txt shows that the Detector Board type is the 16-Channel DB and there are two DBs with addresses 0 and 2 in this Detector Unit. The configuration files for these Detector Boards are MB0_DU0_DB0.txt and MB0_DU0_DB2.txt. For example, MB0_DU0_DB0.txt is shown in Figure 45, which include all the DAC threshold and ADC gain settings for the 16-Channel DB.

```
# This file gives the hardware configuration parameters for
# the Multiplexer Board.

***DU Setup Files Prefix***
MB0_DU
***DU Setup Files Prefix***

***DU Configuration***
# DU Address
0
***end DU Configuration***
```

Figure 43: Example Multiplexer Board configuration file.

```
# This file gives the hardware configuration parameters for
# the Detector Unit.

***DB Setup Files Prefix***
MB0_DU0_DB
***end DB Setup Files Prefix***

***DB Type***
# 0=16-Ch DB
0
***end DB Type***

***DB Configuration***
# DB Address
0
2
***end DB Configuration***
```

Figure 44: Example Detector Unit configuration file.

```
# This file gives the hardware configuration parameters for
# the Detector Board.

***Channel Enable***
# Channel         Enable
0         1
1         1
2         1
3         1
4         1
5         1
6         1
7         1
8         1
9         1
10        1
11        1
12        1
13        1
14        1
15        1
***end Channel Enable***

***Timing Threshold***
# Channel         DAC(V)
0         0.1
1         0.1
2         0.1
3         0.1
4         0.1
5         0.1
6         0.1
7         0.1
8         0.1
9         0.1
10        0.1
11        0.1
12        0.1
13        0.1
14        0.1
15        0.1
***end Timing Threshold***

***Energy Threshold***
# Channel         DAC(V)
0         0.06
1         0.06
2         0.06
3         0.06
4         0.06
5         0.06
6         0.06
7         0.06
8         0.06
9         0.06
10        0.06
11        0.06
12        0.06
13        0.06
14        0.06
15        0.06
***end Energy Threshold***

***Reserved Voltage***
# DAC(V)
0.0
0.0
***end Reserved Voltage***

***ADC Baseline Voltage***
# DAC(V)
0.0
```

Figure 45: Example Detector Board configuration file.

6.2.2 OpenPET CAT Library and Macros

A library of C++ classes and ROOT macros are implemented to configure the system, and acquire and analyze the data. The OpenPET CAT software package can be downloaded from the OpenPET website (http://openpet.lbl.gov) under the "Downloads" menu. In the 'lib' directory, there is a library file called libopenpetcat.dll, which has to be copied to the directory <installed ROOT directory>/bin (i.e., $ROOTSYS/bin). In the 'examples' directory, there are macros and configuration files to help you get started.

1. acquireData.C(int acqTimeInSec, int initSystem=1)
   This macro configures the system using configuration files and acquires a raw data file.
   Inputs (arguments to the function):
   - acqTimeInSec – acquisition time in seconds
   - initSystem – configures the system (by default set to 1)

   Inputs (edit in the macro file):
   - dataMode – see command ID 0x2201
   - dataFormat – see command ID 0x2203 for configuring oscilloscope mode
   - numADCsample – 1 to 241 (see command ID 0x2203)
   - configFile – system configuration file
   - filename – raw data filename (the date will be prepended to this name)

   Example 1: root> .x acquireData.C(100)
   This macro initializes the system using the configuration files and acquires data for 100 s.

   Example 2: root> .x acquireData.C(100, 0)
   This macro acquires data for 100 s without initializing the system.

2. displayWaveform.C
   This macro displays the acquired waveforms event-by-event.
   Inputs (edit in the macro file):
   - dbType – type of Detector Board (will be determined in the data file in next release)
   - dataMode – same as in acquireData.C (will be determined in the data file in next release)
   - dataFormat – same as in acquireData.C (will be determined in the data file in next release)
   - numADCsample – same as in acquireData.C (will be determined in the data file in next release)
   - numSkippedEvent – number of events to be skipped
   - evtTimeWindow – integer in 80 MHz clock cycle window to be classified as coincident event
   - dataFilename – filename of the raw data file

   Example: root> .x displayWaveform.C

3. analyzeRawData.C
   This macro reads the waveforms event-by-event and integrates each waveform to calculate its energy. It also outputs a root file containing an ntuple with the energy and tdc values for every channel. This macro can only be used when only one 16-channel Detector Board is in the system.
   Inputs (edit in the macro file):
   - dbType – type of Detector Board (will be determined in the data file in next release)
   - dataMode – same as in acquireData.C (will be determined in the data file in next release)
   - dataFormat – same as in acquireData.C (will be determined in the data file in next release)
   - numADCsample – same as in acquireData.C (will be determined in the data file in next release)
   - numSkippedEvent – number of events to be skipped
   - evtTimeWindow – integer in 80 MHz clock cycle window to be classified as coincident event
   - startBaselineBin – start bin for calculating the baseline of the waveform
   - endBaselineBin – end bin for calculating the baseline of the waveform
   - startIntegratingBin – start bin for calculating the energy of the waveform
   - endIntegratingBin – end bin for calculating the energy of the waveform
   - dataFilename – filename of the raw data file

- rootFilename – output ntuple filename

Example: root> .x analyzeRawData.C

6.3 OpenPET CAT Graphic User Interface

6.3.1 Introduction

The OpenPET CAT Graphic User Interface (GUI) is a user interface for the OpenPET CAT data acquisition framework. It was developed so that users may conveniently use the OpenPET CAT software in gathering and analyzing data from the OpenPET system. The Openpet CAT GUI executable (openpet_cat.exe) can be found in the OpenPET CAT 'bin' directory. When you double clicked on the executable, two windows will pop up as shown in Figure 46.

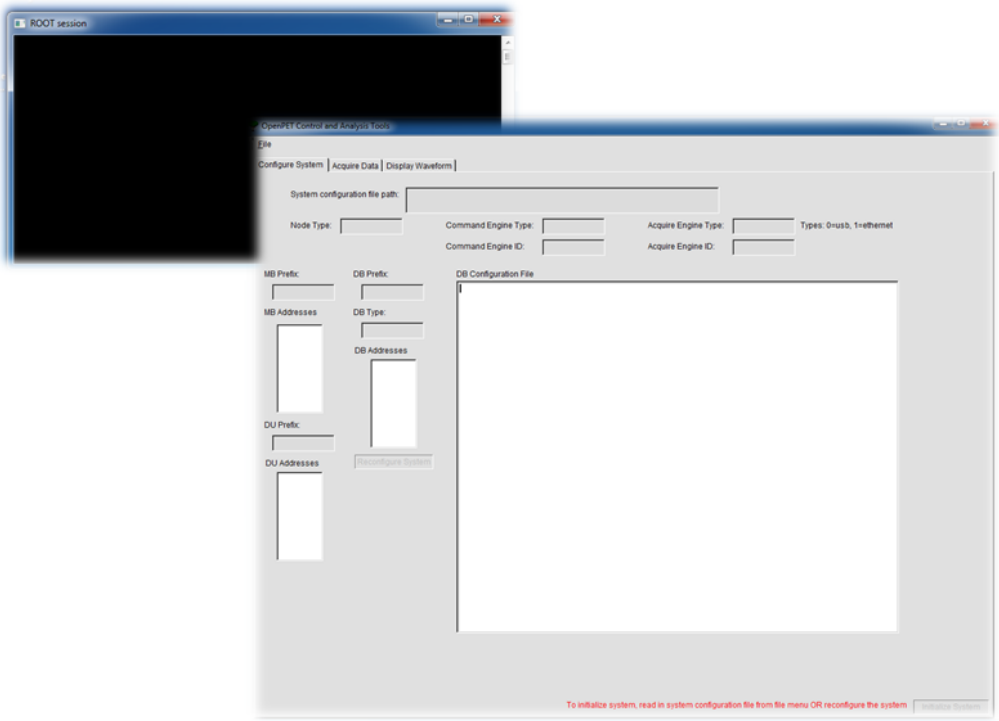


Figure 46: Initial startup windows.

6.3.2 Configure System

Once the application is launched, the initial screen is the Configuration window (Figure 47). This screen will display all hardware configuration data loaded from the system configuration file, as discussed in Section 6.2.1. The top section displays the engine settings and configuration file directory. The bottom section displays the address lists of Multiplexer Boards, Detector Units, and Detector Boards along with an editable text window for the Detector Board configuration file. There is also the Reconfigure System button and Initialize System button. The first button reconfigures the system should any values in the Detector Board configuration file be changed by the user, and the second button initializes the system. Both buttons are disabled until a system configuration file is loaded.

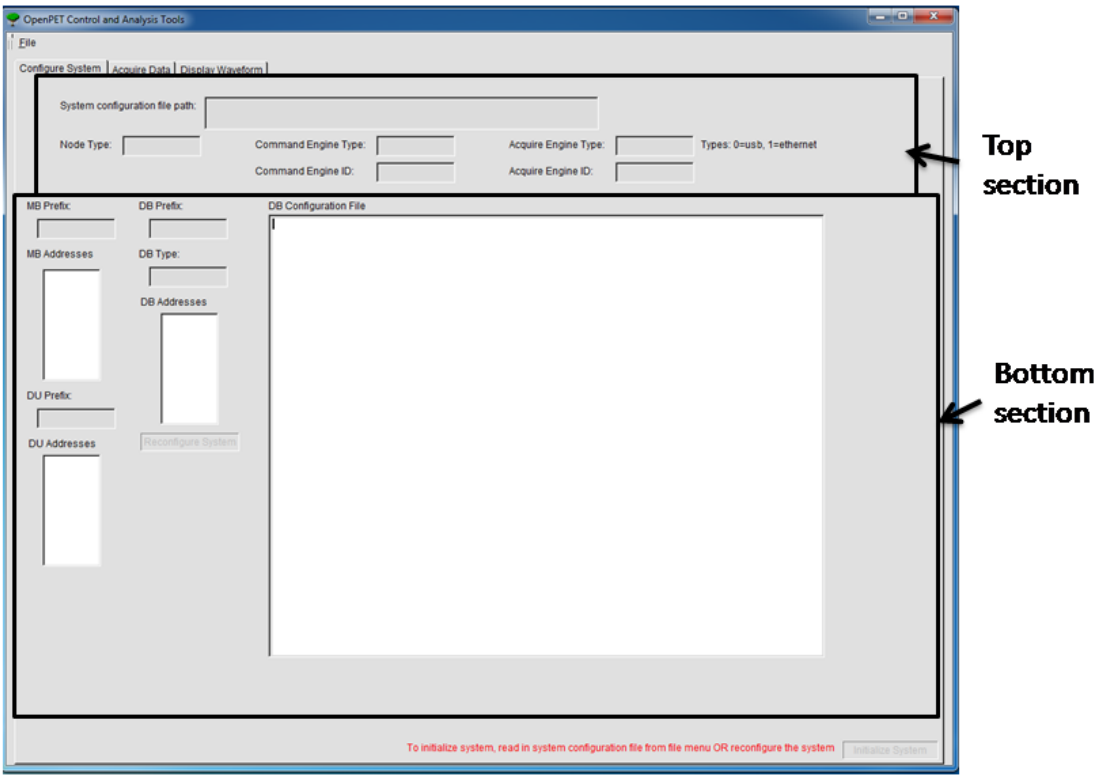


Figure 47: Sections of the Configuration window.

To load a system configuration file, click on the File menu in the top left hand corner and select Read System Configuration File. This will open a dialog window allowing you to select the required system configuration file, as shown in Figure 48. Once the file is selected, the system will configure. You can check the status of the configuration process by reading the terminal window.

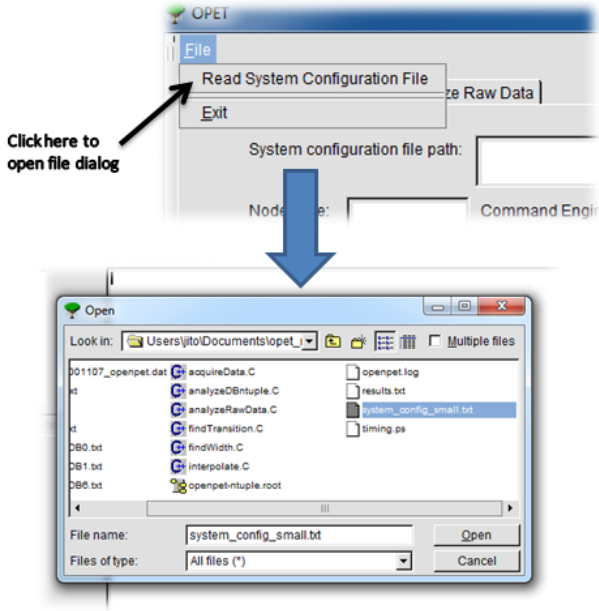


Figure 48: Opening a system configuration file.

After the system has configured, the top section will be filled with the appropriate values along with the list of Multiplexer Board addresses and their file prefix. If you select a certain Multiplexer Board, the list of Detector Unit addresses corresponding to that Multiplexer Board will appear along with their file prefix. Again, if you select a particular Detector Unit, the list of Detector Boards corresponding to that unit will appear along with their type and file prefix. See Figure 49 for an example.

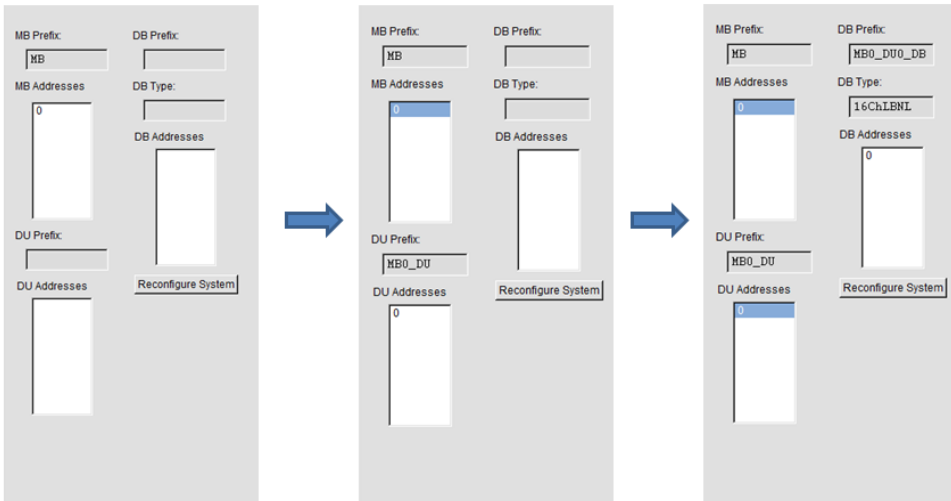


Figure 49: Selecting MB and DU addresses.

At this point, if you are satisfied with the current configuration, you may initialize the system by clicking the Initialize System button (in the bottom right corner of the Configuration window). If not, you can change values in a particular Detector Board's configuration file by selecting the correct MB address, DU address, and DB address to open the desired configuration file, as shown in Figure 50. As you make changes, the configuration file is saved automatically. However, to have these changes take effect in the system, you must reconfigure the system by clicking the Reconfigure System button (on the middle left side of the Configuration window) and then the system may be initialized.

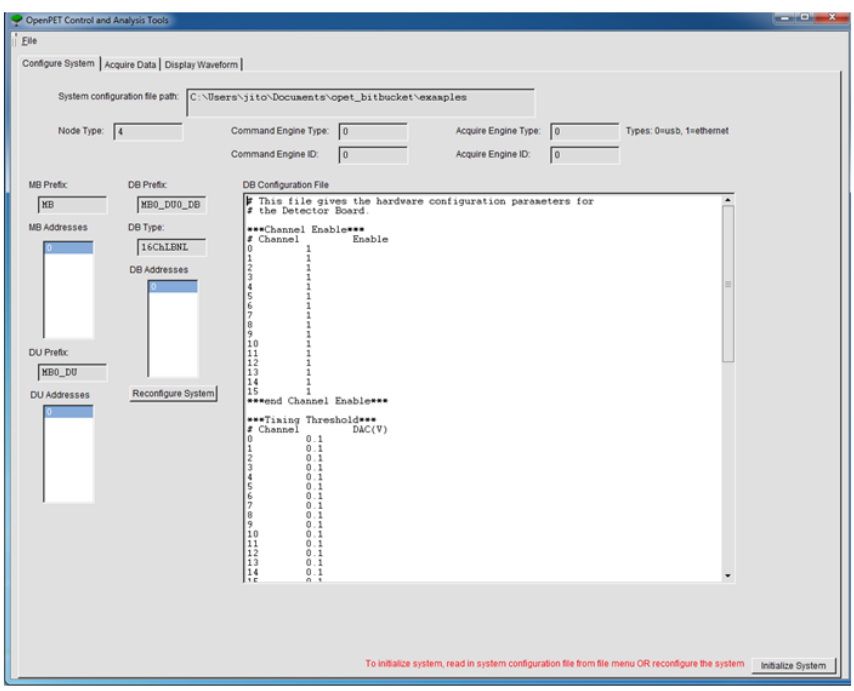


Figure 50: Detector Board configuration file within the Configuration window.

Again, you can observe the status of the system by reading the terminal window. The command and response IDs discussed in Section 5 will be printed there. Therefore, you can determine if the system was correctly initialized.

Once the system is initialized, you may now acquire data.

6.3.3 Acquire Data

Once the system has been initialized, the Acquire Data button (in left upper corner) will now activate so you can acquire data. There are four parameters you must set in the Acquire Data window (see Figure 51):

- Data Mode – see command ID 0x2201 for details
- Data Format – see command ID 0x2203 for details on oscilloscope mode configuration
- Number of ADC samples – 1 to 241; see command ID 0x2203 for details
- Acquisition time – acquisition time in seconds

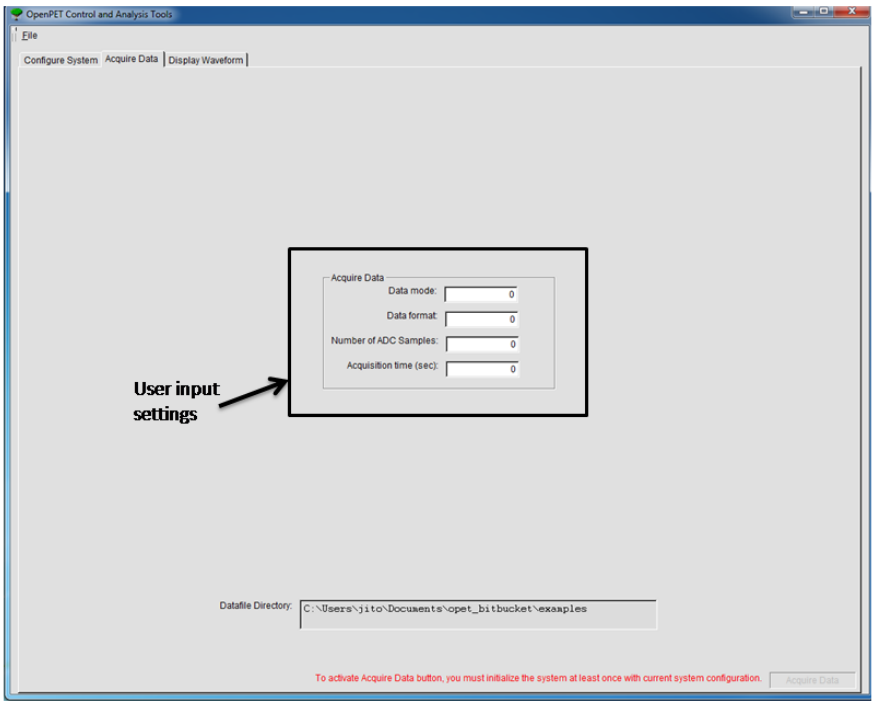


Figure 51: Acquire Data window.

The last values that were used with this system configuration will be set as default values and used the next time the system configuration file is loaded.

The data file will be saved under the current working directory, which is displayed at the bottom of the Acquire Data window. The filename itself will have the format yyyymmdd_hhmmss_openpet.dat, where the current date and time are used as the prefixes.

Example: 20140717_090251_openpet.dat
Data file stored on July 17, 2014 at 9:02:51AM

Once the parameters are set, you may click the Acquire Data button (in the bottom right corner of the Acquire Data window) to begin acquiring data.

6.3.4 Display Waveform

Once you have acquired data in oscilloscope mode (data mode = 0), you can display the waveforms event-by-event by clicking on the Display Waveform button (in the upper right corner) to show the Display Waveform window (Figure 52). The purpose of this window is the same as the displayWaveform macro in Section 6.2.2. The window is arranged vertically in four main sections. The top section is the user input settings, which are required to analyze the data file. The next section is the data file section, which includes the button to load a data file and a field that will display the name of the current data file open. Next is the canvas where the waveforms will be displayed. At the bottom is the event section, which shows the address and event number of the waveform currently displayed, along with the buttons to cycle to the next event or to stop and close the data file.

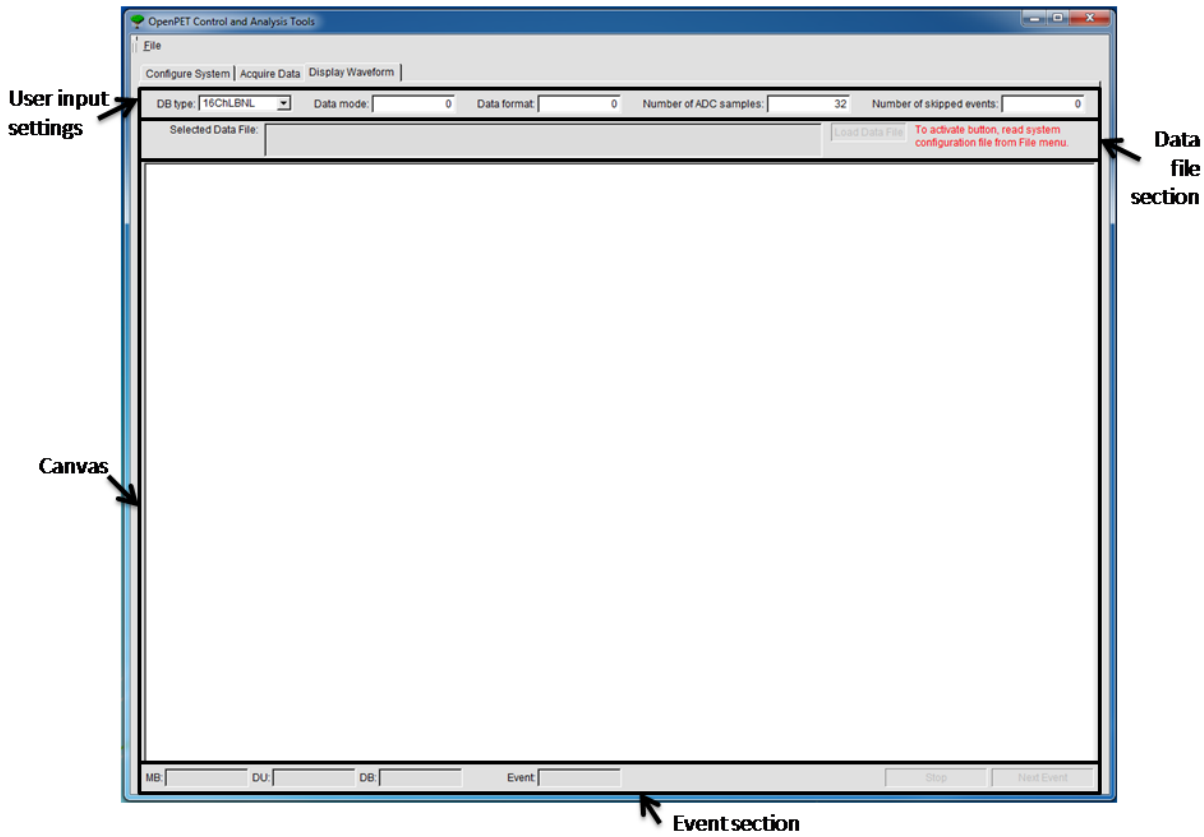


Figure 52: Initial Display Waveform window.

To activate the Load Data File button, you must first configure the system with the same configuration used to acquire the data file. If a different system configuration file has already been loaded, you can read in the correct system configuration file to match the configuration used with that data file. In either case, the steps to load the system configuration file are equivalent to the instructions in Section 6.3.2 and can be executed while in the Display Waveform window.

After activating the Load Data File button, you must check that the user input settings are correct before loading a data file. The values stored in that upper header are as follows:

- DB Type – the type of Detector Board (default is 16ChLBNL Detector Board)
- Data Mode – see command ID 0x2201 (default is zero)
- Data Format – see command ID 0x2203 for configuring oscilloscope mode (default is zero)
- Number of ADC samples – 1 to 241; see command ID 0x2203 (default is 32)
- Number of skipped events – number of events to be skipped (default is zero)

Once these settings are correct, you can then load the data file by clicking on the Load Data File button. A file dialogue will appear. Select the desired file and open it. Once opened, the first event waveforms will appear in the canvas, and the address and event number will appear on the bottom. The Next Event and Stop control buttons will also activate at this time.

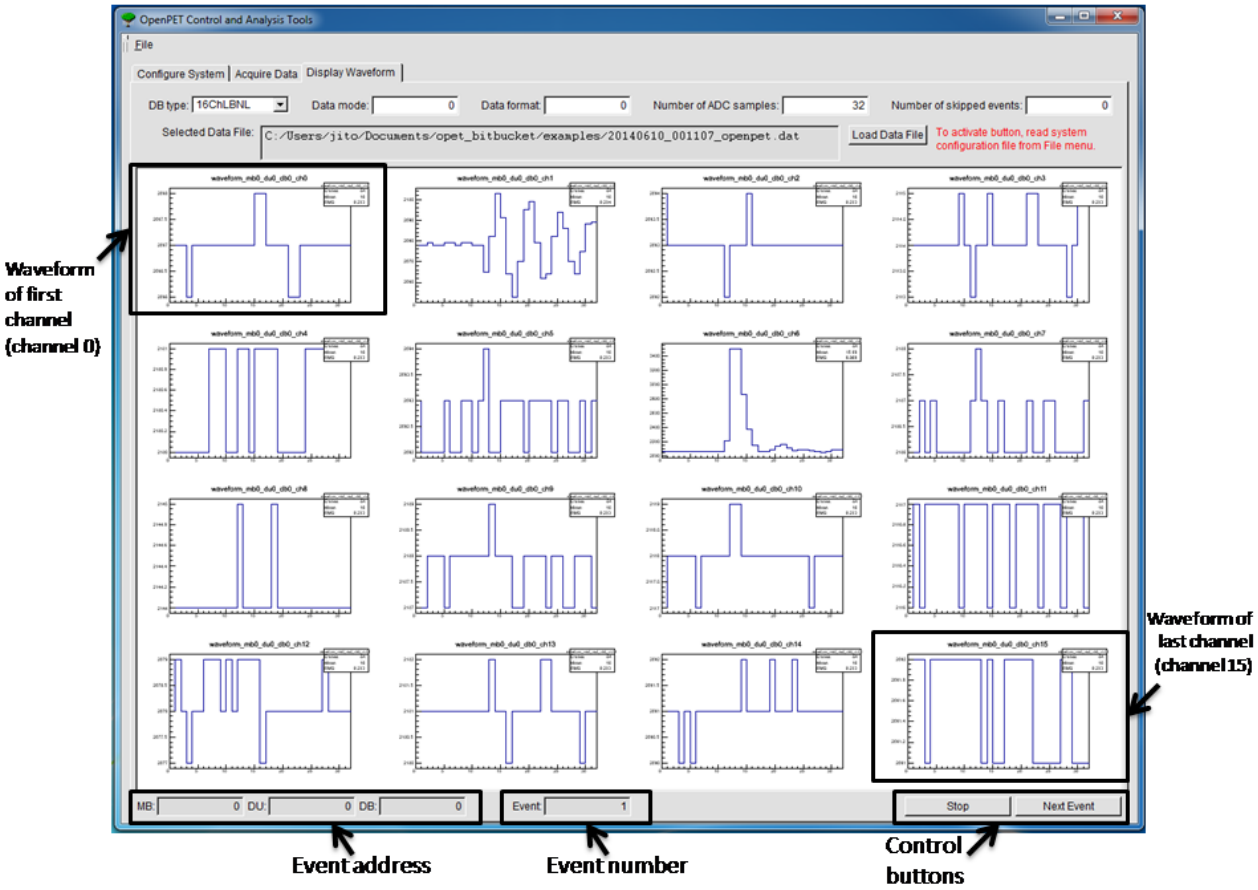


Figure 53: Display Waveform.

The waveforms are displayed by channel from left to right, starting in the top left corner. As shown in Figure 53, the top left waveform is channel 0, the next one to the right is channel 1, and so on with the last channel on the bottom right corner. The address (MB, DU, DB) of the event currently displayed is shown on the bottom left along with the current event number. On the bottom right are the control buttons. Clicking on the Next Event button will display the next event waveforms. The Stop button will close the data file and deactivate the control buttons until another data file is loaded.

In Figure 53, the number of skipped events is set to the default zero. However, if you do not want to start at the beginning of the file, you may skip to a specific event number. For example, if you want to skip to event number 10, you would input ten as the desired number of skipped events and load the data file again. The opening waveforms would then display event ten as shown below in Figure 54.

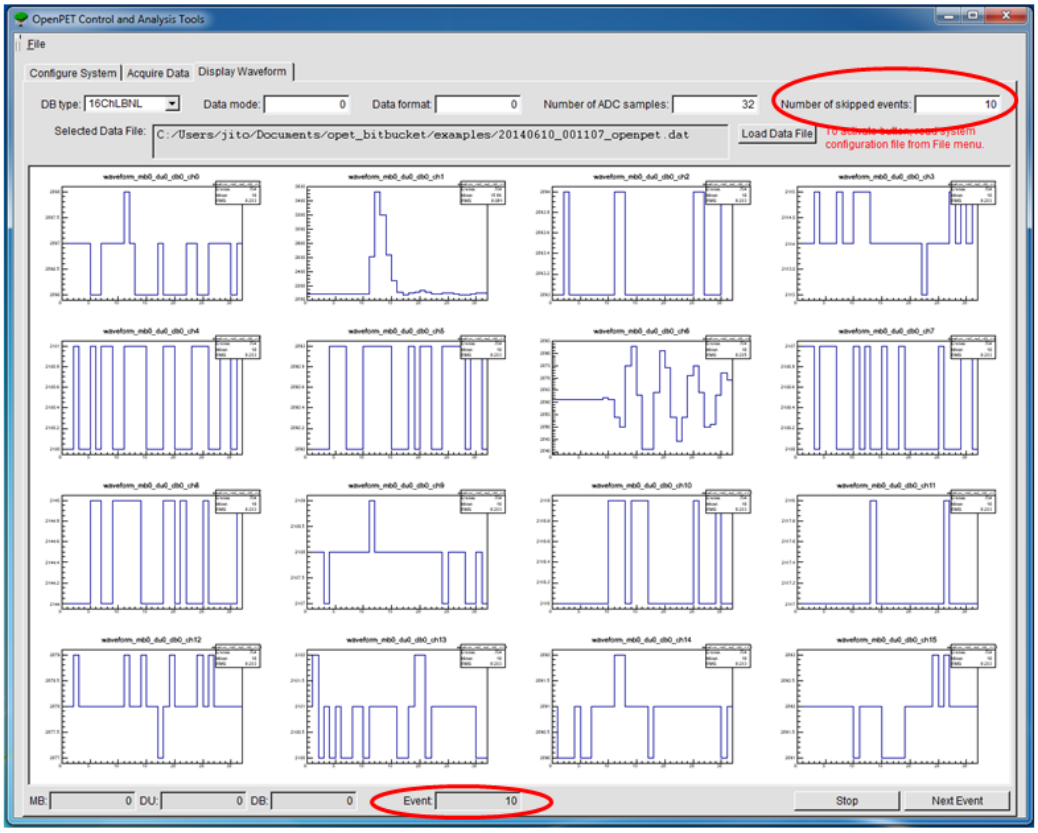


Figure 54: Example highlighting number of skipped events.

7 Acknowledgements

Initial funding for the LBNL portion of this work was supported by the Director, Office of Science, Office of Biological and Environmental Research, Biological Systems Science Division of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Subsequent work is supported by the National Institutes of Health of the US Department of Health and Human Services under grant R01 EB016104.

Reference to a company or product name does not imply approval or recommendation by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

8 Index

16-channel Detector Board 43
Altera Tools 23
Coincidence Unit 9
Coincidence Unit Controller 14
Commands 51
Data Acquisition 31
Data Format 33
Detector Board 40
Detector Unit 8
Detector Unit Controller 14
Downloading Software & Firmware 20
Firmware and Software Structures 13
Getting Started 18
Installing OpenPET Firmware & Software 25
Large System 11
OpenPET CAT 61
QuickUSB 20
Small System 10
Standard System 10
Support Board 47
Support Crate 8
USB-Blaster 23

9 Appendices

9.1 Appendix 1: 16-Channel Detector Board Default Values

As described in Section 5, commands with command id 0x2401 and 0x2403 initialize the DACs and ADCs respectively with default values. After the Support Crate power is turned on, the entire system is also booted up using default values.

These default values are specified as follows:

FPGA firmware settings = 0x00000000	Default (not previous settings)
System data mode = 0x00000000	Oscilloscope mode
Oscilloscope data settings = 0x00000000	ADC+TDC data format; 32 ADC samples per channel
Trigger mask (channels 0-15) = 0xFFFF	Trigger enabled for channels 0-15
Trigger mask (channels 16-31) = 0xFFFF	Trigger enabled for channels 16-31

Register	Name / Function	Default Value
0F	Power Down Configuration	No software power down. Allow complete external hardware power down.
11	LDVS Current Drive Configuration	3.5 mA drive for data bit clock, frame clock and data
12	LVDS Internal Termination Configuration	Internal termination is disabled
14	Low Frequency Noise Suppression	Low frequency noise suppression is disabled
24	Analog Input Inversion	Analog input inversion disabled
25, 26, 27, 45	Test Pattern Configuration 25/26/27/45	All 4 registers need to be combined to output a test pattern. All test patterns are disabled. Test patterns available: 1) Ramp: repeating full-scale ramp pattern 2) User Defined Fixed Constant: CUSTOM_PATTERN1_BIT<9:0> 3) Toggle Between 2 Fixed Constants: CUSTOM_PATTERN1_BIT<9:0> and CUSTOM_PATTERN2_BIT<9:0> 4) Deskew: output 010101010101 5) Sync: output 111111000000
2A	Analog Gain Configuration, Ch 1-4	0 dB
2B	Analog Gain Configuration, Ch 5-8	0 dB
42	Clock & Reference Voltage Mode	DDR_BIT_CLOCK_PHASE_MODE1 (90 degrees phase shift)
46	Data Format Configuration	Straight offset binary mode, LSB first

Table 5: Summary of 16-channel Detector Board ADC default values.

Name / Function	Default Value
DEFAULT_ENERGY_DAC_VALUE	0.3
DEFAULT_TIMING_DAC_VALUE	0.3
DEFAULT_RAMP_DAC_VALUE	0.0

Table 6: Summary of 16-channel Detector Board DAC default values.

9.2 Appendix 2: Troubleshooting Diagnostics

Using the command sequence listed in Section 2.4.3 is a simple way to test if your OpenPET system has been properly setup to take data. However, if you are not properly acquiring data, further diagnosis will be necessary. This Appendix outlines three basic steps for troubleshooting your OpenPET system. Data analysis for these tests can be performed with an optional Matlab program called Call_Analyze_OscTestData.m, which is available on the OpenPET website at http://openpet.lbl.gov/downloads/firmware-software/. You can also develop your own analysis software or use the OpenPET Control and Analysis optional software tool (see Section 6).

9.2.1 Test Digital Communication Chain

The first step in troubleshooting is to test if the basic Support Board and Detector Board digital command and data communication chain is functioning properly. This test does not require a signal input, since the Detector Board generates the counter data input internally.

For this diagnostic test, you configure the CDUC and all connected Detector Boards with the Oscilliscope mode “test communication data format” (address 0010) – mentioned previously in Section 2.4.3 and detailed in Section 5. This differs from standard data acquisition that uses the “ADC plus TDC data format" (address 0000).

First, you need to setup the OpenPET hardware and install the firmware and software by following the instructions detailed in Section 2. You can then configure the system and acquire test communication data using the basic OpenPET command sequence shown in Section 2.4.3. For convenience, an example command sequence for a single Detector Board in slot 3 is shown below:

- opet_cmd_usb 0x2200, 0x4000, 0x00000000, 3000, 0
  Boot-up DBs: loads the FPGA firmware with default values to all connected Detector Boards.
  Correct response: 0xa200, 0x4000, 0x0

- opet_cmd_usb 0x2403, 0x4000, 0x30000000, 1000, 0
  Initializes all the ADC registers with default values for the Detector Board in slot 3.
  Correct response: 0xa403, 0x4000, 0x30000000

- opet_cmd_usb 0x2404, 0x4000, 0x30000106, 1000, 0
  Sets the ADC gain to 6 dB for all ADCs on the Detector Board in slot 3.
  Correct response: 0xa404, 0x4000, 0x30000106

- opet_cmd_usb 0x2401, 0x4000, 0x30000000, 200, 0
  Initializes all the DAC registers with default values for the Detector Board in slot 3.
  Correct response: 0xa401, 0x4000, 0x30000000

- opet_cmd_usb 0x2402, 0x4000, 0x30602150, 200, 0
  Sets all DACs to +2.150 V for all channels on the Detector Board in slot 3.
  Correct response: 0xa402, 0x4000, 0x30602150

- opet_cmd_usb 0x2205, 0x4000, 0x3000FFFF, 200, 0
  Enables the trigger for channels 0-15 for the Detector Board in slot 3.

- opet_cmd_usb 0x2201, 0x4000, 0x00000000, 200, 0
  Configures the system data mode to Oscilloscope mode for all connected Detector Boards.
  Correct response: 0xa202, 0x4000, 0x00000000

- opet_cmd_usb 0x2203, 0x4000, 0x020000F0, 200, 0
  Configures the CDUC and all connected Detector Boards with the Oscilloscope mode settings of test communication data format and 240 raw ADC samples per channel.
  Correct response: 0xa203, 0x4000, 0x020000F0

- opet_cmd_usb 0x2209, 0x4000, 0x00F0F0F0, 200, 0
  Clears the event FIFO in the CDUC and all connected Detector Boards to remove data from previous events.
  Correct Response: 0xa209, 0x4000, 0x00F0F0F0

- opet_acq 5, TestCom.dat, 0
  Acquires data from the CDUC through the USB module 0 for 5 seconds and saves the data to file TestCom_slot3.dat.

However, if you use the optional Matlab code provided on the OpenPET website, then issuing the command sequence above is not necessary. The Matlab program "Call_Analyze_OscTestData.m" (which calls Matlab program Analyze_OscTestData.m) configures the system, acquires the test data, analyzes it, and automatically generates diagnostic plots (described below).

The program Call_Analyze_OscTestData.m is not entirely user friendly, so you need to edit the file by hand in the following ways before running the program:

- test_pattern=1 (Tests the digital communication chain, using Osc. mode test communication data format)
- slot_index=3 (Sets Detector Board slot number to be tested. Must test one Detector Board at a time.)
- filename_all={'TestCom_slot3.dat 0'} (Sets name of data file for Analyze_OSCTestData.m subroutine.)
- system{'opet_acq_usb 5 TestCom_slot3.dat'}; (Sets time and data filename in acq command.)

The program automatically generates two diagnostic plots for this test. The first plot shows the length of a single data train as a function of the event index. For example, a 16-channel Detector Board with 240 raw ADC samples per channel has a data train length of 15436 bytes = 4 bytes/word x {2 Starting words + 1 Ending word + 16 DB channels x (240 ADC samples + 1 TDC sample)}, as shown in Figure 55. This plot can be used to test the data stability. It should be a flat line, indicating that no data bits have been dropped.

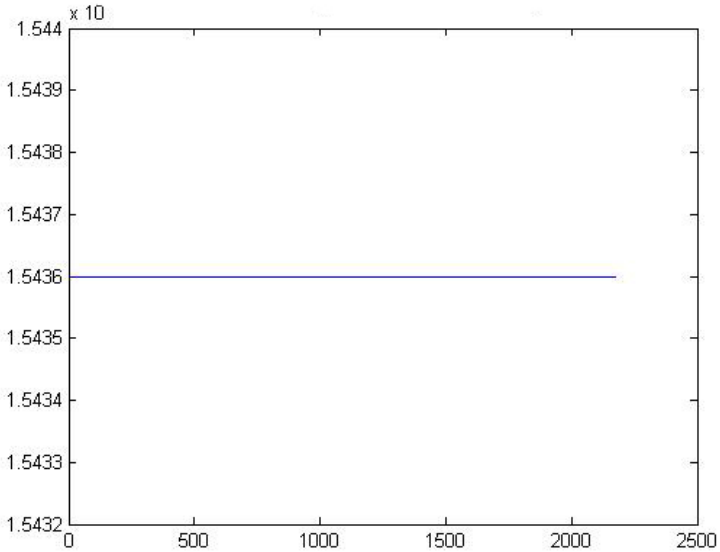


Figure 55: The event index (x-axis) as a function of the length of a data train in bytes (y-axis). Data were acquired for 5 seconds using 240 raw ADC samples per channel and test communication data format.

The second plot shows the waveform for the internally generated counter data of the 600th data train for all 16 Detector Board channels with 240 ADC samples/channel. Specifically, the ADC sample index (x-axis) is plotted as a function of the ADC value (y-axis) for each channel. The counter values increment from 0 to N-1, the ADC sample indices increment from 1 to 239, and the plot for each channel should show a straight line with a slope of 1. The plot on the upper left corner corresponds to channel 0, upper right corner to channel 3, and bottom right corner to channel 15. This plot can be used to test if the Support Board and Detector Board digital command and data communication chain is functioning properly.

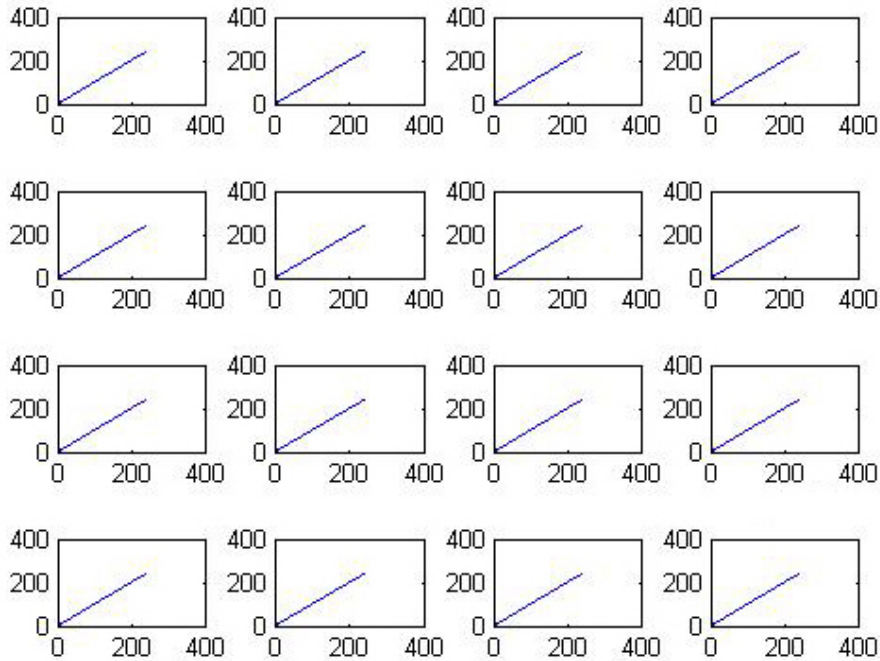


Figure 56: Waveforms for internally generated counter data, shown for 600th data train and all 16 DB channels with 240 ADC samples/channel. For each channel, the ADC sample index (x-axis) is plotted as a function of the ADC value (y-axis).

9.2.2 Test Analog with Internal Trigger

If the first test indicates that the digital command and data communication chain is functioning properly, then the next step in troubleshooting is to test the analog circuitry using an input signal triggered by an internal clock. First, you need to attach a custom adapter board on the front of the Detector Board (or skywire) to input an analog signal. You can then input a signal (such as a sine wave) from a signal generator. Although it is more time consuming, you should test each channel of each Detector Board separately.

Next, you should configure the system and acquire data using a command sequence similar to that shown in Section 9.2.1. However, you need to configure the system for this test using the Oscilloscope mode “test analog data format” (address 0011) and use a different filename, as shown below:

- opet_cmd_usb 0x2203, 0x4000, 0x030000F0, 200, 0
  Configures the CDUC and all connected Detector Boards with the Oscilloscope mode settings of test analog data format and 240 raw ADC samples per channel.
  Correct response: 0xa203, 0x4000, 0x030000F0

- opet_acq 5, TestADC.dat, 0
  Acquires data from the CDUC through the USB module 0 for 5 seconds and saves the data to file TestAnalog_slot3.dat.

If you use the optional Matlab code provided on the OpenPET website, then issuing the command sequence is not necessary. You instead need to edit the file by hand in the following ways before running the program:

- test_pattern=2 (Tests the analog signals, using Oscilloscope mode test analog data format)
- slot_index=3 (Sets Detector Board slot number to be tested. Must test one Detector Board at a time.)
- filename_all={'TestAnalog_slot3.dat'} (Sets name of data file for Analyze_OSCTestData.m subroutine.)
- system{'opet_acq_usb 5 TestAnalog_slot3.dat'}; (Sets time and data filename in acq command.)

The program automatically generates two diagnostic plots for this test. The first plot shows the length of a single data train as a function of the event index, as described in Section 9.2.1, which can be used to test the data stability (see Figure 55).

The second plot shows the waveform of the input signal for all 16 channels of the Detector Board. For instance, Figure 57 shows the waveform plots when a signal generator was used to input a sinewave into Channel 0 only and data were acquired with 240 raw ADC samples per channel and test analog data format. This plot can be used test if you can get output an analog signal without signal shape deformation or saturation and an acceptable baseline. In addition, this test can be used to investigate cross talk between channels.

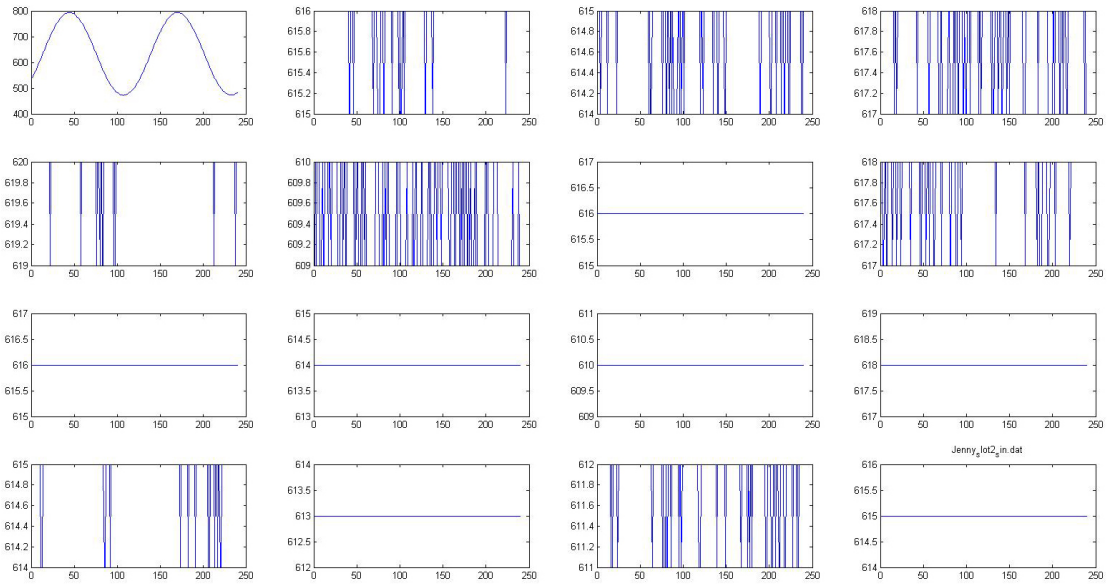


Figure 57: Waveforms for all 16 DB channels, when a sinewave was inputted into only channel 0 and data were acquired with 240 raw ADC samples/channel, test analog data format, and internal clock trigger. For each channel, the sample number index (x-axis) is plotted as a function of amplitude (y-axis).

9.2.3 Test Trigger

If you have confirmed that your digital communication chain and analog circuitry are functioning properly with the two tests described above, then you need to check your detector signals and trigger. Specifically, you should check your trigger mask and DAC settings to make sure that your trigger thresholds are set properly for the input signals from your detector module – see Section 5 for more details on how to set and read the trigger mask.