**Fermi National Accelerator Lab**

# BLM Upgrade Control Card Program Design

Randy Thurman-Keup

**AD / Instrumentation Department**

February 12, 2018

BEAMS-DOC-2433-V12

CVS Code Version v18

*Abstract*

This document details the structure and behavior of the program that runs in the eZ80 on the Controller Card. The main functionality of this program is to start and stop data acquisition and to maintain data buffers that are accessible from VME at any time without disrupting the aborting capability. These data buffers consist of three circular buffers containing summed data at short, medium, and long integration times, two linear buffers housing profile and flash frames, and a single entry buffer for the most recent display frame.

# 1 Introduction

The Controller Card (CC) is an embedded processor (eZ80) board residing in the VME crate of the Beam Loss Monitor (BLM) system [1]. The program's job is to transfer information between the crate processor and the digitizer cards (DC), timing card (TC), and abort card (AC), in a deadtimeless fashion, *i.e.* no interruption in the aborting capability of the system. The program operates with a combination of polling and interrupts.

The Controller Card Program (CCP) responds to 5 types of events.

- **TCLK**
  The TC puts relevant TCLK events into its FIFO. The CCP polls the TCLK FIFO status register to determine if there is data in the FIFO. If there is data, it reads it from the FIFO and handles it.
- **MDAT**
  When the TC receives the relevant MDAT frame, it writes the state information to the MDAT FIFO. The CCP polls the MDAT FIFO status register to determine if there is data in the FIFO. If there is data, it reads if from the FIFO and handles it.
- **DATA_LATCH (*Interrupt*)**
  This is an interrupt generated by the Timing Card when it is time to latch some flavor of Digitizer Card data (fast, slow, or very slow sums). The CCP must then read the status bytes in the Timing Card to determine which data to latch.
- **Abort_Service**
  The abort card generates an interrupt when one of three user selectable states occurs.
    - A digitizer card indicates that a channel is not OK
    - A digitizer channel is over one of the thresholds but the multiplicity requirement is not met for an abort
    - An abort has occurred

  The CCP does not respond to the interrupt. Instead it periodically polls the AC to determine the above information.
- **Crate Processor**
  When settings need to be updated, the CP writes the settings to the CC and sets an appropriate register in the CC which then responds by loading the settings into the appropriate cards at the appropriate time.

The CCP might need to do intelligent checking of the state of cards and possibly issue a Reset.

# 2 Initialization

At boot time, the CP and CC must handshake to properly bring up the system. This handshaking between CC and CP is documented in Figure 1 below. The initialization procedure includes stopping the CC at the beginning of the procedure. Stopping the CC at boot time is necessary for a number of reasons: first, at crate power up time, the other cards will not have gone through their FPGA programming sequence if the CC immediately starts to access them; second, the waiting gives the CP time to download settings to the CC; and third, if the CC reboots by, e.g. hitting an invalid instruction, it would probably be good to notify the outside world, and save a snapshot of the CC's debug memory contents for later analysis. The settings that need to be downloaded to the CC are listed in Table 1.

- **Timing Card**
  The timing card must be downloaded with the appropriate TCLK events (see Table 4). In addition to the events listed in Table 4, there are several other TCLK and BSCLK events that the CCP does not respond to but which are responded to by the other cards in the system.
  - TCLK $8F – 1 Hz event for updating the TC clock
  - TCLK $5B – TeV Pbar Injection TBT trigger
  - TCLK $5C – TeV Proton Injection TBT trigger
  - TCLK $28 – MI Injection TBT trigger
  - BSCLK $AA – Revolution marker used to generate Make_Meas clock
  - BSCLK $DA – MI and TEV Studies TBT trigger
- **Digitizer Card**
  The digitizer card must be setup for the correct memory map. The correct map is enabled by setting bit 7 of address 0xFF to 1. All other settings are part of the CP download.
- **Abort Card**
  The abort card settings are all downloaded as part of the CP download.

The settings that are loaded vary with machine type. Presently there are 7 machine types: Tevatron, Main Injector, Nova, Switchyard, Muon Campus, Switchyard/Muon combination, and Muon g-2. Nova is special in that it does the integration sums in software in the CC to try to separate out RR losses from MI losses. Muon g-2 is special in that it is handling PMTs in which the signals are negative-going. Thus the hardware integration doesn't work, since there is no way to adjust the pedestal without modifying the firmware.

After successfully initializing, the CCP continuously polls for events, periodically interrupted by the Data Latch interrupt.

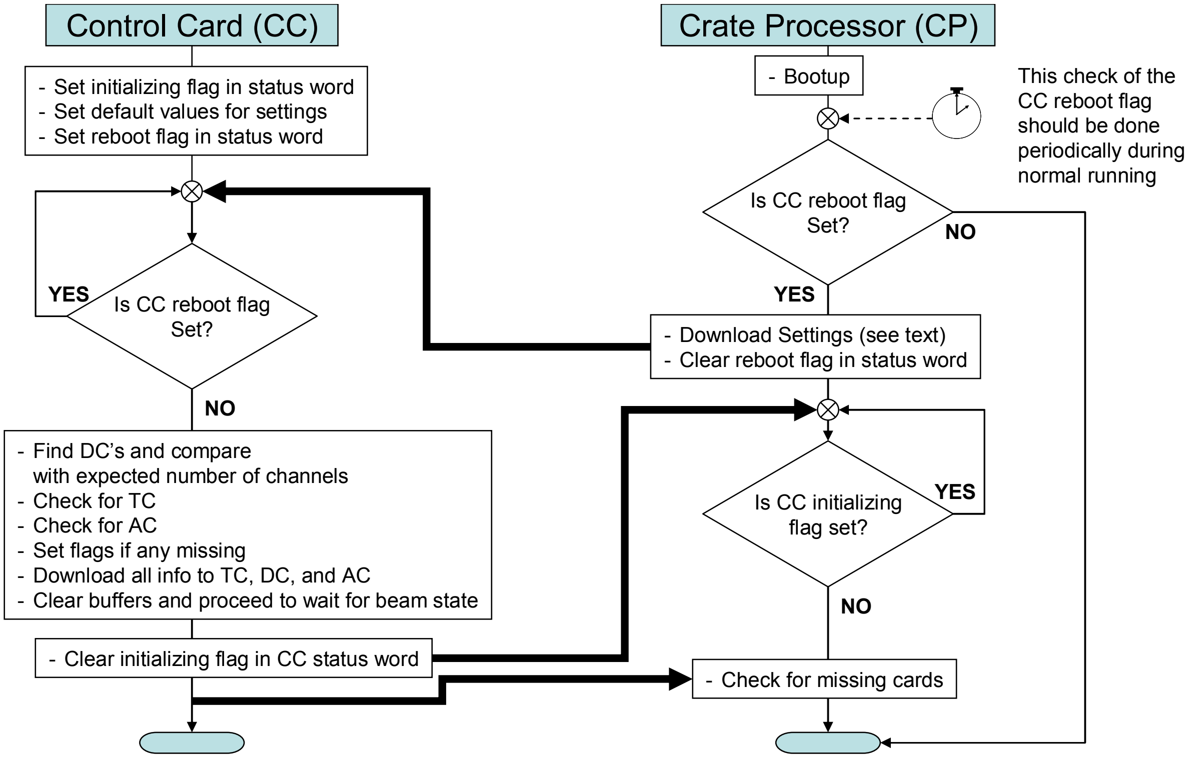

Control Card (CC)
- Set initializing flag in status word
- Set default values for settings
- Set reboot flag in status word
YES
Is CC reboot flag Set?
NO
- Find DC's and compare with expected number of channels
- Check for TC
- Check for AC
- Set flags if any missing
- Download all info to TC, DC, and AC
- Clear buffers and proceed to wait for beam state
- Clear initializing flag in CC status word
Crate Processor (CP)
- Bootup
This check of the CC reboot flag should be done periodically during normal running
Is CC reboot flag Set?
NO
YES
- Download Settings (see text)
- Clear reboot flag in status word
Is CC initializing flag set?
YES
NO
- Check for missing cards

**Figure 1: Flowchart of handshaking that occurs at boot time. The alarm clock symbol indicates that the CP should periodically check the reboot bit in the CC and if it finds it set, it should probably notify the outside world somehow. See Section 0 for a description of the flags in the status word.**

**Table 1: Settings that need to be set on the CC by the CP.**

| Address | Size | Setting | Description | Tevatron | Main Injector | Nova | Switchyard | Muon Campus | Muon & SWYD | Muon G-2 |
|---|---|---|---|---|---|---|---|---|---|---|
| 000004 | 2 | Derippled or Fast sum | Controls whether the Flash/Profile/Display frames contain Fast sums (0) or Derippled sums (1) (for those that use Fast as opposed to Slow) | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 000006 | 2 | Flash, Display, Profile source | This determines the source of the first half of the frame (0=fast or 1=slow). Bit 0 is for Flash, bit 1 is for Profile, and bit 2 is for Display. | 0x0006 | 0x0006 | 0x0006 | 0x0006 | 0x0006 | 0x0006 | 0x0006 |
| 000014 | 4 | System Time | Unix time in seconds since ??? (Little Endian word order) | --- | --- | --- | --- | --- | --- | --- |
| 00001C | 2 | Machine | Which machine is this | 1 | 2 | 4 | 3 | 5 | 6 | 7 |
| 000090 | 2 | Initial state | Initial MDAT machine state | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0001 | 0x0000 | 0x0001 |
| 000100 | 2 | # of Channels | Expected number of channels | --- | --- | --- | --- | --- | --- | --- |
| 000102 | 2 | Make Measure Div | Amount to divide down the clock by (int. osc.) | 0x0001 | 0x0002 | 0x0002 | 0x0013 | 0x0013 | 0x0013 | 0x0013 |
| 000104 | 2 | Fast Sum Length | # of measurements to accumulate in DC fast sum. | 64 | 64 | 141 | 64 | 64 | 64 | 64 |
| 000106 | 2 | Slow Sum Length | # of measurements to accumulate in DC slow sum. | 1769 | 1769 | 1769 | 1769 | 1769 | 1769 | 1769 |
| 000108 | 2 | Very Slow Sum Length | # of measurements to accumulate in DC very slow sum. (In MI this is the integral and must be 1/16 the pedestal length) | 50000 | 47 | 141 | 47 | 47 | 47 | 47 |
| 00010A | 2 | DC FPGA Control Register | Funny name for something which contains the number of make_meas to skip before doing pedestals divided by 16 ( Nskip / 16 ) and the derippling down conversion offset divided by 32 (see the User's Guide for details) | 0x10CC | 0x1000 | 0x1000 | 0x1000 | 0x1000 | 0x1000 | 0x1000 |
| 00010C | 2 | DC Test DAC Value | DAC Value used for pedestal in DCs modified for PMT use | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x7B00 |
| 00010E | 2 | TC Operation mode | Value to be written to TC control bus CSR. Controls whether TC uses AA marker or internal oscillator. | 0x0004 | 0x0000 | 0x0000 | 0x0003 | 0x0003 | 0x0003 | 0x0003 |
| 000112 | 2 | IRQ3 Enable Bits | 3 bits that determine what generates an IRQ3 interrupt on the AC | 0x0007 | 0x0007 | 0x0007 | 0x0007 | 0x0007 | 0x0007 | 0x0007 |
| 000114 | 2 | Abort Enable | Bit 0 Enables the functioning of the AC, Bit 4 requires two consecutive make_meas cycles with an abort before actually pulling the abort, Bit 2 controls inverting thresh. | 0x0011 | 0x0011 | 0x0011 | 0x0011 | 0x0011 | 0x0011 | 0x0015 |
| 000116 | 2 | Pedestal Length | Must be 16 * Very Slow Sum Length for machines which do integrations | 795 | 752 | 2256 | 752 | 752 | 752 | 752 |
| 000118 | 2 | End of beam delay | Delay (in Fast latch periods) after end-of-beam event before asserting AIP | 0x0012 | 0x0012 | 0x0012 | 0x0012 | 0x0012 | 0x0012 | 0x0012 |
| 00011A | 2 | Flash Delay | Delay from flash frame clock event until data is grabbed from buffer | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 00011C | 2 | Profile Delay | Delay from profile frame clock event until data is grabbed from buffer | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 00011E | 2 | Display Delay | Delay from display frame clock event until data is grabbed from buffer | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 000120 | 2 | Input Switch state for peds | Whether (1) or not (0) to disable the inputs while taking pedestals | 0x0001 | 0x0001 | 0x0001 | 0x0001 | 0x0001 | 0x0001 | 0x0000 |
| 000126 | 2 | Delay after F sector end TCLK event | Delay (slow sum units) after the end TCLK event before re-enabling normal F sector aborts | 0x0002 | 0x0002 | 0x0002 | 0x0002 | 0x0002 | 0x0002 | 0x0002 |
| 000128 | 2 | Max DY | Maximum difference between successive CIC sums to replace baseline waveform (Deripple parameter) | 256 | 256 | 256 | 256 | 256 | 256 | 256 |
| 00012A | 2 | CIC Sum Length | # of measurements to accumulate in the CIC sum (Deripple parameter) | 128 | 128 | 128 | 128 | 128 | 128 | 128 |
| 00012C | 2 | Pedestal Clock Event | Clock event to stop and start make meas to produce a pedestal (only used for NOVA) | 0x00 | 0x00 | 0xFE | 0x00 | 0x00 | 0x00 | 0x00 |
| 00012E | 2 | Pedestal Delay | Delay after receiving the pedestal clock event before executing the pedestal measurement | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 000130 | 2 | Pedestal Period | Maximum time allowed between pedestal clock events | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF |
| 000200 | 2 per chan | DC Mode Selects | Digitizer channel mode select word (one word / channel). Controls integration mode and whether squelch is enabled among other things. | 0x0002 | 0x000A | 0x0002 | 0x000A | 0x000A | 0x000A | 0x0000 |
| 000202 | 2 per chan | DC Manual Set Value | The manual setting value if manual setting mode is chosen | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 000400 | 120 | Squelch Values | This must be in the appropriate units which is $X\,(\sigma_{raw}\sqrt{16\times 17\times n_{vs}})$ where X is the # of sigma to place the squelch | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 0E0000 | 256 | MDAT to Abort State map | Maps MDAT state to abort state | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… | 0 --> 0, 1 --> 1, etc… |
| 0E0100 | 256 | F sector Start TCLKs | Starting MI TCLKs for F sector aborts | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |
| 0E0200 | 256 | F sector End TCLKs | Ending MI TCLKs for F sector aborts | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |
| 0F0000 | 64K | F sector actions | See Text about F sector aborts | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF | 0xFFFF |
| 100000 | 256K | Abort Info | Thresholds, Masks, and Multiplicities for all States | Thresholds 0xFF, Masks 0x00, Mult 0xFF, Crate Mask should be 0xFF | | | | | | |

# 3 Program Components

## 3.1 CC VME Status Register

The status register consists of a 16-bit status word in VME memory and a 16-bit extended status word also in VME memory. Tables 2 and 3 show the bit definitions of the status words.

**Table 2: Bit definitions in the CC Status word.**

| Bit | Description |
|---|---|
| 15 | Running |
| 14 | ERROR line on backplane has been asserted (latched until CC reboot) |
| 13 | DeRippled Buffer has wrapped |
| 12 | Mismatched raw data pointers in TC, DC, or AC |
| 11 | Pedestals Valid; Don't read pedestals until this is set |
| 10 | Very Slow Buffer has wrapped |
| 9 | Slow Buffer has wrapped |
| 8 | Fast Buffer has wrapped |
| 7 | Wrong number of Digitizer Card channels found |
| 6 | Abort Card not found |
| 5 | Timing Card not found |
| 4 | Crate has triggered an Abort |
| 3 | Some channels are indicating abort |
| 2 | Some channels are not OK |
| 1 | CCP is in the process of initializing |
| 0 | The CC has rebooted; clearing this kicks off the CC |

**Table 3: Bit definitions in the CC Extended Status word.**

| Bit | Description |
|---|---|
| 15 | Unused |
| 14 | Unused |
| 13 | Unused |
| 12 | Unused |
| 11 | Unused |
| 10 | Unused |
| 9 | RR Integrated abort |
| 8 | MI Integrated abort |
| 7 | RR Integrated channel abort |
| 6 | MI Integrated channel abort |
| 5 | RR Integrated Buffer has wrapped |
| 4 | MI Integrated Buffer has wrapped |
| 3 | RR Abort |
| 2 | MI Abort |
| 1 | CCRR Running |
| 0 | MI Running |

## 3.2 VME Accessible Circular Data Buffers

The data from all the digitizers is stored in VME memory in the form of circular buffers. There are index counters which indicate which frame is the current frame, and flags in the status register to indicate when each buffer has wrapped around. Each data frame header contains a flag byte which indicates a number of things. At end of beam, the last frame contains a 1 in the flag byte. This is nominally to allow the CP to ignore this frame at the beginning of the next beam cycle. The first frame of a cycle has a 2 in the flag byte to allow the CP to correct for a bug in the system whereby the DC waits to start summing, but the TC sends out latches immediately. The result is that the slower sums are incomplete when the first latch is received. The CP can divide by the actual sum length if it knows a frame is the first one of the cycle. The next frames contain a 3 in the flag byte until the input switch is closed. All other frames contain 0 in the flag byte.

There are 5 circular buffers on the CC.

- **Derippled Sum**
  This buffer contains the derippled version of the fast sum.
- **Fast Sum**
  This buffer is used in all machine types.
- **MI Integrated Sum**
  This buffer is only filled in the Nova machine type.
- **RR Integrated Sum**
  This buffer is only filled in the Nova machine type.
- **Slow Sum**
  This buffer is filled in all machine types.
- **Very Slow Sum**
  This buffer is filled in all machine types, but the data is different for the various machine types. In the Tevatron type, it contains very slow sum values. In the Nova type, it contains integrated values for continuous fixed-duration time periods. This allows a measurement of the total loss the tunnel. In the Muon g-2 machine type, it contains sum periods that overlap transfers from RR to the Delivery Ring. All other machine types contain the integrated values.

### 3.3 State Machines

The CCP contains one hardware and one or more software state machines. The hardware state machine keeps track of the state of the physical data acquisition. The software state machines handle the state of each accelerator in the system. For Nova, both the Recycler and Main Injector are operating simultaneously in the tunnel but with different starting and stopping points. The software state machines keep track of these differing starts and stops. Both kinds of state machines have the same states and state inputs, although some may be unused.

Figures 2 and 3 illustrate the hardware and software CCP state machines.

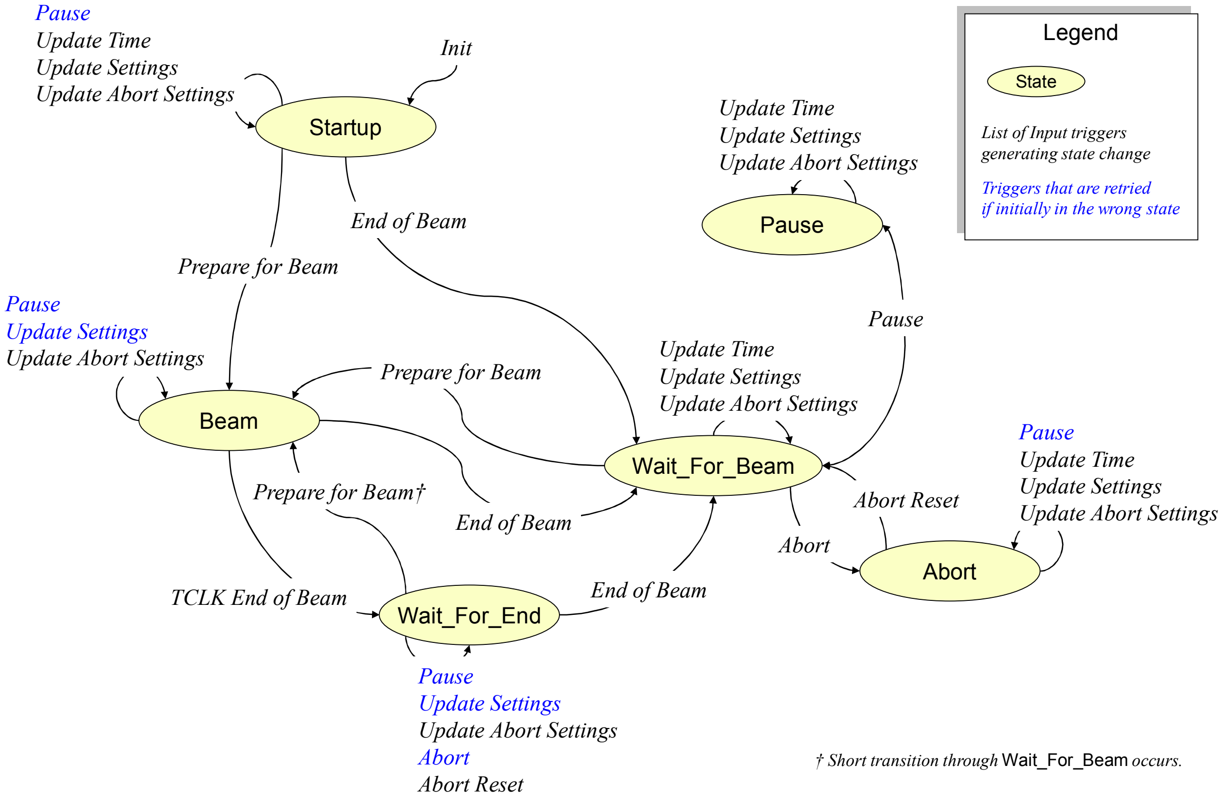


**Figure 2: Hardware State Machine diagram showing states and state change triggers. The triggers in blue are persistent triggers in that they hang around if they occur during an invalid state. When a valid state is reached, they are replayed.**

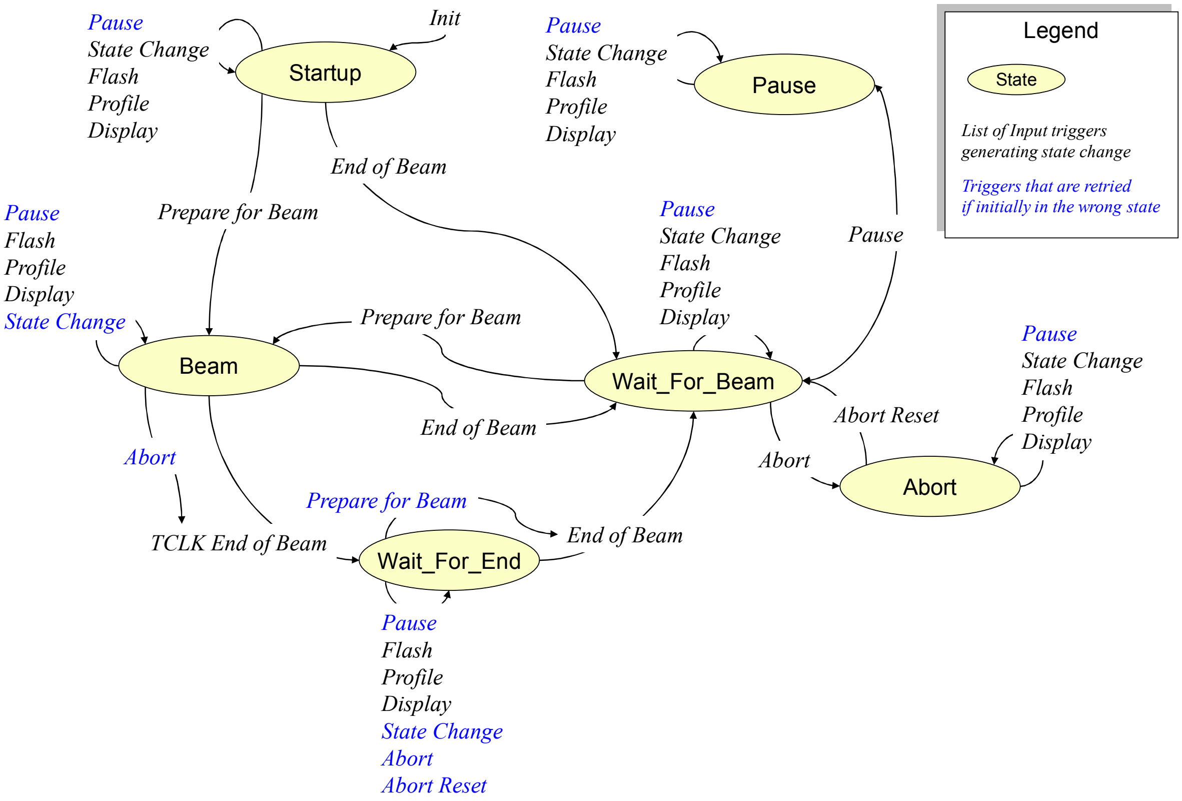


**Figure 3: Software State Machine diagram showing states and state change triggers. The triggers in blue are persistent triggers in that they hang around if they occur during an invalid state. When a valid state is reached, they are replayed.**

Figure 4 illustrates a typical beam cycle for a single accelerator type system (i.e. one hardware state machine and one software state machine).

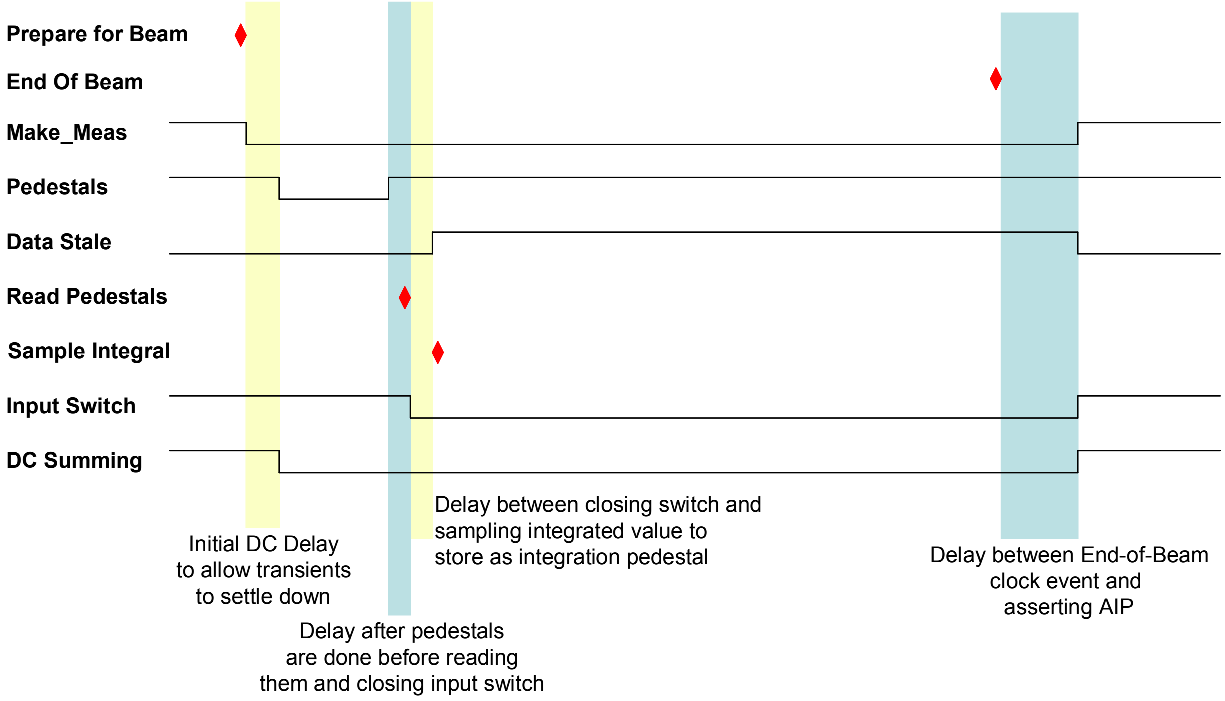

Prepare for Beam
End Of Beam
Make_Meas
Pedestals
Data Stale
Read Pedestals
Sample Integral
Input Switch
DC Summing
Initial DC Delay to allow transients to settle down
Delay after pedestals are done before reading them and closing input switch
Delay between closing switch and sampling integrated value to store as integration pedestal
Delay between End-of-Beam clock event and asserting AIP

**Figure 4: Beam cycle showing the relative timing of various events for a single accelerator. The shaded areas are delays between various key events. The integrated value, if it is used, must be sampled as a pedestal after closing the input switch, since there is a glitch upon closing the switch causing the integral to go negative by some amount. The lines are true low.**

Figure 5 illustrates NOVA multi-accelerator beam cycles.

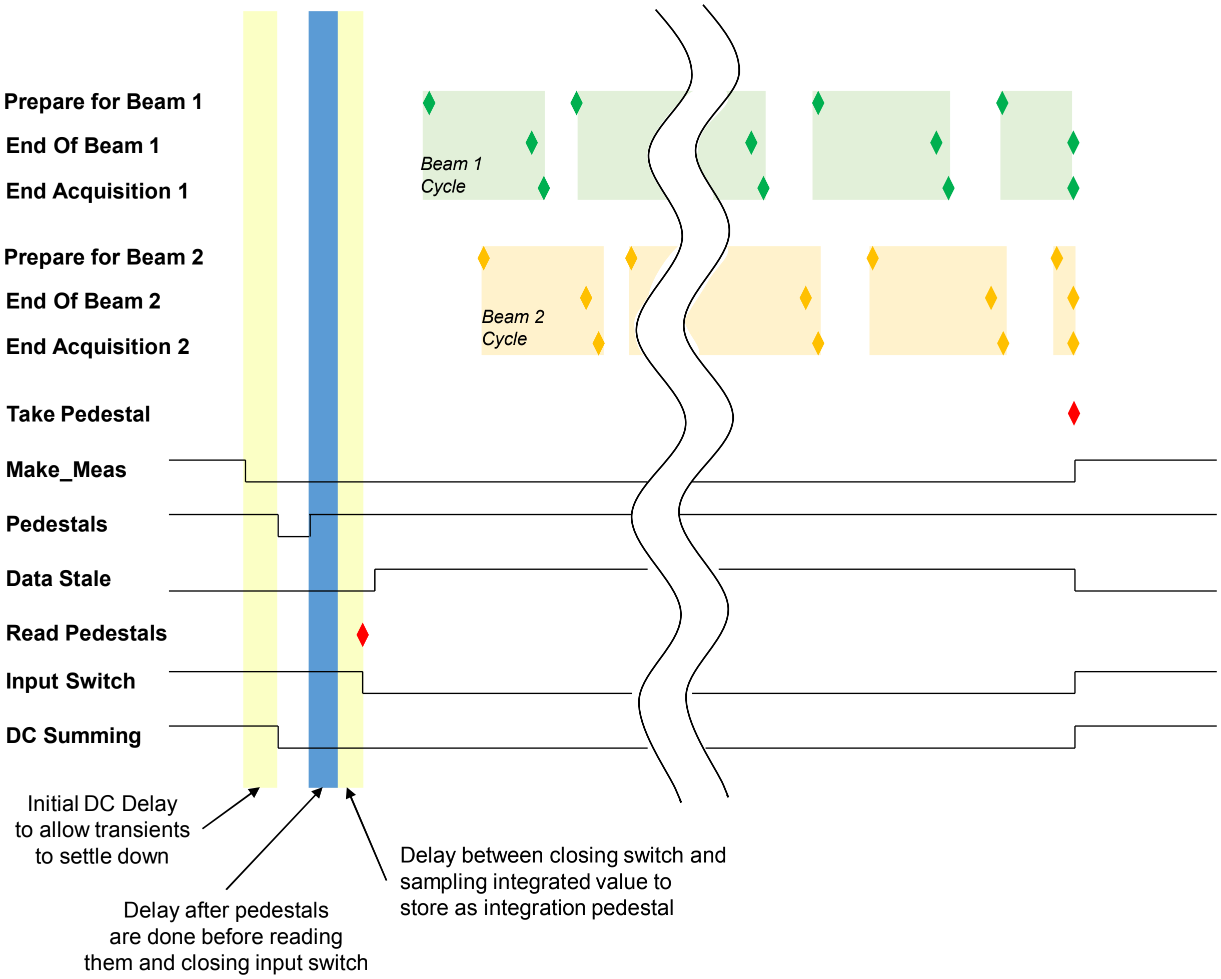

Prepare for Beam 1
End Of Beam 1
End Acquisition 1
Prepare for Beam 2
End Of Beam 2
End Acquisition 2
Take Pedestal
Make_Meas
Pedestals
Data Stale
Read Pedestals
Input Switch
DC Summing
Beam 1 Cycle
Beam 2 Cycle
Initial DC Delay to allow transients to settle down
Delay after pedestals are done before reading them and closing input switch
Delay between closing switch and sampling integrated value to store as integration pedestal

**Figure 5: Beam cycle showing the relative timing of various events for Nova featuring two accelerators. The shaded areas are delays between various key events. The lines are true low.**

## 3.4 Abort Information

Abort information is specified as a function of abort state. The CCP receives MDAT frames containing a state which is then mapped to abort state. This mapping is created by the console application user and sent to the CP which then copies the map to the CC as part of update abort settings.

### 3.4.1 Handling of Tevatron F Sector Aborts

*With the demise of the Tevatron, this section is no longer relevant, but is left here since the code is still in place.*

The abort handling in the Tevatron is more complicated than in the Main Injector. Since there is routinely beam traversing the p2 and p3 beam lines in F sector located directly above the Tevatron beamline, there may be false aborts caused by losses from those lines hitting the Tevatron BLMs. To reject these false aborts, the CCP uses a table of MI states and corresponding begin and end TCLK events during which to change the behavior of the aborts. The default behavior is to mask off the aborts at the AC. The actual implementation on the CCP is in the form of the list of begin / end TCLKs indexed by MI MDAT state, and a table of actions indexed by TEV abort state and MI MDAT state. Both the list of TCLKs and the action table are byte arrays. A TCLK entry of 0x00 means do nothing. The 2-D action table is arranged such that the MI MDAT state index increments the fastest. The actions are defined in the following table

| Action # | What it does |
|---|---|
| 0x00 | Globally mask off the aborts at the AC |
| 0xFF | Do nothing |
| 0xNN | Switch to TEV abort state 0xNN |

As stated above, the default is set to be 0x00 which masks off the crate aborts. The memory maps for these lists are given in Appendix B.

### 3.5 Debugging Information

There is a section in VME accessible memory that contains possibly useful information for debugging and error trapping. There are counts of various quantities like the number of TCLKs, the number of each type of data latch, etc… This list is in Appendix A and in the BLM Users' Guide [1]. In addition, there is a CP trigger to dump Control Bus memory to a section of VME memory.

There are debugging features built in to the code that are worth calling out here: a collection of buffers to record the CPU time used for a variety of events, and a buffer to record the time of arrival of TCLK's, MDAT's, end-of-beam, aborts, settings updates, and abort settings updates. These two data collections can be read out through VME. Each of these methods uses a separate programmable reload timer running off the system clock and set to a time resolution of ~5-6 µs. The latter is referred to as an event monitor in the code and makes use of an interrupt generated from the timer which then increments a counter to provide a longer period timestamp.

## 4 External Triggers

### 4.1 TCLK Event

The code responds to TCLK events in the form of polling the TC. A supported TCLK event causes the TC to put the TCLK number in the FIFO. The CCP polls the FIFO by checking the TCLK FIFO status register and reading the TCLK event. Table 4 documents the supported TCLK events.

**Table 4: List of CCP supported TCLK events, state inputs resulting from them, and a description of any action taken.**

| TCLK | State Input | Action |
|---|---|---|
| **Tevatron** | | |
| $71 | *Prepare for Beam* | Prepare for beam by issuing a Reset_DC, resetting the circular buffers, and clearing the abort in progress line. |
| $4B | *End of Beam* | End of beam. Start timer for delay before telling TC to assert Abort In Progress line. |
| $47 | *Abort* | End of beam, then goto abort state and wait for reset. |
| $48 | *Abort Reset* | Get out of abort state and wait for beam. |

| TCLK | State Input | Action |
|---|---|---|
| $77 | *Flash* | Create a flash frame. The current chosen sum and very slow sum buffer frames are appended to the flash frame buffer which contains a maximum of 256 flash frames and is cleared at Reset_DC. |
| $75 | *Profile* | Create a profile frame. The current chosen sum and very slow sum buffer frames are appended to the profile frame buffer which contains a maximum of 256 profile frames. Cleared at Reset_DC. |
| $76<br>$78 | *Display* | Create a display frame. The current chosen sum and very slow sum buffer frames are placed in the display frame. Only one display frame is allowed at a time. |
| $70 | *None* | Reset linear buffers for flash and profile frames |
| **Main Injector** | | |
| $A0 | *Prepare for Beam* | Prepare for beam by issuing a Reset_DC, resetting the circular buffers, and clearing the abort in progress line. In Nova, this only affects the software integrators. |
| $26 | *End of Beam* | End of beam. |
| $27 | *Abort* | End of beam, then goto abort state and wait for abort reset. |
| $24 | *Abort Reset* | Get out of abort state and wait for beam. |
| $7C | *Flash* | Create a flash frame. |
| $7A | *Profile* | Create a profile frame. |
| $7B | *Display* | Create a display frame. |
| **Switchyard** | | |
| $31 | *Prepare for Beam* | Prepare for beam by issuing a Reset_DC, resetting the circular buffers, and clearing the abort in progress line. |
| $36 | *End of Beam* | End of beam. |
| $3E | *Abort* | End of beam, then goto abort state and wait for abort reset |
| $38 | *Abort Reset* | Get out of abort state and wait for beam |
| $39 | *Flash* | Create a flash frame |
| $3A | *Profile* | Create a profile frame |
| $3B | *Display* | Create a display frame |
| **Recycler** | | |
| $A1 | *Prepare for Beam* | Prepare for beam by resetting the software integrators. |
| $E6 | *End of Beam* | End of beam. |
| $E7 | *Abort* | End of beam, then goto abort state and wait for reset. |
| $E8 | *Abort Reset* | Get out of abort state and wait for beam. |
| $B5 | *Flash* | Create a flash frame. |
| $B3 | *Profile* | Create a profile frame. |
| $B4 | *Display* | Create a display frame. |
| **Nova Specific** | | |
| $8F | *Clear Integration* | Clear the total integration value to prevent overflow. |
| $FE | *Take Pedestal* | Run the end of beam, prepare for beam cycle. |
| **Muon Campus** | | |
| $84 | *Prepare for Beam* | Prepare for beam by issuing a Reset_DC, resetting the circular buffers, and clearing the abort in progress line. |
| $86 | *End of Beam* | End of beam. |
| $87 | *Abort* | End of beam, then goto abort state and wait for abort reset |

NOT IMPLEMENTED

| TCLK | State Input | Action |
|---|---|---|
| $88 | *Abort Reset* | Get out of abort state and wait for beam |
| $8C | *Flash* | Create a flash frame |

## 4.2 MDAT Frame

An MDAT frame forces a change of abort thresholds if the machine state has changed. The TC receives MDAT frames and places the state number in the FIFO. The CCP polls the FIFO status and reads the MDAT frame from the FIFO. It then switches to the proper thresholds page in the DC, copies the abort mask information corresponding to the MDAT machine state from VME memory to the AC, and tells the TC to generate an update abort settings signal. MDAT frame $12 used to contain the Tevatron state, but now contains the switchyard state, and MDAT frame $56 contains the Main Injector state. The state number received by the CC is in the range 0-255 with 0-127 being the SWYD state, and 128-255 being 128 + the Main Injector state.

## 4.3 Data Latch Interrupt

Periodically, at each sum period, a corresponding latch is generated by the TC by setting IRQ2 on the control bus. The CC is interrupted and the CCP reads the latch status registers in the TC to determine which sum data to read from the Digitizer Card. It then reads the data and stores it in the circular buffers in VME accessible memory. Finally, it clears the corresponding status register which in turn clears IRQ2. Most of the actual memory copies from DC's to circular buffers are handled by assembly language instructions that are modularized in C language #defines. The assembly code makes use of specialized instructions to do multi-byte copies from one memory location to another, and is optimized to do most address manipulations in registers.

In the case of Nova, abort threshold comparisons are also performed in software during the data latch interrupt. The comparisons are done on the MI and RR integrated values. These results are then checked by the abort card service routine to determine if an abort is present.

## 4.4 Abort Card Service

The AC is periodically polled by the CCP and if the current status indicates over threshold, or abort, or channel not OK, the CCP status register is modified to reflect the state of the AC. If the AC is indicating an abort was requested, the snapshot of the last frame of the abort card before the abort is copied to VME memory for access by the CP. The cumulative OR of the snapshots is also copied to VME memory. In the case of Nova, if the software thresholds are exceeded, then the appropriate abort line in the AC is set by the CCP.

## 4.5 Crate Processor

The CCP must poll various VME registers to respond to new data and/or requests from the front end. When the CP wants to initiate one of these events, it loads the proper settings and then writes a non-zero value to the appropriate register. The italicized entries are triggers that are located in the debug section of VME memory.

- **Update Time**
  The time value must be copied to the TC and the TC must be told to update its time value.
- **Update Settings**
  The DC and TC global settings (e.g. Fast Sum Length) must be copied to the DC and TC

boards. This is only done between beam cycles. So in the Tevatron, if there is a necessity to do this during a store, the CCP must be given a fake end-of-beam event (see below) and then restarted after.

- **Update Abort Settings**
  The DC thresholds, and the AC abort settings must be written to the DC and AC boards. Additionally, the TC must be told to assert the update settings line in order for the changes to take effect in the AC. In the DC, changes are immediate. Since they are immediate, the CCP switches the DC to threshold page zero which contains pseudo-infinite thresholds, and then updates the threshold pages and sets the threshold page back to the proper one.

- **Read Pedestals**
  Read the pedestals from the digitizer cards and place them in VME memory.

- **Clear Abort Information**
  Bit 0 tells the CCP to reset the bits in the Status Word that indicate the state of the AC. This also clears the local abort snapshot buffer and resets the abort snapshot buffer on the AC.
  Bit 1 tells the CCP to clear the abort snapshot OR, both locally and on the AC.
  Bit 2 tells the CCP to reset the channel OK list, both locally and on the AC.
  Bit 3 tells the CCP to unset the integrated abort lines.
  Bit 4 tells the CCP to clear the MI integrated abort info.
  Bit 5 tells the CCP to clear the RR integrated abort info.

- **Read / Write Flash Memory**
  Reading copies the flash memory to the VME memory section reserved for flash downloads/uploads. Writing copies the VME memory section to flash memory. Writing to flash memory involves a protocol and is more elaborate than reading. To initiate a write, the CP must write a 0xA596 to the register.

- **Reboot CC**
  Reboot the CC which asserts RESET_SM on the control bus. For when you really would like to reset the crate but not the CP.

- ***Read/Write Control Bus Memory***
  *Read copies 256 bytes of control bus memory to VME memory starting at the specified 16 bit address. The write copies the data byte to the specified 16 bit control bus address. To do this requires writing 0xA596 to the register to avoid accidentally doing this. These are used for debugging purposes and are not part of normal operation.*

- ***Run User Function***
  *This trigger runs a user specified chunk of assembly code that was downloaded to VME memory. It gets copied to normal RAM and the entry point is called.*

- ***Fake TCLK***
  *Send the specified TCLK to the TCLK handler. Exactly the same as if that clock event was received.*

- ***Fake MDAT***
  *Send the specified MDAT frame to the MDAT handler. Exactly the same as if that MDAT frame had been received.*

- ***Change Program State***
  *Send the requested state input to the requested state machine.*

- ***Pause the system***
  *When this is received, the system goes into a paused state as soon as it enters a non-beam state. So in the main injector, it would go here after the end of the current cycle (or immediately if currently between cycles). In this state, the control card effectively ignores clock events. This input acts like a toggle button, so successive triggers take the system in and out of the paused state.*

- ***Change Machine State***
  *Change state to the specified state. The specified state must first be written to the vme_MachineState register.*

- ***Toggle TCLK polling***
  *Toggles whether or not the ez80 is processing TCLKs.*

- ***Toggle MDAT polling***
  *Toggles whether or not the ez80 is processing MDATs.*

- ***Toggle Data Latch Interrupts***
  *Toggles whether or not the ez80 is processing data latches.*

- ***Toggle Abort Card polling***
  *Toggles whether or not the ez80 is processing AC information.*

- ***Enable/Disable CPU Time measurements***
  *Toggles the CPU time measurments.*

- ***Update custom TCLK events***
  *Triggers an update to the custom TCLK ←→ State Input map used during debugging sessions.*

- ***Install new TCLK in TC***
  *Write a new TCLK with specified action to the TC. Can be used, for instance, to install TBT clock events.*

- ***Install new BSCLK in TC***
  *Write a new BSCLK with specified action to the TC. Can be used, for instance, to install TBT clock events.*

- ***Toggle Event Monitor Readings***
  *Toggles whether or not TCLKs, MDATs, and other such slow things are recorded in a timestamped buffer which can be dumped to VME memory.*

- ***Dump Event Monitor Readings***
  *Copy the event monitor buffer to VME memory, starting from the oldest data.*

- ***Freeze Pedestals***
  *Put the system into a pseudo state where just before make_meas is started, the settings page in use on the DC is flipped to a temporary one. The pedestals are then taken and written to that page. Once pedestals are done, the settings page is restored. This allows the same pedestals to be used forever, rather than updating every cycle.*

- ***Thaw Pedestals***
  *This undoes the previous trigger. So pedestals are then updated every pedestal cycle.*

# 5 Output Control Lines

The control bus Reset State Machine signal can be used to quickly reset the system on the fly in the event that a card gets confused. A Reset is wired to port B pin 1 on the eZ80 which is setup as an output pin. This pin must be pulled low to assert the reset. This is done at boot time but currently at no other time. It would probably be initiated at other times from the CP.

# 6 Remote Programming

To remotely change the CC control program, one must first obtain a file containing the new code, and then write it to the flash memory of the remote board. The simplest way to do this is to flash a test board (which has to be done anyway in order to test changes to the code) using the normal Zilog tool and then copy the contents of the flash memory for use in the remote board. This avoids having to decipher where to install the various interrupt vector tables, not to mention *what* to install.

The exact procedure for reprogramming the flash memory on the CC is as follows:

1) Make whatever changes are needed to the code
The code is contained in the blmez80 project in the controls CVS repository. It is 'tagged' by the current version (vNN).

2) Build the application under the Zilog IDE
The Zilog IDE resides on the BLM PC upstairs in the transfer gallery. The code is usually fetched using FileZilla(sp?) from nova and put in the C:\BLM_data\blmez80\ directory. The readme.txt file specifies all the files.

3) Flash the memory of the test board using the Zilog IDE
Under the Tools menu, there is flash loader entry maybe? That talks to the little box that is attached to an empty panel in the VME crate. If it can't communicate with the box, then pull the power cord from the little box and plug it back in. You will typically click the 'Burn and Verify' button. When it is done in a few seconds, close the window.

4) Reboot the crate and start the ez80 program running from the crate processor.

5) Write a 1 to address NN800008 to trigger a read of the flash memory

6) Copy the contents of 128Kb of memory starting at NN820000, which contains the 'just read' contents of the flash memory, to a file someplace

7) Copy the contents of the file to address NN820000 of the remote board

8) Write A596 to address NN800008 which triggers a write to flash memory

9) Reboot the board

From the BLM front end, one can use the ez80FlashRead("filename") and ez80FlashWrite("filename") functions to do steps 5…8.

One thing to be aware of is that the existing code is limited to 128kb of space. When the board is flashed using the Zilog tool, it tells you how many blocks of flash memory will be erased. This is the size of the program. If the program ever needs more than the first 4 blocks (more than 128k), then the eZ80 program and the front end code will need to be modified to handle the larger code package (the details of which are left as an exercise for the student).

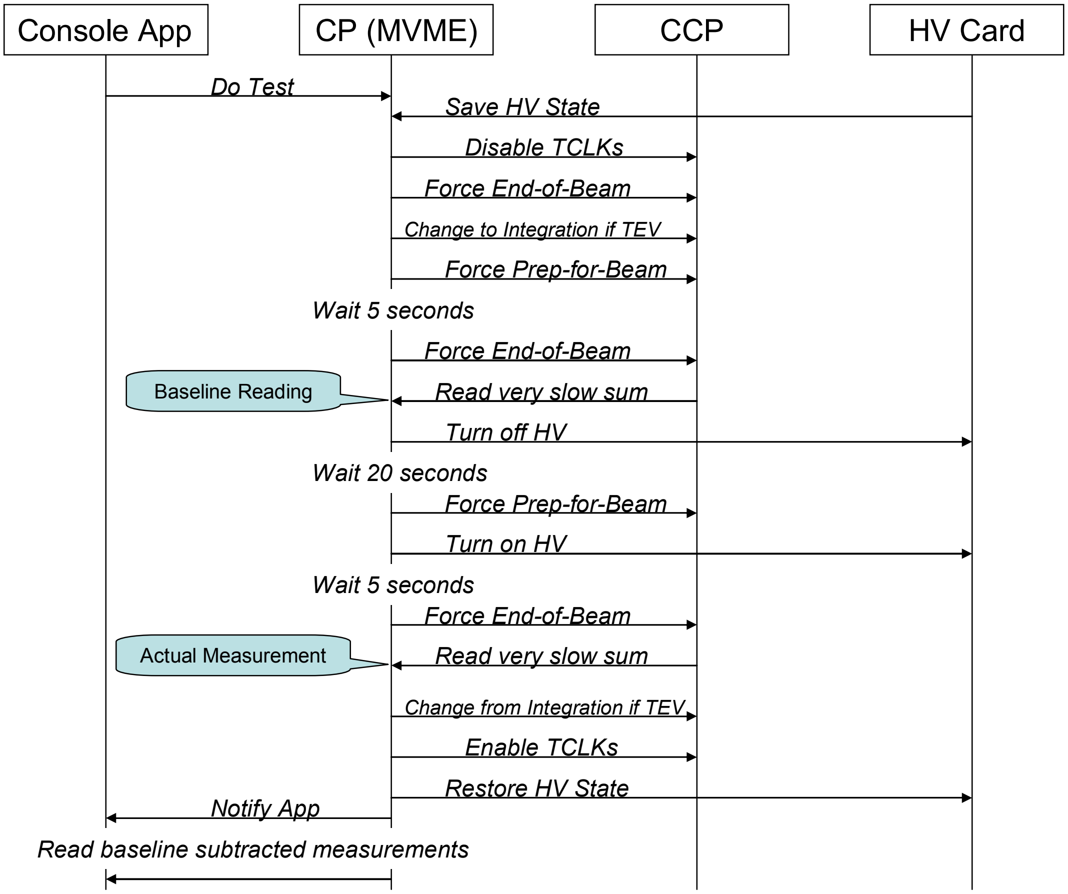
# 7 Miscellaneous Stuff

The ERROR line on the J2 backplane connector is wired up to port B pin 2 on the eZ80. It is periodically monitored by the CCP and if it is ever asserted (low), a bit is latched in the status word. Currently, the ERROR line is used for random purposes by at least the TC, so it should not be actively monitored by the CP.

The CCP also periodically copies the temperatures of the AC, TC, and DCs to VME memory for monitoring by the CP.

# 8 Test Mode

When testing the system hardware, it is advantageous to be able to pulse all the channels in a crate and see that they respond properly. One way of doing this with the BLM system, is to turn the high voltage off for a few minutes, and then turn it back on and record the signal coming from each channel. This is best accomplished by the following procedure on the CP.



# 9 References

[1] **BEAMS-DOC-1410** Beam Loss Monitor Upgrade Users' Guide. Documents hardware and functionality of the system.

# A Data Record

The fundamental unit stored in the various data buffers has the following format:

| Byte Offset | Length | Description |
|---|---|---|
| 0x00 | 1 | Abort State |
| 0x01 | 1 | Measurement Divisor |
| 0x02 | 2 | Sum Divisor |
| 0x04 | 1 | Instantaneous Abort Status from AC |
| 0x05 | 1 | Channel Count |
| 0x06 | 1 | Data flag to indicate normal (0), last of cycle (1), new cycle (2), or waiting for stable data (3) frame |
| 0x07 | 1 | MDAT State |
| 0x08 | 4 | Time stamp: Microseconds since last TCLK 1 Hz Event |
| 0x0C | 4 | Time stamp: seconds since T0 (*i.e.* 1 January 1970) |
| 0x10 + n*4 | 240 | Sliding Sum data for channel *n* |

The data flag normally cycles through the following progression:

- at start of beam, wait for the digitizer delay, then write a 2 in the first frame collected (for each type of sum)
- write 3 in all the succeeding frames until after the input switch is closed for the specified amount of time
- write 0 in the rest of the frames of the cycle
- at end of beam, write a 1 in the last frame in the circular buffer

In the abort status word, bits 0-3 are the status of the abort from the Immediate, Fast, Slow, and Very Slow measurements, respectively. Bits 4-7 are not used. Loss data are stored as 32-bit long words with data in order from least significant to most significant byte.

Profile, Flash, and Display frames each contain 2 of these data frames. The first one is the user selectable Fast or Slow Sum frame and the second is the Very Slow Sum frame which in the Main Injector contains the integrated value.

# B Abort Information Layout

Note that the maps below contain enough entries for 256 states to keep consistency with the abort threshold arrays. Presently the code only handles states 0-63 (zero cannot be changed from its default and is used internally when changing settings).

## B.1 Channel Masks

For each species of abort, there are 7 bytes of channel masks. Byte 0 is unused. So, for instance, if there were 44 (0 to 43) channels in some crate, the channel assignment within the 8 bytes would be as follows (x means unused).

| **Bytes** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Bits** | **7… 0** | **7… 0** | **7… 0** | **7… 0** | **7… 0** | **7… 0** | **7… 0** | **7… 0** |
| **Channel** | x…x | x…x 43…40 | 39…32 | 31…24 | 23…16 | 15…8 | 7…0 | x…x |

So to enable channel 17, one would write a 1 to byte 3, bit 1; and to enable channel 7, one would write a 1 to byte 1, bit 7.

## B.2 MDAT → Abort State Map

The map to go from MDAT state to Abort state is a 256 byte array indexed by MDAT state. The following table illustrates how the information is stored in each word. There is room for 256 entries to cover all possible states, but only 64 are presently used.

| VME address | Most Significant Byte | Least Significant Byte |
|---|---|---|
| 0x098E0000 | Abort state for MDAT state 1 | Abort State for MDAT state 0 |
| 0x098E0002 | Abort state for MDAT state 3 | Abort State for MDAT state 2 |
| 0x098E0004 | Abort state for MDAT state 5 | Abort State for MDAT state 4 |
| … | … | … |

## B.3 TEV F Sector

### B.3.1 List of MI Start and End TCLK Events

This list is actually 2 lists of 256 bytes each: a list of starting TCLKs, and a list of ending TCLKs, each indexed by MI MDAT state. The format of each list is identical. The following table shows the format for either list. A starting TCLK value of 0x00 indicates there is no action to take for that MI MDAT state. There is room for 256 entries in each list to cover all possible states, but currently only the first 64 are used.

| VME address | | Most Significant Byte | Least Significant Byte |
|---|---|---|---|
| Starting TCLKs | Ending TCLKs | | |
| 0x098E0100 | 0x098E0200 | TCLK for MDAT state 1 | TCLK for MDAT state 0 |
| 0x098E0102 | 0x098E0202 | TCLK for MDAT state 3 | TCLK for MDAT state 2 |
| 0x098E0104 | 0x098E0204 | TCLK for MDAT state 5 | TCLK for MDAT state 4 |
| … | … | … | … |

### B.3.2 Table of Actions vs. TEV Abort States / MI MDAT States

This is a 2-D byte array containing the action to take when a MI starting TCLK event occurs. It is indexed by both TEV abort state and MI MDAT state. A value of 0x00 indicates that the AC outputs should just be masked off. A value of 0xFF indicates that nothing should be done. Any other value is interpreted as a TEV abort state to switch to between the starting and ending clock events. The default is 0x00, *i.e.* mask off the AC outputs. The indices are arranged with MI MDAT state incrementing the fastest. There is enough memory reserved for a 256 x 256 byte array, but currently, it is only 64 x 64. The following table illustrates the layout presently.

| VME address | Most Significant Byte | Least Significant Byte |
|---|---|---|
| 0x098F0000 | Action for MI MDAT state 1 and TEV abort state 0 | Action for MI MDAT state 0 and TEV abort state 0 |
| 0x098F0002 | Action for MI MDAT state 3 and TEV abort state 0 | Action for MI MDAT state 2 and TEV abort state 0 |
| 0x098F0004 | Action for MI MDAT state 5 and TEV abort state 0 | Action for MI MDAT state 4 and TEV abort state 0 |
| … | … | … |
| 0x098F0040 | Action for MI MDAT state 1 and TEV abort state 1 | Action for MI MDAT state 0 and TEV abort state 1 |
| 0x098F0042 | Action for MI MDAT state 3 and TEV abort state 1 | Action for MI MDAT state 2 and TEV abort state 1 |
| 0x098F0044 | Action for MI MDAT state 5 and TEV abort state 1 | Action for MI MDAT state 4 and TEV abort state 1 |
| … | … | … |

# C VME Dual Port Memory Map

VME dual port memory addresses start at 0x09800000. The eZ80 addressing of the dual port memory begins at 0x800000 (24 bit addressing). The following addresses are specified as offsets from the base addresses.

| Address | Size | Description |
|---|---|---|
| **System Status** | | |
| 000000 | 2 | Status Word |
| 000002 | 2 | Reboot the CC (requires 0xA596 to be written here) |
| 000004 | 1 | Choice for whether fast sum (0) or derippled (1) data is used for FPD frames |
| 000006 | 1 | Bit pattern for selecting Flash/Profile/Display source<br>Bit 0 = flash, bit 1 = profile, bit 2 = display; 0 = fast, 1 = slow |
| 000008 | 1 | Read flash memory |
| 00000A | 2 | Load flash memory (requires 0xA596 to be written here) |
| 00000C | 2 | Flash memory load status |
| 00000E | 1 | Clear Abort Info<br>Bit 0 = abort info bits in status word and snapshot<br>Bit 1 = snapshot OR<br>Bit 2 = channel OK list |
| 000010 | 1 | Force Pedestal Read |
| 000012 | 1 | Update Time Setting |
| 000014 | 2 | Time Setting LSW |
| 000016 | 2 | Time Setting MSW |
| 000018 | 1 | Update Settings |
| 00001A | 1 | Update Abort Settings |
| 00001C | 1 | Machine Type |
| 00001E | 2 | Extended Status (machine specific status info) |
| **Flash, Profile and Snapshot Indexes** | | |
| 000020 | 2 | Flash Frame Counter |
| 000022 | 2 | Profile Frame Counter |
| 000024 | 2 | Fast Sum Circular Buffer Frame Index 1 |
| 000026 | 2 | Fast Sum Circular Buffer Frame Index 2 |
| 000028 | 2 | Slow Sum Circular Buffer Frame Index 1 |
| 00002A | 2 | Slow Sum Circular Buffer Frame Index 2 |
| 00002C | 2 | Very Slow Sum Circular Buffer Frame Index 1 |
| 00002E | 2 | Very Slow Sum Circular Buffer Frame Index 2 |
| 000030 | 1 | Fast Sum Frame Index Pointer |
| 000032 | 1 | Slow Sum Frame Index Pointer |
| 000034 | 1 | Very Slow Sum Frame Index Pointer |
| 000036 | 2 | TC Raw data pointer |
| 000038 | 2 | DC Raw data pointer |
| 00003A | 2 | AC Raw data pointer |
| 00003C | 2 | DeRippled Sum Circular Buffer Frame Index 1 |
| 00003E | 2 | DeRippled Sum Circular Buffer Frame Index 2 |

| Address | Size | Description |
|---|---|---|
| 000040 | 1 | DeRippled Sum Frame Index Pointer |
| 000042 | 2 | RR Flash Frame Counter |
| 000044 | 2 | RR Profile Frame Counter |
| 000046 | 2 | MI Very Slow Sum Circular Buffer Frame Index 1 |
| 000048 | 2 | MI Very Slow Sum Circular Buffer Frame Index 2 |
| 00004A | 1 | MI Very Slow Sum Frame Index Pointer |
| 00004C | 2 | RR Very Slow Sum Circular Buffer Frame Index 1 |
| 00004E | 2 | RR Very Slow Sum Circular Buffer Frame Index 2 |
| 000050 | 1 | RR Very Slow Sum Frame Index Pointer |
| 000052 | 2 | Last Very Slow Frame Index 1 |
| 000054 | 2 | Last Very Slow Frame Index 2 |
| 000056 | 2 | Last Very Slow Frame Index Pointer |
| 000058 | 2 | Last MI Integrated Frame Index 1 |
| 00005A | 2 | Last MI Integrated Frame Index 2 |
| 00005C | 2 | Last MI Integrated Frame Index Pointer |
| 00005E | 2 | Last RR Integrated Frame Index 1 |
| 000060 | 2 | Last RR Integrated Frame Index 2 |
| 000052 | 2 | Last RR Integrated Frame Index Pointer |
| **Abort Info** | | |
| 000080 | 7 | Channel OK List from Abort Card (1 means OK, 0 means problem) |
| **Current MDAT and Abort States** | | |
| 000090 | 4 | MDAT State (Word for each machine) |
| 000094 | 4 | Abort State (Word for each machine) |
| **Settings** | | |
| 000100 | 1 | Channel Count |
| 000102 | 2 | Make_Meas Divisor |
| 000104 | 2 | Fast Sum Length |
| 000106 | 2 | Slow Sum Length |
| 000108 | 2 | Very Slow Sum Length |
| 00010A | 2 | Digitizer FPGA Control Register ( high byte is number of make_meas to skip before performing pedestal measurements / 16 ) |
| 00010C | unused | Digitizer Test DAC |
| 00010E | 1 | Timing Card Control Bus CSR Register |
| 000110 | unused | Timing Card Control Settings (reserved) |
| 000112 | 1 | What generates an IRQ3 interrupt from the Abort Card? |
| 000114 | 1 | Abort Card enable (Bit 0 is *global enable*, Bit 4 is *require 2 consecutive aborts*) |
| 000116 | 2 | Pedestal Sum Length |
| 000118 | 1 | End of beam delay before asserting AIP (in fast latch units) |
| 00011A | 1 | Delay after Flash clock event before collecting flash frame (in fast latch units) |
| 00011C | 1 | Delay after Profile clock event before collecting profile frame (in fast latch units) |

| Address | Size | Description |
|---|---|---|
| 00011E | 1 | Delay after Display clock event before collecting display frame (in fast latch units) |
| 000120 | 1 | Turn off inputs while doing pedestals |
| 000122 | 1 | Optional extra delay after pedestals before re-enabling DC inputs |
| 000124 | 1 | Delay after re-enabling DC inputs before reading data |
| 000126 | 1 | Delay (slow sum units) after the end TCLK event before re-enabling normal F sector aborts |
| 000128 | 1 | Max DY for CIC derippling procedure to allow new pedestal waveform |
| 00012A | 1 | CIC sum length |
| 00012C | 1 | Take Pedestal TCLK event |
| 00012E | 2 | Delay between Pedestal TCLK and Pedestal in ms |
| 000130 | 2 | Maximum period between Pedestals before alarming (front end job) |
| 000200 | 2 | Channel 0 Mode Select |
| 000202 | 2 | Channel 0 MADC Manual Setting |
| | | ↓ |
| 0002EC | 2 | Channel 59 Mode Select |
| 0002EE | 2 | Channel 59 MADC Manual Setting |
| **Pedestal Storage** | | |
| 000300 | 256 | 32-bit pedestals stored in standard data record |
| **Squelch Storage** | | |
| 000400 | 120 | 16-bit squelch values |
| **Abort Snapshot Storage** | | |
| 000500 | 32 | Abort snapshot record from last frame before abort |
| 000520 | 32 | Abort snapshot OR of all abort snapshots since this was last cleared |
| 000540 | 2 | Abort circular buffer pointer at time of abort |
| 000542 | 2 | Fast Sum index at time of abort |
| 000544 | 2 | Slow Sum index at time of abort |
| 000546 | 2 | Very Slow Sum index at time of abort |
| 000548 | 2 | DeRippled Sum index at time of abort |
| 00054A | 2 | MI Integrated Sum index at time of abort |
| 00054C | 2 | RR Integrated Sum index at time of abort |
| 000550 | 32 | MI Abort snapshot record from last frame before abort |
| 000570 | 32 | MI Abort snapshot OR of all abort snapshots since this was last cleared |
| 000590 | 32 | RR Abort snapshot record from last frame before abort |
| 0005B0 | 32 | RR Abort snapshot OR of all abort snapshots since this was last cleared |
| **Integration Pedestal Storage** | | |
| 000600 | 256 | 32-bit Integration pedestals |
| **Temperatures** | | |
| 000700 | 2 | TC Temperature Value |
| 000702 | 2 | AC Temperature Value |
| 000704 | 30 | DC Temperature Values for all DCs (each 2 bytes) |
| **Debug Information** | | |

| **Address** | **Size** | **Description** |
|---|---|---|
| 010000 | 48 | Program name and build date |
| 010030 | 4 | Test Sequence ( 0x44332211 ) |
| 010034 | 4 | TCLK count |
| 010038 | 4 | MDAT frame count |
| 01003C | 2 | Last TCLK received |
| 01003E | 2 | Last MDAT state received |
| 010040 | 4 | Data Interrupt count |
| 010044 | 4 | Fast Sum Latch count |
| 010048 | 4 | Slow Sum Latch count |
| 01004C | 4 | Very Slow Sum Latch count |
| 010050 | 4 | Profile request count |
| 010054 | 4 | Flash request count |
| 010058 | 4 | Display request count |
| 01005C | 4 | Abort Interrupt count |
| 010060 | 4 | Channel not OK interrupt count |
| 010064 | 4 | Channel abort interrupt count |
| 010068 | 4 | Crate abort interrupt count |
| 01006C | 4 | Missed Fast Latch count |
| 010070 | 4 | Missed Slow Latch count |
| 010074 | 4 | Missed Very Slow Latch count |
| 010078 | 2 | Hardware State machine state |
| 01007A | 2 | Number of fast data latches to wait until pedestals are done |
| 01007C | 4 | Last Fast Latch time microsecs |
| 010080 | 4 | Last Fast Latch time seconds |
| 010084 | 4 | Last Slow Latch time microsecs |
| 010088 | 4 | Last Slow Latch time seconds |
| 01008C | 4 | Last Very Slow Latch time microsecs |
| 010090 | 4 | Last Very Slow Latch time seconds |
| 010094 | 4 | Stack Pointer (Updated every time through polling loop) |
| 010098 | 4 | Stack Integrity (0xA4A3A2A1; also updated every loop) |
| 01009C | 16 | Digitizer Map (map of found digitizers; 16th byte is 0) |
| 0100AC | 4 | Timing Card clock error count |
| 0100B0 | 4 | Count of times TCLK queue has more than 1 entry |
| 0100B4 | 4 | Count of times MDAT queue has more than 1 entry |
| 0100B8 | 4 | Number of times a state ≥ 64 was received |
| 0100BC | 2 | Last state machine input |
| 0100BE | 2 | Trigger to simulate TCLK (requires 0xA596 to be written here) |
| 0100C0 | 2 | TCLK to simulate |
| 0100C2 | 2 | Pause the system (requires 0xA596 to be written here). This happens now if waiting for beam, or after next end of beam event. |
| 0100C4 | 2 | Change the system state (requires 0xA596 to be written here). |
| 0100C6 | 2 | Toggle TCLK polling (requires 0xA596 to be written here) |
| 0100C8 | 2 | Toggle MDAT polling (requires 0xA596 to be written here) |
| 0100CA | 2 | Toggle Data interrupts (requires 0xA596 to be written here) |

| Address | Size | Description |
|---|---|---|
| 0100CC | 2 | Toggle Abort polling (requires 0xA596 to be written here) |
| 0100CE | 2 | Enable CPU time measurements (just write a 0x0001 here; 0x0000 turns off CPU time measurements) |
| 0100D0 | 2 | Number of species of CPU time measurements |
| 0100D2 | 2 | DC Debug output. (0=off, 1=pedestal waveform is put in very slow sum buffer, 2=CIC output is put in very slow sum buffer) |
| 0100D4 | 2 | Update custom TCLKs (requires 0xA596 to be written here) |
| 0100D6 | 2 | Toggle the event history recording (requires 0xA596 to be written there) |
| 0100D8 | 2 | Copy history data to buffer (requires 0xA596 to be written there) |
| 0100DA | 2 | Freeze the next set of pedestals (requires 0xA596 to be written there) |
| 0100DC | 2 | Unfreeze the pedestal collection (requires 0xA596 to be written there) |
| 0100DE | 2 | Send a State Machine input (requires 0xA596 to be written there) |
| 0100E0 | 2 | State Machine input to send |
| 0100E2 | 2 | State Machine to send to |
| 0100E4 | 2 | For Nova, determines whether the Very Slow Sum gets the total integration value (0), or the original Very Slow Sum data (1) |
| 0100E6 | 2 | Trigger to simulate MDAT (requires 0xA596 to be written there) |
| 0100E8 | 2 | MDAT frame to simulate |
| 0100EA | 4 | Software State Machines' states |
| 0100F0 | 2 | Last event executed in the event loop |
| 0100F2 | 4 | Number of times TCLK FIFO is full |
| 0100F6 | 4 | Number of times TCLK claimed to have data, but didn't |
| 0100FA | 2 | Add a new TCLK to TC; Low byte is clock event; high byte is action |
| 0100FC | 2 | Add a new BSCLK to TC; Low byte is clock event; high byte is action |
| 010100 | 1024 | TCLK counts for each TCLK type (256 types) |
| 010500 | 1024 | MDAT counts for each MDAT state (256 states) |
| 010900 | 2048 | CPU time measurement buffer (each measurement type occupies 32 words) |
| 011200 | 512 | State inputs for each custom TCLK |
| 011400 | 512 | Which state machine to call for each state input |
| 012000 | 8192 | History data buffer |
| 018000 | 2 | Request a Control Bus memory dump |
| 018002 | 2 | Request a Control Bus write (requires 0xA596 to be written here) |
| 018004 | 2 | Control Bus address to dump / write (dumps 256 bytes / write 1 byte) |
| 018100 | 256 | Data copied from / written to Control Bus (only 1 byte in case of write) |
| 019000 | 2 | Request User Code to be run (requires 0xA596 to be written here) |
| 01A000 | 8192 | Buffer for User Code to be placed |
| | | **Flash Memory Download** |
| 020000 | 128K | Buffer for new flash memory contents |
| | | **RR Flash Frames** |
| 040000 | 512 | Flash Frame 0 |
| 040200 | 512 | Flash Frame 1 |

| Address | Size | Description |
|---|---|---|
| ↓ | | |
| 05FE00 | 512 | Flash Frame 255 |
| **RR Profile Frames** | | |
| 060000 | 512 | Profile Frame 0 |
| 060200 | 512 | Profile Frame 1 |
| ↓ | | |
| 07FE00 | 512 | Profile Frame 255 |
| **Flash Frames (MI in Nova case)** | | |
| 080000 | 512 | Flash Frame 0 |
| 080200 | 512 | Flash Frame 1 |
| ↓ | | |
| 09FE00 | 512 | Flash Frame 255 |
| **Profile Frames (MI in Nova case)** | | |
| 0A0000 | 512 | Profile Frame 0 |
| 0A0200 | 512 | Profile Frame 1 |
| ↓ | | |
| 0BFE00 | 512 | Profile Frame 255 |
| **Display Frame** | | |
| 0C0000 | 512 | Display Frame (MI in Nova case) |
| 0C0200 | 512 | RR Display Frame |
| **Abort Stuff** | | |
| 0E0000 | 128 | MDAT state to Abort state map (MI in Nova case) |
| 0E0080 | 128 | MDAT state to RR Abort state map |
| 0E0100 | 256 | F sector start TCLKs (indexed by MI MDAT state; a value of 0 means not implemented) This is a 1-D byte array, one byte per MI state. |
| 0E0200 | 256 | F sector end TCLKs (also indexed by MI MDAT state) Also a byte array |
| 0F0000 | 65536 | Array of F sector actions to take, indexed by TEV MDAT state and MI MDAT state. (An action of 0 means mask off the Abort card; a value of 0xFF means don't do anything, any other value means switch to that TEV abort state) These actions are implemented between the start and end TCLKs given by the lists above. This is a 2-D byte array. The fastest changing index is the MI state, i.e.<br>unsigned char array[TEV states][MI states] |
| **Abort States (MI in Nova case)** | | |
| **State 0** | | |
| 100000 | 1 | State Number |
| 100002 | 2 | Channel Masks 0 1 for Immediate Abort |
| 100004 | 2 | Channel Masks 2 3 for Immediate Abort |

| Address | Size | Description |
|---|---|---|
| 100006 | 2 | Channel Masks 4 5 for Immediate Abort |
| 100008 | 2 | Channel Masks 6 7 for Immediate Abort |
| 10000A | 2 | Channel Masks 0 1 for Fast Abort |
| 10000C | 2 | Channel Masks 2 3 for Fast Abort |
| 10000E | 2 | Channel Masks 4 5 for Fast Abort |
| 100010 | 2 | Channel Masks 6 7 for Fast Abort |
| 100012 | 2 | Channel Masks 0 1 for Slow Abort |
| 100014 | 2 | Channel Masks 2 3 for Slow Abort |
| 100016 | 2 | Channel Masks 4 5 for Slow Abort |
| 100018 | 2 | Channel Masks 6 7 for Slow Abort |
| 10001A | 2 | Channel Masks 0 1 for Very Slow Abort |
| 10001C | 2 | Channel Masks 2 3 for Very Slow Abort |
| 10001E | 2 | Channel Masks 4 5 for Very Slow Abort |
| 100020 | 2 | Channel Masks 6 7 for Very Slow Abort |
| 100022 | 2 | Abort Multiplicity for Immediate and Fast Abort |
| 100024 | 2 | Abort Multiplicity for Slow and Very Slow Abort |
| 100026 | 2 | Crate Abort Mask |
| 100028 | 8 | Unused |
| 100030 | 2 | Channel 0 Immediate Threshold |
| 100032 | 2 | Channel 1 Immediate Threshold |
|  |  | ↓ |
| 1000A6 | 2 | Channel 59 Immediate Threshold |
| 1000A8 | 8 | Unused |
| 1000B0 | 2 | Channel 0 Fast Threshold LSW |
| 1000B2 | 2 | Channel 0 Fast Threshold MSW |
| 1000B4 | 2 | Channel 1 Fast Threshold LSW |
| 1000B6 | 2 | Channel 1 Fast Threshold MSW |
|  |  | ↓ |
| 10019C | 2 | Channel 59 Fast Threshold LSW |
| 10019E | 2 | Channel 59 Fast Threshold MSW |
| 1001A0 | 16 | Unused |
| 1001B0 | 2 | Channel 0 Slow Threshold LSW |
| 1001B2 | 2 | Channel 0 Slow Threshold MSW |
| 1001B4 | 2 | Channel 1 Slow Threshold LSW |
| 1001B6 | 2 | Channel 1 Slow Threshold MSW |
|  |  | ↓ |
| 10029C | 2 | Channel 59 Slow Threshold LSW |
| 10029E | 2 | Channel 59 Slow Threshold MSW |
| 1002A0 | 16 | Unused |
| 1002B0 | 2 | Channel 0 Very Slow Threshold LSW |

| Address | Size | Description |
|---|---|---|
| 1002B2 | 2 | Channel 0 Very Slow Threshold MSW |
| 1002B4 | 2 | Channel 1 Very Slow Threshold LSW |
| 1002B6 | 2 | Channel 1 Very Slow Threshold MSW |
| ↓ | | |
| 10039C | 2 | Channel 59 Very Slow Threshold LSW |
| 10039E | 2 | Channel 59 Very Slow Threshold MSW |
| 1003A0 | 80 | Unused |
| **States 1 – 127; each with same offsets as State 0** | | |
| 100400 | 1024 | State 1 Settings |
| 100800 | 1024 | State 2 Settings |
| ↓ | | |
| 11F800 | 1024 | State 127 Settings |
| **RR Abort Settings for States 0-127** | | |
| 120000 | 128 K | RR abort states 0-127 |
| **In Use Abort Settings for States 0-127** | | |
| 140000 | 128 K | Copy of settings that are in use. These are the settings that are copied to AC and DCs. |
| **RR In Use Abort Settings for States 0-127** | | |
| 160000 | 128 K | Copy of settings that are in use for RR. |
| **Circular Buffers** | | |
| 180000 | 512 K | DeRippled Sum Buffer 2K frames deep |
| 200000 | 2 Meg | Fast Sum Buffer 8K frames deep |
| 400000 | 1 Meg | MI Integrated Sum Buffer 4K frames deep |
| 500000 | 1 Meg | RR Integrated Sum Buffer 4K frames deep |
| 600000 | 1 Meg | Slow Sum Buffer 4K frames deep |
| 700000 | 1 Meg | Very Slow Sum Buffer 4K frames deep |