CESR BPM/BSM/FLM SYSTEM DIGITAL PROCESSOR BOARD

(For XILINX V5.13)

DSP\_Board\_Programming\_V5-13.doc

7/17/2014 3:35 PM

*Reminders:*

* The code must be compiled with Xilinx ISE version 9.2, since many ‘coregen’ modules were dropped in version 10. DO NOT open the ISE9 project files in any later version.
* On windows 7, one must use IMPACT version 11 or later to program the configuration ROM with the USB programmer. XP can use an older version.

*New in 5.13:*

* [ALL] A new module type BPM4\_TLINE was added to ‘constants.txt’.
	+ MODULE\_TYPE
		- = 1 BSM
		- = 2 BPM
		- = 3 FLM
		- = 4 FLMA
		- = 5 ERL\_BPM
		- = 6 BPM4 w/FPGA
		- =7 BSM4 w/FPGA
		- =8 BPM4\_TLINE
	+ MAJOR\_REV = 5
	+ MINOR\_REV = 13
* A new project BPM4\_TLINE was added. This will be used for modules that are installed on the sync-to-cesr transfer lines. They incorporate changes to the DATA\_ACQ hardware register that enable use of external triggers without needing DSP support. It looks more like the BSM4 style of triggering, rather than the BPM4.
* Changed the CERN ODR “IPR” (Intensifier Pulse Repetition Rate) register into “ITD” (Intensifier Turns Delay). Normally, a change that breaks old software would cause the MAJOR\_REV to increment to 6, but the use of this register is so minimal that I have chosen not to.

*New in 5.12:*

* [ALL] The VECTOR ADDRESS TABLE had the addresses of the 8 EXT\_SPI\_OUT\_DAT registers at addresses 0X10040040 thru 0X100400407 inserted in the first 8 locations 1 thru 8. This is primarily to support XBUS vector readout of position data for the CERN ODR experiment.
	+ MAJOR\_REV = 5
	+ MINOR\_REV = 12

*New in 5.11:*

* [BPM4] The ability to send a serial data stream to a remote D/A board (6048-168) was added.
* [BPM4] The ability to send timing signals to a remote LVDS-to-TTL buffer (6048-170) appropriate for triggering the CERN ODR camera was added.
* The top level name of “current” was applied. When the files are copied to the “Q” drive, they will be put under the “V5.-11” directory.
	+ MAJOR\_REV = 5
	+ MINOR\_REV = 11

*New in 5.10:*

* [BSM4] Wait states in ‘dsp\_wait\_ctrl\_BSM4’ and ‘slave\_wait\_ctrl\_BSM4’ were diddled somewhere in V5-9. Therefore, a new V5-10 has been started with the generally longer wait times. We should use a scope and verify the appropriate wait periods.
* [BSM4] Support for the current monitor was added to BSM4.
* [BSM4] The DATA\_ACQ register has been restructured. Trigger control bits have been added and some little used read-only DCM status bits have been shifted from bits 7:4 to bits 10:7 or deleted.
* Obsolete projects were deleted. This included:
	+ ‘nate\_bpm’: This project was in the development path towards the 4 nsec FPGA-based BPM.
	+ ‘bpm’: This project was the original BPM for CESR with the 6048-113 DSP motherboard. The original analog boards are still used in the ERL. However, the ERL now has its own project called ‘erl\_bpm’
	+ ‘flma’: This project was the fast luminosity monitor with accelerator board. It became obsolete when CLEO shut down.
* [BPM4] The ‘bpm\_proto2’ project was renamed to ‘bpm4’. This is the current code for the 4 nsec FPGA-based BPM.
* [BPM4] All Verilog files that contained “PROTO2” in their name were renamed to “BPM4’.
* The top level name of “current” was applied. When the files are copied to the “Q” drive, they will be put under the “V5.-10” directory.
	+ MAJOR\_REV = 5
	+ MINOR\_REV = 10

*New in 5.9:*

1. Timing was altered to support new FLASH (AT29LV040A) chips. New ‘cbi\_net’ code was required to program these devices (cbi\_net version 0.99.1 or newer).

MAJOR\_REV = 5

MINOR\_REV = 9

*New in 5.8:*

1. BSM4 (with FPGA per channel) in Xilinx ISE 9.2.

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

= 5 ERL\_BPM

= 6 BPM w/FPGA

=7 BSM w/FPGA

MAJOR\_REV = 5

MINOR\_REV = 8

*New in 5.7:*

1. BPM\_PROTO2 (for BPM board with FPGA) in Xilinx ISE 9.2. I still need to include programming info in this document.

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

= 5 ERL\_BPM

= 6 BPM w/FPGA

MAJOR\_REV = 5

MINOR\_REV = 7

1. New timing board from Bob Meller. I still need to include programming info in this document.

*New in 5.6:*

1. Recompiled ERL\_BPM with Xilinx ISE 9.2. Still need to do others. No functional changes yet.

*New in 5.5:*

1. Removed the FLM project (we only use the FLMA).

*New in 5.4:*

 1. Address autoincrement for ColdFire readout has been fixed.

2. The ERL\_BPM module type has been created. It uses a 25 MHz input clock (instead of 24 MHz in CESR) and it doubles that to 50 MHz for data acquisition (instead of tripling it to 72 MHz in CESR). The ERL\_BPM only support 122 bunches (instead of 183 in CESR).

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

= 5 ERL\_BPM

MAJOR\_REV = 5

MINOR\_REV = 4

3. Timing of all DSP operations have been verified. After it boots, the DSP can no longer access the FLASH memory. It requires too many wait states.

*New in 5.3:*

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

MAJOR\_REV = 5

MINOR\_REV = 3

1. The BSM current monitor board is supported.

*New in 5.2:*

1. “Force Hi Hit Register” added to the accumulator board for testing.

2. Added detail about accumulator Geo Bunch Rate Register.

3. Scrambled accumulator board mapping to use ADC channels 1 thru 6 to generate the lookup table address.

*New in 5.1:*

1. Support for Ethernet access thru a ColdFire DIMM board from Arcturus.

2. Reset for the DIMM module is provided thru the new register “DIMM\_RESET”.

3. The “MODULE\_TYPE” register has been modified so each project has its own identifier.

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

MAJOR\_REV = 5

MINOR\_REV = 1

4. The FLMA accumulator board is supported.

To Do:

Implement ‘ACQ\_SKIP\_CNT’ register.

Implement shadow readback for timing board

Make sign extension be programmable for unipolar/bipolar

Speed up ‘loc\_dat\_tri\_drive’ with combinatorial switching

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### DATA FORMAT

When reading any memory that is less than 32-bits wide, the data will be considered to be signed 2’s-complement and will be sign extended. When writing any memory that is less than 32-bits wide, the high bits will be ignored.

When reading any register that is less than 32 bits wide, the data will be considered to be unsigned and will contain zeroes in the high bits. If a register needs signed data, it will be designed as a 32 bit register. When writing any register that is less than 32 bits wide, the high bits will be ignored.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ADDRESS MAPS

The local (on-board) address bus, LOC\_ADR[31..0], addresses longword (4-byte) entities. The address map for all peripherals is driven by the addressing capabilities of the DSP. The DSP breaks down the 32-bit address range as follows:

DSP INTERNAL SPACE 0X00000000 – 0X003FFFFF

DSP UNUSED 0X00400000 – 0X07FFFFFF

DSP BANK 0 (/MS0) 0X08000000 – 0X0BFFFFFF

DSP BANK 1 (/MS1) 0X0C000000 – 0X0FFFFFFF

DSP HOST (/MSH) 0X10000000 – 0XFFFFFFFF

The region labeled “unused” is specific to this project. The DSP actually defines features in this region, such as multiprocessor memory space and SDRAM memory space, but this project does not use any of the features.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *DSP (from the XBUS or Ethernet perspective)*

## One has to use 'multiprocessor space' to access things inside the DSP from the outside. For the ADSP-TS101STigerSHARCprocessor with ID=0 (which is the ID for this project), the address range is from 0x02000000 to 0x023fffff. When you want to access a location inside of the DSP, you will need to add 0x02000000 to the actual internal DSP addresses so that the XBUS or Ethernet uses the correct address. The internal DSP address will be calculated by masking off the high byte (0x02).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *DSP (from the DSP perspective)*

The DSP memory map is unique to the ADSP-TS101S TigerSHARC chip. Other DSPs may use different mapping.

DSP INTERNAL SPACE 0X00000000 – 0X03FFFFFF

 MEMORY BLOCK 0 0X00000000 – 0X0000FFFF 64 kW

 MEMORY BLOCK 1 0X00080000 – 0X0008FFFF 64 kW

 MEMORY BLOCK 2 0X00100000 – 0X0010FFFF 64 kW

INT REGISTERS (UREGS) 0X00180000 – 0X001807FF 2 kW

The LDF file defines how the memory is allocated for various functions. The current version of the file “BPM\_ADSP-TS101\_C.LDF” makes the following allocations:

// Start with full M0 block for code. We may use high addresses for some data structures.

// This gives 64k of code space.

 M0Code { TYPE(RAM) START(0x00000000) END(0x0000FFFF) WIDTH(32) }

// M1 block will support data, heap, and stack. We expect no heap usage and very

// little stack usage. Start with 56k data, 2k heap, and 6k stack.

 M1Data { TYPE(RAM) START(0x00080000) END(0x0008DFFF) WIDTH(32) }

 M1Heap { TYPE(RAM) START(0x0008E000) END(0x0008E7FF) WIDTH(32) }

 M1Stack { TYPE(RAM) START(0x0008E800) END(0x0008FFFF) WIDTH(32) }

// M2 block will support raw data from the ADCs. Start with one buffer using

// 56k. An "M2Stack" is required by the C/C++ runtime. Make it be 8k.

 M2Data { TYPE(RAM) START(0x00100000) END(0x0010DFFF) WIDTH(32) }

 M2Stack { TYPE(RAM) START(0x0010E000) END(0x0010FFFF) WIDTH(32) }

// This project does not use the SDRAM address range

 SDRAM { TYPE(RAM) START(0x04000000) END(0x07FFFFFF) WIDTH(32) }

// MS0 bank will address the ADC boards.

// MS0mem will address memory on all 4 cards contiguously

 MS0mem { TYPE(RAM) START(0x08000000) END(0x08FFFFFF) WIDTH(32) }

// MS0reg will address register space on all 4 cards contiguously

 MS0reg { TYPE(RAM) START(0x09000000) END(0x09FFFFFF) WIDTH(32) }

// MS0unused is the remaining part of the MS0 bank

 MS0unused { TYPE(RAM) START(0x0A000000) END(0x0BFFFFFF) WIDTH(32) }

// MS1 bank will address the FLASH and SRAM.

 MS1 { TYPE(RAM) START(0x0C000000) END(0x0FFFFFFF) WIDTH(32) }

// The HOST region will address the XILINX chip and the timing board

// Memory blocks need to be less than 2 Gig, and the total HOST space is almost 4 Gig.

// Arbitrarily, we create 7 segments of 1/4 Gig and 1 segment of 1/8 Gig.

// For this project, all of the hardware is in the first segment.

 HOST { TYPE(RAM) START(0x10000000) END(0x2FFFFFFF) WIDTH(32) }

 HOST1 { TYPE(RAM) START(0x30000000) END(0x4FFFFFFF) WIDTH(32) }

 HOST2 { TYPE(RAM) START(0x50000000) END(0x6FFFFFFF) WIDTH(32) }

 HOST3 { TYPE(RAM) START(0x70000000) END(0x8FFFFFFF) WIDTH(32) }

 HOST4 { TYPE(RAM) START(0x90000000) END(0xAFFFFFFF) WIDTH(32) }

 HOST5 { TYPE(RAM) START(0xB0000000) END(0xCFFFFFFF) WIDTH(32) }

 HOST6 { TYPE(RAM) START(0xD0000000) END(0xEFFFFFFF) WIDTH(32) }

 HOST7 { TYPE(RAM) START(0xF0000000) END(0xFFFFFFFF) WIDTH(32) }

Hardware can be accessed by using pointers, as the following code snippet that accesses the timing card shows. An ‘include’ file should be created that symbolically defines all of the various addresses.

main() {

int \*tim\_ptr = (int \*)0x10020000;

 int i;

 for(;;) {

 for (i=0; i<1024; i++) {

 \*tim\_ptr = i;

 }

 }

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Analog Cards*

For the BSM, FLM, and FLMA systems, each analog card has eight channels. Each channel has 512kW of memory space. There are no registers on the BSM/FLM analog cards. The FLMA cards have an accumulator module. Address line A[24] selects either memory space or accumulator space. Address lines A[23..22] select one of the four analog boards. Address lines A[21..19] select one of eight channels on a board. Address lines A[18..0] select a memory address.

ANALOG CARD 0 CHAN 0 0X08000000 - 0X0807FFFF (dec=134217728)

ANALOG CARD 0 CHAN 1 0X08080000 - 0X080FFFFF (dec=134742016)

ANALOG CARD 0 CHAN 2 0X08100000 - 0X0817FFFF (dec=135266304)

ANALOG CARD 0 CHAN 3 0X08180000 - 0X081FFFFF (dec=135790592)

ANALOG CARD 0 CHAN 4 0X08200000 - 0X0827FFFF (dec=136314880)

ANALOG CARD 0 CHAN 5 0X08280000 - 0X082FFFFF (dec=136839168)

ANALOG CARD 0 CHAN 6 0X08300000 - 0X0837FFFF (dec=137363456)

ANALOG CARD 0 CHAN 7 0X08380000 - 0X083FFFFF (dec=137887744)

ANALOG CARD 1 CHAN 0 0X08400000 - 0X0847FFFF (dec=138412032)

ANALOG CARD 1 CHAN 1 0X08480000 - 0X084FFFFF (dec=138936320)

ANALOG CARD 1 CHAN 2 0X08500000 - 0X0857FFFF (dec=139460608)

ANALOG CARD 1 CHAN 3 0X08580000 - 0X085FFFFF (dec=139984896)

ANALOG CARD 1 CHAN 4 0X08600000 - 0X0867FFFF (dec=140509184)

ANALOG CARD 1 CHAN 5 0X08680000 - 0X086FFFFF (dec=141033472)

ANALOG CARD 1 CHAN 6 0X08700000 - 0X0877FFFF (dec=141557760)

ANALOG CARD 1 CHAN 7 0X08780000 - 0X087FFFFF (dec=142082048)

ANALOG CARD 2 CHAN 0 0X08800000 - 0X0887FFFF (dec=142606336)

ANALOG CARD 2 CHAN 1 0X08880000 - 0X088FFFFF (dec=143130624)

ANALOG CARD 2 CHAN 2 0X08900000 - 0X0897FFFF (dec=143654912)

ANALOG CARD 2 CHAN 3 0X08980000 - 0X089FFFFF (dec=144179200)

ANALOG CARD 2 CHAN 4 0X08A00000 - 0X08A7FFFF (dec=144703488)

ANALOG CARD 2 CHAN 5 0X08A80000 - 0X08AFFFFF (dec=145227776)

ANALOG CARD 2 CHAN 6 0X08B00000 - 0X08B7FFFF (dec=145752064)

ANALOG CARD 2 CHAN 7 0X08B80000 - 0X08BFFFFF (dec=146276352)

ANALOG CARD 3 CHAN 0 0X08C00000 - 0X08C7FFFF (dec=146800640)

ANALOG CARD 3 CHAN 1 0X08C80000 - 0X08CFFFFF (dec=147324928)

ANALOG CARD 3 CHAN 2 0X08D00000 - 0X08D7FFFF (dec=147849216)

ANALOG CARD 3 CHAN 3 0X08D80000 - 0X08DFFFFF (dec=148373504)

ANALOG CARD 3 CHAN 4 0X08E00000 - 0X08E7FFFF (dec=148897792)

ANALOG CARD 3 CHAN 5 0X08E80000 - 0X08EFFFFF (dec=149422080)

ANALOG CARD 3 CHAN 6 0X08F00000 - 0X08F7FFFF (dec=149946368)

ANALOG CARD 3 CHAN 7 0X08F80000 - 0X08FFFFFF (dec=150470656)

ACCUMULATOR

ANALOG CARD 0 0x0A000000 (dec=167772160)

ANALOG CARD 1 0x0A400000 (dec=171966464)

ANALOG CARD 2 0x0A800000 (dec=176160768)

ANALOG CARD 3 0x0AC00000 (dec=180355072)

## Refer to the section on ACCUMULATOR PROGRAMMING for details.

---------------------------------------------------------------------------------------------------------------------

For the BPM system, each analog card has two channels. Each channel has 512kW of memory and 1 gain register. Address line A[24] selects either memory space or register space. Address lines A[21..20] select one of the four analog boards. Address line A[19] selects one of two channels on a board. Address lines A[18..0] select either a memory address or a register address.

MEMORY

ANALOG CARD 0 CHAN 0 0X08000000 - 0X0807FFFF (dec=134217728)

ANALOG CARD 0 CHAN 1 0X08080000 - 0X080FFFFF (dec=134742016)

ANALOG CARD 1 CHAN 0 0X08100000 - 0X0817FFFF (dec=135266304)

ANALOG CARD 1 CHAN 1 0X08180000 - 0X081FFFFF (dec=135790592)

ANALOG CARD 2 CHAN 0 0X08200000 - 0X0827FFFF (dec=136314880)

ANALOG CARD 2 CHAN 1 0X08280000 - 0X082FFFFF (dec=136839168)

ANALOG CARD 3 CHAN 0 0X08300000 - 0X0837FFFF (dec=137363456)

ANALOG CARD 3 CHAN 1 0X08380000 - 0X083FFFFF (dec=137887744)

GAIN REGISTERS

ANALOG CARD 0 GAIN 0 0X09000000 (dec=150994944)

ANALOG CARD 0 GAIN 1 0X09080000 (dec=151519232)

ANALOG CARD 1 GAIN 0 0X09100000 (dec=152043520)

ANALOG CARD 1 GAIN 1 0X09180000 (dec=152567808)

ANALOG CARD 2 GAIN 0 0X09200000 (dec=153092096)

ANALOG CARD 2 GAIN 1 0X09280000 (dec=153616384)

ANALOG CARD 3 GAIN 0 0X09300000 (dec=154140672)

ANALOG CARD 3 GAIN 1 0X09380000 (dec=154664960)

------------------------------------------------------------------------------------------------------------------------

For the BSM4, each carrier board has eight analog channels. Each channel has 1MW of memory space. There are also registers on the BSM4 analog cards. Address line A[25] selects either memory space or register space. Address lines A[24..23] select one of the four carrier boards. Address lines A[22..20] select one of eight channels on a carrier board. Address lines A[19..0] select a memory or register address.

MEMORY

CARRIER CARD 0 CHAN 0 0X08000000 - 0X080FFFFF (dec=134217728)

CARRIER CARD 0 CHAN 1 0X08100000 - 0X081FFFFF (dec=135266304)

CARRIER CARD 0 CHAN 2 0X08200000 - 0X082FFFFF (dec=136314880)

CARRIER CARD 0 CHAN 3 0X08300000 - 0X083FFFFF (dec=137363456)

CARRIER CARD 0 CHAN 4 0X08400000 - 0X084FFFFF (dec=138412032)

CARRIER CARD 0 CHAN 5 0X08500000 - 0X085FFFFF (dec=139460608)

CARRIER CARD 0 CHAN 6 0X08600000 - 0X086FFFFF (dec=140509184)

CARRIER CARD 0 CHAN 7 0X08700000 - 0X087FFFFF (dec=141557760)

CARRIER CARD 1 CHAN 0 0X08800000 - 0X088FFFFF (dec=142606336)

CARRIER CARD 1 CHAN 1 0X08900000 - 0X089FFFFF (dec=143654912)

CARRIER CARD 1 CHAN 2 0X08A00000 - 0X08AFFFFF (dec=144703488)

CARRIER CARD 1 CHAN 3 0X08B00000 - 0X08BFFFFF (dec=145752064)

CARRIER CARD 1 CHAN 4 0X08C00000 - 0X08CFFFFF (dec=146800640)

CARRIER CARD 1 CHAN 5 0X08D00000 - 0X08DFFFFF (dec=147849216)

CARRIER CARD 1 CHAN 6 0X08E00000 - 0X08EFFFFF (dec=148897792)

CARRIER CARD 1 CHAN 7 0X08F00000 - 0X08FFFFFF (dec=149946368)

CARRIER CARD 2 CHAN 0 0X09000000 - 0X090FFFFF (dec=150994944)

CARRIER CARD 2 CHAN 1 0X09100000 - 0X091FFFFF (dec=152043520)

CARRIER CARD 2 CHAN 2 0X09200000 - 0X092FFFFF (dec=153092096)

CARRIER CARD 2 CHAN 3 0X09300000 - 0X093FFFFF (dec=154140672)

CARRIER CARD 2 CHAN 4 0X09400000 - 0X094FFFFF (dec=155189248)

CARRIER CARD 2 CHAN 5 0X09500000 - 0X095FFFFF (dec=156237824)

CARRIER CARD 2 CHAN 6 0X09600000 - 0X096FFFFF (dec=157286400)

CARRIER CARD 2 CHAN 7 0X09700000 - 0X097FFFFF (dec=158334976)

CARRIER CARD 3 CHAN 0 0X09800000 - 0X098FFFFF (dec=159383552)

CARRIER CARD 3 CHAN 1 0X09900000 - 0X099FFFFF (dec=160432128)

CARRIER CARD 3 CHAN 2 0X09A00000 - 0X9AFFFFF (dec=161480704)

CARRIER CARD 3 CHAN 3 0X09B00000 - 0X9BFFFFF (dec=162529280)

CARRIER CARD 3 CHAN 4 0X09C00000 - 0X9CFFFFF (dec=163577856)

CARRIER CARD 3 CHAN 5 0X09D00000 - 0X9DFFFFF (dec=164626432)

CARRIER CARD 3 CHAN 6 0X09E00000 - 0X9EFFFFFF (dec=165675008)

CARRIER CARD 3 CHAN 7 0X09F00000 - 0X9FFFFFFF (dec=166723584)

REGISTERS

CARRIER CARD 0 REG 0 0X0A000000 - 0X0A0FFFFF (dec=167772160)

CARRIER CARD 0 REG 1 0X0A100000 - 0X0A1FFFFF (dec=168820736)

CARRIER CARD 0 REG 2 0X0A200000 - 0X0A2FFFFF (dec=169869312)

CARRIER CARD 0 REG 3 0X0A300000 - 0X0A3FFFFF (dec=170917888)

CARRIER CARD 0 REG 4 0X0A400000 - 0X0A4FFFFF (dec=171966464)

CARRIER CARD 0 REG 5 0X0A500000 - 0X0A5FFFFF (dec=173015040)

CARRIER CARD 0 REG 6 0X0A600000 - 0X0A6FFFFF (dec=174063616)

CARRIER CARD 0 REG 7 0X0A700000 - 0X0A7FFFFF (dec=175112192)

CARRIER CARD 1 REG 0 0X0A800000 - 0X0A8FFFFF (dec=176160768)

CARRIER CARD 1 REG 1 0X0A900000 - 0X0A9FFFFF (dec=)

CARRIER CARD 1 REG 2 0X0AA00000 - 0X0AAFFFFF (dec=)

CARRIER CARD 1 REG 3 0X0AB00000 - 0X0ABFFFFF (dec=)

CARRIER CARD 1 REG 4 0X0AC00000 - 0X0ACFFFFF (dec=)

CARRIER CARD 1 REG 5 0X0AD00000 - 0X0ADFFFFF (dec=)

CARRIER CARD 1 REG 6 0X0AE00000 - 0X0AEFFFFF (dec=)

CARRIER CARD 1 REG 7 0X0AF00000 - 0X0AFFFFFF (dec=)

CARRIER CARD 2 REG 0 0X0B000000 - 0X0B0FFFFF (dec=)

CARRIER CARD 2 REG 1 0X0B100000 - 0X0B1FFFFF (dec=)

CARRIER CARD 2 REG 2 0X0B200000 - 0X0B2FFFFF (dec=)

CARRIER CARD 2 REG 3 0X0B300000 - 0X0B3FFFFF (dec=)

CARRIER CARD 2 REG 4 0X0B400000 - 0X0B4FFFFF (dec=)

CARRIER CARD 2 REG 5 0X0B500000 - 0X0B5FFFFF (dec=)

CARRIER CARD 2 REG 6 0X0B600000 - 0X0B6FFFFF (dec=)

CARRIER CARD 2 REG 7 0X0B700000 - 0X0B7FFFFF (dec=)

CARRIER CARD 3 REG 0 0X0B800000 - 0X0B8FFFFF (dec=)

CARRIER CARD 3 REG 1 0X0B900000 - 0X0B9FFFFF (dec=)

CARRIER CARD 3 REG 2 0X0BA00000 - 0X0BAFFFFF (dec=)

CARRIER CARD 3 REG 3 0X0BB00000 - 0X0BBFFFFF (dec=)

CARRIER CARD 3 REG 4 0X0BC00000 - 0X0BCFFFFF (dec=)

CARRIER CARD 3 REG 5 0X0BD00000 - 0X0BDFFFFF (dec=)

CARRIER CARD 3 REG 6 0X0BE00000 - 0X0BEFFFFF (dec=)

CARRIER CARD 3 REG 7 0X0BF00000 - 0X0BFFFFFF (dec=)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *FLASH Memory*

FLASH 0X0C000000 - 0X0C07FFFF (dec=201326592)

The FLASH memory is 512k by 8-bits, using an Atmel AT49LV040 chip. Its primary use is to store the DSP code.

Unlike the FLASH in other BPM projects, this chip does not have multiple sectors that can be individually erased. If the FLASH is to be used for non-volatile storage of anything other than the DSP code, the user’s program will need to read and save the permanent information before reprogramming the memory. The saved information will then need to be written back to the FLASH after it has been erased and is ready for new DSP code.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Static RAM*

STATIC RAM 0X0C080000 - 0X0C0FFFFF (dec=201850880)

The static RAM is 512k by 32-bits.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Vector and Packet Support*

VECTOR ADDRESS TABLE 0x10000000 - 0x100001FF (dec=268435456)

The vector address table holds 512 addresses, each 32-bits wide. Xbus vector commands (‘vxgetn’ and ‘vxputn’) specify the first vector and the number of vectors to access. The vector address table maps Xbus vectors to hardware addresses.

The following vectors are currently defined:

0x078: FLASH (0x0C005555)

0x079: FLASH (0x0C002AAA)

0x07A: FLASH (0x0C005555)

0x07B: FLASH (0x0C005555)

0x07C: FLASH (0x0C002AAA)

0x07D: FLASH (0x0C005555)

0x07E: direct address register (0x10040000)

0x07F: This is a special vector number. The actual address comes from

the 'direct\_adr' register.

The standard MPM database address nodes (like “CBPM ADR TST”) are initialized with vector number 0x7E. They access the ‘DIRECT\_ADR’ register.

The standard MPM database data nodes (like “CBPM DAT TST”) are initialized with vector number 0x7F. Operations to these data nodes use the address that was programmed thru the address node.

As an example, to write ‘some\_data” to “some\_address” in the module associated with element 2 of the nodes “CBPM ADR TST” and “CBPM DAT TST”, the control system program makes the following calls:

 call vxputn(‘CBPM ADR TST’, 2, 2, some\_address)

 call vxputn(‘CBPM DAT TST’, 2, 2, some\_data)

Vector 0X078 thru 0x07D (120 thru 125) are used for flash programming. To erase the FLASH chip, write the following data to the corresponding vector:

0x078=aa 0x079=55 0x07a=80 0x07b=aa 0x07c=55 0x07d=10

To program the FLASH chip with a vector operation, write the following data to the corresponding vector (PA is the address to program, PD is the data to program):

0x07b=aa 0x07c=55 0x07d=a0 0x07e=PA 0x07f=PD

PACKET START ADDRESS TABLE 0x10001000 - 0x100011FF (dec=268439552)

PACKET MORE ADDRESS TABLE 0x10001800 - 0x100019FF (dec=268441600)

##  PACKET SIZE TABLE 0x10010000 - 0x100101FF (dec=268500992)

The ‘packet start address table’ and ‘packet more address table’ each hold 512 addresses, each 32-bits wide. The packet size table holds 512 values, each 12-bits wide.

The ‘packet start address table’ holds the first address of each data structure or block that is accessed for a given packet tag. The address is loaded into a counter. After each access, the counter is incremented and the result is written into the ‘packet more address table’. After the amount of data specified in the ‘packet size table’ has been transferred, the address stored in the ‘packet more address table’ will be the address of the next piece of data in the data structure. If another packet operation is performed and the tag is offset by 2048, the first address will be retrieved from the ‘packet more address table’. This scheme allows for access to blocks of memory that are larger than the maximum size of a single packet by simply using the regular tag for the first block and the offset tag for all of the remaining blocks.

A constraint imposed by this scheme is that the size of any data structure must be a multiple of the size written in the ‘packet size table’. If a 260 word structure needs to be transferred, one can either use a 256 word packet size and pad the structure out to 512 words, or use a 130 word packet size. Additionally, the ‘packet more address table’ is read-only.

Xbus packet commands (‘vugetn’ and ‘vuputn’) specify the packet tag to use for a data transfer. The packet address tables map Xbus packet numbers to the first hardware address involved in the transfer. Packet tags from 1 thru 2047 will find the first address in the ‘packet start address table’. Packet tags from 2049 thru 4095 will find the first address in the ‘packet more address table. The packet size table specifies how many 32-bit words to transfer for a ‘read’ (vugetn) operation. For ‘write’ (vuputn) operations, the number of words written determines the transfer size.

Generally, the DSP will initialize the address and size tables with the addresses and sizes of internal data structures that the control system needs to access. An exception is for packet 0x1ff (511). When the DSP is initializing the ‘packet start address table’ with the address of memory locations that are internal to the DSP, the offset of 0x02000000 must be added to the actual DSP address. Refer to DSP (from the XBUS or Ethernet perspective).

Packet 0x1FF (511) is reserved for FLASH programming. Any data written to this packet will cause the FLASH programming sequence in the XILINX chip to be invoked. The procedure to program FLASH with packet operations is:

! write the address of PACKET ADDRESS TABLE entry #511 to the address node

call vxputn(‘CBPM ADR TST, 2, 2, ‘100011ff’x)

!write the next FLASH address to program to the data node

call vxputn(‘CBPM DAT TST’, 2, 2, flash\_adr)

! write the address of PACKET SIZE TABLE entry #511 to the address node

call vxputn(‘CBPM ADR TST, 2, 2, ‘100101ff’x)

!write the size of the packet to the data node

call vxputn(‘CBPM DAT TST’, 2, 2, size)

!send the packet of data with packet tag #511

call vuputn(‘CBPM PKT TST’, 2, 2, data\_array, 511, size)

!note: the ‘data\_array’ is a longword (32-bit) array with one byte per longword

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Timing Board (72 MHz only boards)*

##  TIMING BOARD 0X10020000 - 0X100200FF (dec=268566528)

These registers control delay settings on the timing board. All registers are 10-bits, and are write-only. The timing board has two timing blocks: A and B. Each block has four clock outputs, one for each analog card. The four clock outputs of a block consist of a global delay that is the sum of two global delays settings (common to all four outputs), plus a specific delay setting for each channel. The global delay setting can span the 14 nsec bunch spacing. The channel delays are intended to compensate for cable length variations and should generally be within +/- 1.5 nsec of each other. The delay for each chip can vary from 3.2ns to 14.8ns in 10ps increments.

BSM/FLM (Block A Only; Block B not used for BSM/FLM)

Offset Contents
0 Block A, Global Delay 1
1 Block A, Global Delay 0
2 Block A, Card 3 Delay  (chan 24 - 31)
3 Block A, Card 2 Delay  (chan 16 - 23)
4 Block A, Card 1 Delay  (chan 8 - 15)
5 Block A, Card 0 Delay  (chan 0 – 7)

BPM (Block A for one species; Block B for the other)

Offset Contents
0 Block A, Global Delay 1
1 Block A, Global Delay 0
2 Block A, Card 3 Delay
3 Block A, Card 2 Delay
4 Block A, Card 1 Delay
5 Block A, Card 0 Delay

10 Block A, Turns Marker Delay

11 Block A, Clock Frequency

12 Block A, Clock Width

8 Block B, Global Delay 1
9 Block B, Global Delay 0
A Block B, Card 3 Delay
B Block B, Card 2 Delay
C Block B, Card 1 Delay
D Block B, Card 0 Delay

18 Block B, Turns Marker Delay

19 Block B, Clock Frequency

1A Block B, Clock Width

1F Common Turns Marker Delay

A typical calibration scheme is to set the channel delays to mid-scale and adjust the global delay for optimal results looking at the sum of all four channels. Then increment or decrement the individual channel delays to optimize the results for each channel.

## *Timing Board (250 MHz only boards)*

##  TIMING BOARD 0X10020000 - 0X100200FF (dec=268566528)

ADD PROGRAMMING DATA FOR MELLER’S BOARD

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Auxiliary Board*

## The auxiliary board is not accessible from the DSP board. Instead, it contains a ColdFire ‘dimm’ CPU module that can access all of the registers and memory described in this document. Refer to the section on COLDFIRE PROGRAMMING for details.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Registers*

DIRECT\_ADR 0X10040000 (dec=268697600)

This register holds the 32-bit address used for XBUS vector operations when the vector equals #127. This register is programmed from the XBUS by specifying vector #126. Refer to the discussion of VECTOR ADDRESS TABLE. It is not generally accessed by the DSP.

DSP\_RESET 0X10040001 (dec=268697601)

This register controls the /DSP\_RESET pin on the DSP\_CHIP. If data bit D0 is zero, the reset signal is asserted. This stops the DSP. When data bit D0 changes to a one, the DSP booting and configuration process begins.

When the board is initialized (power-up or front panel pushbutton), data bit D0 will be set to zero. This differs from the DIMM\_RESET, which is set to one.

DATA\_ACQ 0X10040002 (dec=268697602)

This register controls acquisition of data by the analog cards.

bit 0: ACQ\_MODE (READ/WRITE)

[BPM4, ERL\_BPM, BSM]

Once the acquisition parameters have been initialized, setting the ACQ\_MODE bit to ‘1’ will switch control of the analog card signals to the acquisition controller and start the collection of data. This bit must be cleared to ‘0’ before programmed readout of the analog boards can occur. It can be cleared at any time. If acquisition is in progress, it will be halted.

[BPM4\_TLINE]

Once the acquisition parameters have been initialized, setting the ACQ\_MODE bit to ‘1’ will, along with the proper trigger conditions of bits 6:4, signal the analog cards to start the collection of data. This bit must be cleared to ‘0’ before programmed readout of the analog boards can occur. It can be cleared at any time. If acquisition is in progress, it will be halted.

[BSM4]

Once the acquisition parameters have been initialized, setting the ACQ\_MODE bit to ‘1’ will, along with the proper trigger conditions of bits 6:4, signal the analog cards to start the collection of data. Registers on the analog cards can still be accessed. Memory accesses are locked out and will return meaningless data. This bit must be set back to ‘0’ before another acquisition can be activated.

bit 1: ACQ\_ACTIVE (READ ONLY)

[BPM4, ERL\_BPM, BSM]

 This bit shows the status of data collection. A ‘1’ after setting ACQ\_MODE indicates that a turn marker has been received and data is being collected. This bit will revert back to ‘0’ when acquisition is complete or when ACQ\_MODE is cleared to ‘0’.

[BSM4, BPM4\_TLINE]

This bit is unused and will always be read as zero.

bit 2: ACQ\_DONE (READ ONLY)

[BPM4, ERL\_BPM, BSM]

This bit shows the status of data collection. A ‘1’ after setting ACQ\_MODE indicates that data collection is finished. This bit will revert back to ‘0’ when ACQ\_MODE is cleared to ‘0’.

[BSM4, BPM4\_TLINE]

This bit is unused and will always be read as zero. The front-end boards must be polled for the DONE condition.

bit 3: ACQ\_CONT (READ/WRITE)

[BPM4, ERL\_BPM, BSM]

This bit controls single-shot vs. continuous mode data acquisition. Once the acquisition parameters have been initialized, setting the ACQ\_CONT bit to ‘1’ at the same time that the ACQ\_MODE bit is set to ‘1’ will cause the system to acquire data continuously. If the ACQ\_CONT’ bit is later set to ‘0’ while ACQ\_MODE is left at ‘1’, acquisition will continue until ACQ\_TURN\_REQ additional turns have been acquired (post-trigger mode).

If the ACQ\_CONT bit is ‘0’ when the ACQ\_MODE bit is set to ‘1’, then the system will acquire a single shot of data as controlled by the contents of ACQ\_TURN\_REQ.

For single-shot mode, write 0x01 to the DATA\_ACQ register and monitor bit 2 for completion. For continuous data acquisition, write 0x09 to the DATA\_ACQ register. Then if you want acquisition to stop immediately, write 0x00 to the DATA\_ACQ register. If you want is to continue for ACQ\_TURN\_REQ additional turns, write 0x01 to the DATA\_ACQ register.

[BSM4, BPM4\_TLINE]

This bit is unused and will always be read as zero. There is not a continuous mode.

bit 6:4: TRIG\_CONT (READ/WRITE)

[BPM4, ERL\_BPM, BSM]

These bits are unused and will always be read as zero

[BSM4, BPM4\_TLINE]

bit 4: use hardware trigger 0

bit 5: use hardware trigger 1

bit 6: use enable bit

'enable' is bit-1 in the 8-bit command word sent on the timing cable

'trigger 0' and 'trigger 1' are the two hardware trigger bits sent out on the timing cable, usually 60H and Injection triggers.

A state machine will control acquisition according to the following flow:

IF (ACQ\_MODE == 1)

THEN // bit-0 has been set

 IF (use\_enable)

 THEN wait for enable bit

 ELSE continue

 END IF

 IF (use\_trig1)

 THEN wait for trigger 1

 ELSE continue

 END IF

 IF (use\_trig0)

 THEN wait for trigger 0

 ELSE continue

 END IF

 WAIT for new turn marker // systems all go on next turn mark

 ASSERT adc\_sync // drive ana\_mem\_cs[7]

END IF

If DATA\_ACQ[6..4] are all zero, then the system will operate with an immediate acquisition as soon as you write a '1' to bit-0. Notice that there is a priority on the two hardware triggers. This is how the CBPM DSP code operates.

bit 7: DCM\_LOCKED (READ ONLY)

[BPM4, ERL\_BPM, BSM]

NOTE: This was “bit 4” in previous versions

The LOCKED signal activates after the DCM has achieved lock. To achieve lock, the DCM may need to sample several thousand clock cycles.  After the DCM achieves lock, the LOCKED signal goes high.  To guarantee that the system clock is established prior to the device waking up, the DCM can delay the completion of the device configuration process until after the DCM locks. Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

[BSM4, BPM4\_TLINE]

These bits are unused and will always be read as zero

bit 10:8: DCM\_STATUS (READ ONLY)

[BPM4, ERL\_BPM, BSM]

NOTE: This was “bit 7:5” in previous versions

The 3 signals connect to the status register in the DCM. Bit 8 (STATUS[0]) indicates the overflow of the phase shift numerator and that the absolute delay range of the phase shift delay line is exceeded.  Bit 9 (STATUS[1]) indicates the loss of the input clock, CLKIN, to the DCM.  Bit 10 (STATUS[2]) indicates that CLKFX has stopped.

Under normal operating conditions, the data read from bits [10:7] should be 0x1.

[BSM4, BPM4\_TLINE]

 These bits are unused and will always be read as zero

bit 20:11: Acquisition Trigger State Machine (READ ONLY)

[BPM4, ERL\_BPM, BSM]

These bits are unused and will always be read as zero

[BSM4, BPM4\_TLINE]

These bits show the current state of the one-hot trigger state machine. They can be useful for debugging, like for finding out that the device appears to hang because trigger 0 is never received.

bit state

11 IDLE

12 CHK\_CABLE\_ENABLE

13 WAIT\_CABLE\_ENABLE

14 CHK\_TRIG1\_ENABLE

15 WAIT\_TRIG1\_ENABLE

16 CHK\_TRIG0\_ENABLE

17 WAIT\_TRIG0\_ENABLE

18 TST\_NEW\_TURN

19 WAIT\_NEW\_TURN

20 RUN

RECV\_ERR 0X10040003 (dec=268697603)

This register holds error information from the serial Xbus receiver. Its usefulness is questionable, since we would need to use the serial Xbus to read it.

GLOBAL\_TURN\_CNT 0X10040004 (dec=268697604)

This read-only register holds a 20-bit turn counter. The counter can be reset globally by a command from the control system. This allows the counters in every module to be synchronized. With 20 bits, the counter wraps around every 2.5 seconds. The size of the counter could be extended, but then it would not be compatible with the counter in the first generation DSP modules.

The transition of the command byte in GLOBAL\_TURN\_DAT from not being equal to 0x01 to being equal to 0x01 clears the turn count register.

GLOBAL\_TURN\_DAT 0X10040005 (dec=268697605)

This read-only register holds the 27-bit data stream that is sent with the 24 MHz (25 MHz for ERL) system clock on every turn. The packing of data is:

bit 0: hardware trigger 0

bit 1: hardware trigger 1

bit 9..2: command (bit 9=MSB, bit 2=LSB)

 bit 18..10: vertical phase data (bit 18=MSB, bit 10=LSB)

 bit 27..19: horizontal phase data (bit 27=MSB, bit 19=LSB)

A signal will be provided to the DSP every time GLOBAL\_TURN\_DAT is updated. The DSP may need to examine trigger bits or the command word, and it may need to store the phase data.

TOD\_SECONDS 0X10040006 (dec=268697606)

This register holds a 17-bit counter that is incremented once per second. It can provide a time-of-day timestamp. The current time can be written to this register, and should be written at least once per day, as well as whenever power is cycled.

This register is currently decoupled from the GLOBAL\_TURN\_CNT register and does not change when the GLOBAL\_TURN\_CNT in all modules is synchronized.

SEMAPHORE 0X10040007 (dec=268697607)

This register is used to provide a semaphore to guard 'critical sections' for xbus vs. dsp access. When read, it will return its current state. The act of reading it will change its state to '1'. The act of writing to it will change its state to '0'. If a '0' is read, that indicates that the semaphore was free, and it is now owned by the reader. If a '1' is read, that indicates that the semaphore was already owned, and the reader should try again.

DIMM\_RESET 0X10040008 (dec=268697608)

This register controls the /DIMM\_RESET pin to the DIMM module on the auxiliary I/O board. If data bit D0 is zero, the reset signal is asserted. This stops the DIMM. When data bit D0 changes to a one, the DIMM booting and configuration process begins.

When the board is initialized (power-up or front panel pushbutton), data bit D0 will be set to one. This differs from the DSP\_RESET, which is set to zero.

UNUSED 0X10040009 (dec=268697609)

UNUSED 0X1004000A (dec=268697610)

UNUSED 0X1004000B (dec=268697611)

UNUSED 0X1004000C (dec=268697612)

UNUSED 0X1004000D (dec=268697613)

UNUSED 0X1004000E (dec=268697614)

UNUSED 0X1004000F (dec=268697615)

These registers may be defined in the future.

BUNCH\_PATTERN\_A 0X10040010 (dec=268697616)

BUNCH\_PATTERN\_B 0X10040011 (dec=268697617)

BUNCH\_PATTERN\_C 0X10040012 (dec=268697618)

BUNCH\_PATTERN\_D 0X10040013 (dec=268697619)

BUNCH\_PATTERN\_E 0X10040014 (dec=268697620)

BUNCH\_PATTERN\_F 0X10040015 (dec=268697621)

These 6 registers hold the bunch pattern.

CESR: The revolution time of CESR is divided into 183 slots that are spaced 14 nsec apart. To store the data for a particular slot, a ‘1’ is programmed into the BUNCH\_PATTERN register. No data is stored when the register value is ‘0’. The first 5 registers are 32-bit and the 6th is 23 bits.

ERL: The “revolution” time of the ERL is divided into 122 slots that are spaced 20 nsec apart. To store the data for a particular slot, a ‘1’ is programmed into the BUNCH\_PATTERN register. No data is stored when the register value is ‘0’. The first 3 registers are 32-bit and the 4th is 26 bits.

These registers must be loaded prior to collecting data. Their contents will be used to initialize a large shift register.

The LSB of BUNCH\_PATTERN\_A will control storage of the first slot after the turn marker. Higher bits control slots that occur later in time. Notice that the LSB of BUNCH\_PATTERN\_A *does not* correspond to “train 1, bunch 1”. The user must start with a pattern that has “train 1 bunch 1” in the first location, then rotate that pattern until the ‘train 1 bunch 1” bit matches the arrival time of “train 1 bunch 1” at a given detector, relative to the time that the turn marker arrives at the same detector. A timing calibration routine should be provided which can determine the appropriate shift at a given detector.

ADC\_MEM\_FIRST 0X10040016 (dec=268697622)

This register holds the initial memory address for an acquisition sequence. This address will be loaded into the address counter at the start of every acquisition sequence.

The address counter is a 19-bit counter that generates memory addresses. This register also holds 19 bits.

ACQ\_TURN\_REQ 0X10040017 (dec=268697623)

This register holds the requested number of turns to be acquired. This count will be loaded into the acquisition turn counter at the start of every acquisition block.

The acquisition turn counter is a 20-bit counter that controls how many turns of data are stored in memory. The counter will initially be loaded from ACQ\_TURN\_REQ. It will decrement every time a turn marker arrives (unless ACQ\_SKIP\_CNT is not zero). When the counter has decremented to zero, the acquisition sequence is finished.

Remember that more than one data point can be acquired per turn (up to 183 for CESR and 122 for ERL). To avoid memory address wrap-around, be sure that the product of ACQ\_TURN\_REQ time the number of ‘1’s in the BUNCH\_PATTERN is less than 512k.

ACQ\_SKIP\_CNT 0X10040018 (dec=268697624)

This function has not been implemented yet

This register controls how many turns are skipped over between turns that are stored in memory. It is an 8-bit register, allowing up to 255 turns to be skipped. It is initialized to zero, meaning that no turns are skipped. A skip count value of 1 means that every-other turn is stored. This is a different meaning than in the 1st generation of DSP-based BPMs.

TURN\_MARK\_DELAY 0X10040019 (dec=268697625)

This register controls the arrival time of the turn marker used by the acquisition logic. A ‘0’ specifies ‘normal’ arrival time, while a ‘1’ specifies ‘delayed’ arrival.

The turn marker is synchronous to the 24 MHz CESR clock (25 MHz ERL) extracted from the coax timing cable. The turn marker is used in logic that is synchronous to the ADC clock. However, the delay setting of the ADC clock can be anywhere in a 14 nsec. Window (20 nsec ERL), resulting in a possible situation where the setup and hold times of the turn marker relative to the ADC clock are not met. This register provides a way to shift the time that the setup and hold is not met by 7 nsec for CESR or 10 nsec for ERL. Some testing will need to be performed to decide what range of the global delay can be safely used with each setting of the TURN\_MARK\_DELAY.

ADC\_CLOCK\_CONTROL 0X1004001A (dec=268697626)

This register is present only on the BPM modules or other modules that use a 2-channel ADC card with independent clocking. It does not exist on the BSM/FLM modules, nor on other modules that use 8-channel ADC cards with ganged clocking.

This register is used to select which ADC clock drives the Xilinx electronics, and to set the clock phase for clocking the data from the ADCs into the ADC data registers and then writing the data to memory.

The timing card generates two ADC clocks, and the phase of each clock relative to the fixed incoming CESR clock can be set independently on the timing card. The data from each ADC is then clocked into a register. Finally, the data from both registers is written to memory. The timing of clocking the data into the registers and writing the data to memory can be adjusted in ¼ period intervals. The trick is to choose one of the ADC clocks and the appropriate setting for the register and memory timing so that all setup and hold times are met.

This register is divided into the following bit fields:

bits 1..0 = ADC0\_REG\_CLK:

Selects one of four delay settings relative to the chosen ADC clock for clocking data from ADC #0 into the data register. With a 72 MHz CESR sample rate, each increment in the settings of these bits represents an additional delay of 3.5 nsec. With a 50 MHz ERL sample rate, each increment in the settings of these bits represents an additional delay of 5.0 nsec.

bits 3..2 = ADC1\_REG\_CLK

Selects one of four delay settings relative to the chosen ADC clock for clocking data from ADC #1 into the data register. With a 72 MHz CESR sample rate, each increment in the settings of these bits represents an additional delay of 3.5 nsec. With a 50 MHz ERL sample rate, each increment in the settings of these bits represents an additional delay of 5.0 nsec.

bits 5..4 = MEM\_WR\_CLK

Selects one of four delay settings relative to the chosen ADC clock for writing data from both ADC data registers into the memory. With a 72 MHz CESR sample rate, each increment in the settings of these bits represents an additional delay of 3.5 nsec. With a 50 MHz ERL sample rate, each increment in the settings of these bits represents an additional delay of 5.0 nsec.

bits 6 = ADC\_CLK\_SEL

Chooses which ADC clock is the master clock for clocking data into the ADC data registers and writing it to memory. A value of ‘0’ corresponds to ADC #0 and timing card block ‘A’. A value of ‘1’ corresponds to ADC #1 and timing card block ‘B’.

Refer to the section on *Setting the ADC\_CLOCK\_CONTROL Register* for examples of how to set up this register.

RESERVED 0X1004001B (dec=268697627)

RESERVED 0X1004001C (dec=268697628)

RESERVED 0X1004001D (dec=268697629)

ACQ\_TURN\_CNT 0X1004001E (dec=268697630)

This is a read-only register that shows the current value of the acquisition turn counter. Since the turn counter counts backwards, the number of turns of data in memory can be calculated by subtracting the value of ACQ\_TURN\_CNT from the initial value specified in ACQ\_TURN\_REQ.

NEXT\_MEM\_ADR 0X1004001F (dec=268697631)

This is a read-only register that shows the memory location where the next sample will be stored.

TEMPERATURE\_CSR 0X10040020 (dec=268697632)

This register controls the readout of the AD7814 temperature chips. Writing any data to this register will initiate reading of the serial data from all six chips. Bit 0 of this register will be a ‘1’ while serial data is being transferred from the temperature chips. Once the transfer is complete, bit 0 will change back to a ‘0’. This will indicate that the data is stable and can be read from the individual temperature registers.

The readout time from the AD7814 chip is about 2 usec. Since the DSP can access data more quickly than that, the DSP should check the status bit. Accesses from the XBUS are slow enough so that the data will be ready by the time one tries to access it.

ANA\_0\_TEMPERATURE 0X10040021 (dec=268697633)

ANA\_1\_TEMPERATURE 0X10040022 (dec=268697634)

ANA\_2\_TEMPERATURE 0X10040023 (dec=268697635)

ANA\_3\_TEMPERATURE 0X10040024 (dec=268697636)

TIMING\_TEMPERATURE 0X10040025 (dec=268697637)

DIGITAL\_TEMPERATURE 0X10040026 (dec=268697638)

These read-only registers contain the data read from the AD7814 temperature chips. The data is in units of 0.25 degrees centigrade. It is assumed that the temperature will never be below 32 degrees F, so this number will always be positive.

MODULE\_TYPE 0X10040027 (dec=268697639)

MAJOR\_REV 0X10040028 (dec=268697640)

MINOR\_REV 0X10040029 (dec=268697641)

These are read-only registers that show the type of Xilinx code that is running on a module, as well as the major and minor revision numbers. For this document:

MODULE\_TYPE

= 1 BSM

= 2 BPM

= 3 FLM

= 4 FLMA

= 5 ERL\_BPM

= 6 BPM4 w/FPGA

= 7 BSM4

=8 BPM4\_TLINE

MAJOR\_REV = 5

MINOR\_REV = 11

These values are defined in a ‘Constants.txt’ file that is included by the main Verilog program for each product type. They should correspond to notes in the file “VersionChanges.txt”

RESERVED 0X1004002A (dec=268697642)

RESERVED 0X1004002B (dec=268697643)

RESERVED 0X1004002C (dec=268697644)

RESERVED 0X1004002D (dec=268697645)

RESERVED 0X1004002E (dec=268697646)

RESERVED 0X1004002F (dec=268697647)

CUR\_MON\_STAT 0X10040030 (dec=268697648)

CUR\_MON\_WR\_DAT 0X10040031 (dec=268697649)

CUR\_MON\_WR\_ADR 0X10040032 (dec=268697650)

CUR\_MON\_RD\_ADR 0X10040033 (dec=268697651)

CUR\_MON\_RD\_DAT 0X10040034 (dec=268697652)

The current monitor board interface uses indirect addressing to access data within the current monitor chip. First, the program must read from the CUR\_MON\_STAT register to be sure that the serial link from the DSP board to the current monitor board is idle, and is not in a timeout error condition. A value of '0' (‘0x00000000’) indicates that the interface is ready to receive commands and/or data. A value of '1' (‘0x00000001’) indicates that the interface is busy. A value of '-32768' (‘0xffff8000’) indicates that a timeout has occurred.

If the interface is busy, any data written to the CUR\_MON\_WR\_DAT, CUR\_MON\_WR\_ADR, or CUR\_MON\_RD\_ADR registers will be ignored. Data read from the CUR\_MON\_RD\_DAT register will be undefined.

If a timeout has occurred, the only way to clear the error code and re-enable the interface is to write a value of ‘-1’ (0xffffffff) to the CUR\_MON\_STAT register. This will reset all of the logic on the DSP board side of the current monitor board interface. The timeout time is about 2.5 usec. A timeout is a serious condition that requires investigation.

Do not confuse the CUR\_MON\_STAT register with the CSR register on the current monitor board. The CUR\_MON\_STAT register is solely concerned with the communications link to the current monitor board.

To write data, the data is written to the CUR\_MON\_WR\_DAT register. The destination address is then written to the CUR\_MON\_WR\_ADR register. The serial transfer will then start and the BUSY bit in the CUR\_MON\_STAT register will be set. When the transfer is complete, the BUSY bit will be cleared.

To read data, the destination address is written to the CUR\_MON\_RD\_ADR register. The serial transfer will then start and the BUSY bit in the CUR\_MON\_STAT register will be set. When the transfer is complete, the BUSY bit will be cleared. The data can then be read from the CUR\_MON\_RD\_DAT register.

Any operations that write to the CUR\_MON\_WR\_DAT, CUR\_MON\_WR\_ADR, and CUR\_MON\_RD\_ADR registers will be ignored if the interface is busy.

Addresses written to the CUR\_MON\_WR\_ADR and CUR\_MON\_RD\_ADR registers are 8-bit. All data written to the CUR\_MON\_WR\_DAT register will be truncated at 16-bits. All data read from the CUR\_MON\_RD\_DAT register will be 16-bit data that is sign-extended to 32-bits.

.

EXT\_SPI\_OUT\_DAT0 0X10040040 (dec=268697664)

EXT\_SPI\_OUT\_DAT1 0X10040041 (dec=268697665)

EXT\_SPI\_OUT\_DAT2 0X10040042 (dec=268697666)

EXT\_SPI\_OUT\_DAT3 0X10040043 (dec=268697667)

EXT\_SPI\_OUT\_DAT4 0X10040044 (dec=268697668)

EXT\_SPI\_OUT\_DAT5 0X10040045 (dec=268697669)

EXT\_SPI\_OUT\_DAT6 0X10040046 (dec=268697670)

EXT\_SPI\_OUT\_DAT7 0X10040047 (dec=268697671)

[ERL\_BPM, BSM, BSM4]

These registers are not implemented.

[BPM4]

These registers hold data destined for 1 of 8 Digital-to-Analog converter channels that are located on a remote 6048-168 SPI DAC board. The signals connect through the front panel EXT I/O port and come out in an SPI format appropriate for the DAC.

The data is 16-bit 2’s-complement and drives the DAC through the range of -10 volts to +10 volts. The 16-bit data is packed into a 24-bit word that is formatted to access the correct DAC registers.

The EXT\_PORT\_CONFIG register needs to be programmed to connect SPI data to the front panel I/O connector.

In addition, these 8 registers have their address entered in the first 8 locations of the VECTOR ADDRESS TABLE. This allows the data to be read from the XBUS in a single operation, as opposed to an address/data pair of operations.

EXT\_SPI\_OUT\_DAT8 0X10040048 (dec=268697672)

[ERL\_BPM, BSM, BSM4]

This register is not implemented.

[BPM4]

This register holds data destined for the Analog Devices AD5362 DAC that is located on a remote 6048-168 SPI DAC board. The signals connect through the front panel EXT I/O port and come out in an SPI format appropriate for the DAC.

The data is 24-bit binary and is transmitted with no alterations. All 24 bits must be specified as described in the AD5362 datasheet.

The EXT\_PORT\_CONFIG register needs to be programmed to connect SPI data to the front panel I/O connector.

EXT\_PORT\_CONFIG 0x10040050 (dec=268697680)

[ERL\_BPM, BSM, BSM4]

This register is not implemented.

[BPM4]

This register enables various signals to be routed to the front panel EXT I/O port.

This register is divided into the following bit fields:

bit 0 = SOURCE\_MODULE: If this bit=0, then signals from the “EXT\_SPI\_OUT” registers are connected to the I/O port. If this bit=1, then signals from the “EXT\_TIM” registers are connected to the I/O port. The EXT\_TIM signals are assigned to the following pins:

TURN MARKER --- CLK

INTENSIFIER GATE --- CS

CAMERA GATE --- DOUT

The DIN pin is not currently used.

EXT\_TIM\_CSR 0x10040051 (dec=268697681)

[ERL\_BPM, BSM, BSM4]

This register is not implemented.

[BPM4]

Control and Status Register – This registers controls the trigger and operating modes for the timing outputs that are primarily intended for the CERN ODR experiment.

This register is divided into the following bit fields:

bit 0 = Trigger Enable - Set to 0 to return to timing state machines to their idle state and negate the

intensifier and camera outputs. Set to 1 to arm the trigger. If the trigger mode is 0, start at the next turn marker. If the trigger mode is 1, wait first for the CBPM trigger, then start at the next turn marker.

bit 1 = Trigger Mode - Set to 0 for immediate triggering on the next turn marker. Set to 1 to wait for the CESR BPM trigger. The CESR BPM trigger will be some combination of a 60 Hz line trigger, an injection trigger, and a system-wide software trigger.

bit 2 = Intensifier Pulse/DC Mode - Set to 1 to enable pulse mode operation. Set to 0 to enable continuous "trigger once" operation. Once triggered, the "Trigger Enable" bit must be set to 0 to negate the output and prepare for another assertion.

NOTE: If the intensifier is in pulse mode, the camera must also be in pulse mode. This is due to the fact that the intensifier is triggered relative to the triggering of the camera.

bit 3 = Camera Pulse/DC Mode - Set to 1 to enable pulse mode operation. Set to 0 to enable continuous "trigger once" operation. Once triggered, the "Trigger Enable" bit must be set to 0 to negate the output and prepare for another assertion.

bit 4 = Turn Mark Shift - Set to 0 to use the normal turn marker. Set to 1 to use a turn marker that is delayed by about ½ turn. Refer to the note below about using this bit to avoid the 3 clock dead zone in the FPGA logic when the sum of IPD + IPW is more than 58.

EXT\_TIM\_IPW 0x10040052 (dec=268697682)

EXT\_TIM\_IPD 0x10040053 (dec=268697683)

~~EXT\_TIM\_IPR 0x10040054 (dec=268697684)~~

EXT\_TIM\_ITD 0x10040054 (dec=268697684)

EXT\_TIM\_CPW 0x10040055 (dec=268697685)

EXT\_TIM\_CPD 0x10040056 (dec=268697686)

EXT\_TIM\_CPR 0x10040057 (dec=268697687)

[ERL\_BPM, BSM, BSM4]

These registers are not implemented.

[BPM4]

Intensifier Pulse Width (IPW) - Sets “1 less than” the number of 42 nsec ticks that the intensifier output is asserted. Valid data is 0 through 255. A value of 0 will give an output of 1 period.

Intensifier Pulse Delay (IPD) - Sets “1 less than” the number of 42 nsec ticks that the intensifier output is delayed relative to the arrival of the turn marker. Valid data is 0 through 255. A value of 0 will give a delay of 1 period.

NOTE: If the sum of IPD + IPW is more than 58, then the next turn will be missed and pulses will come every-other turn. Similarly at 119. This is due to the simple state machine, which needs 3 clock periods for overhead, missing the turn marker and having to wait another full turn. If the desired bunch falls in this dead region, use the ‘Turn Mark Shift’ bit in the EXT\_TIM\_CSR register to shft the arrival of the turn marker by ½ turn.

~~Intensifier Pulse Repetition Rate (IPR) - Sets the number of turns from pulse to pulse. Valid data is 4 through 2\*\*20, yielding a maximum period of 2.86 seconds. However, since the intensifier has a maximum trigger rate of 10 kHz, the practical minimum is 40.~~

Intensifier Turn Delay (ITD) - Sets “1 less than” the number of turns from when the Camera Pulse is asserted until the first Intensifier Pulse is asserted. Valid data is 1 through 15, yielding a maximum period of 38.4 useconds.

Camera Pulse Width (CPW) - Sets “1 less than” the number of turn periods that the camera output is asserted. It also sets the number of turns that the intensifier output is asserted. Valid data is 1 through 2\*\*20, yielding a maximum period of 2.86 seconds.

Camera Pulse Delay (CPD) - Sets “1 less than” the number of 42 nsec ticks that the camera output is delayed relative to the arrival of the turn marker. Valid data is 0 through 182. A value of 0 will give a delay of 1 period.

Camera Pulse Repetition Rate (CPR) - Sets the number of turns from pulse to pulse. Valid data is 1 through 2\*\*20, yielding a maximum period of 2.86 seconds.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Setting the ADC\_CLOCK\_CONTROL Register*

There are a few correct settings for the ADC\_CLOCK\_CONTROL register and the registers on the timing board; there are a great many incorrect settings. These guidelines will help you to do things correctly. They are divided into single ADC vs. dual ADC cases.

For all cases, set the 8 card delay registers on the timing board to a center value of 700. Then limit excursions to the range of 600 to 800. This will allow for +/- 1 nsec. adjustment for cable and card delay mismatches. If you go beyond this range then you will end up clocking data into a register or writing data to memory when the data is changing, resulting in invalid data.

Single ADC

Set the ADC\_CLK\_SEL bit (bit 6) to ‘0’ if you are controlling the conversion time with ‘Block A’ from the timing card. Set the ADC\_CLK\_SEL bit to ‘1’ if you are using ‘Block B’ from the timing card.

Set the ADC0\_REG\_CLK bits (bits 1..0) and/or ADC1\_REG\_CLK bits (bits 3..2) to binary ‘11’. This will cause the ADC data to be clocked into the register about 1.5 nsec before the ADC clock goes high. With the +/- 1 nsec range for the individual cards, the setup and hold times of the data register will always be met.

Set the MEM\_WR\_CLK bits (bits 5..4) to binary ‘00’.

The final result is to write the binary pattern ‘0001111’ for ‘Block A’ timing or ‘1001111’ for ‘Block B’ timing.

Dual ADC

To be written

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Interrupts*

Various signals in the Xilinx chip are connected to interrupt lines on the DSP. The DSP can be configured to actually generate interrupts when these signals are asserted, or it can simply examine the state of the signal. If interrupts are being generated, the lowest priority interrupt is IRQ0 and the highest is IRQ3. The state of the interrupt signal can be observed in the DSP’s ILAT register. If the interrupt is edge-sensitive, the signal observed in the ILAT register can be cleared by writing a ‘1’ to the corresponding bit in the ILATCL register. Level sensitive interrupts should be cleared by removing the interrupting condition. Edge vs. level sensitivity is determined by the DSP’s SQCTL register.

Enabling of interrupt generation is controlled by the DSP’s IMASK and PMASK registers. Bit 60 of the IMASK register needs to be set as well to enable any hardware interrupts.

IRQ0

Not connected. Associated with bit 41 of the DSP’s ILAT/IMASK/PMASK registers.

IRQ1

This interrupt is connected to the ACQ\_DONE bit of the DATA\_ACQ register. It will be asserted when data collection is finished. This interrupt is associated with bit 42 of the DSP’s ILAT/IMASK/PMASK registers.

IRQ2

This interrupt is connected to the ‘turn\_mark’ signal in the Xilinx chip. It will be asserted for 42 nsec when the turn marker arrives. A new value for GLOBAL\_TURN\_DAT is then available. This interrupt is associated with bit 43 of the DSP’s ILAT/IMASK/PMASK registers.

IRQ3

Not connected. Associated with bit 44 of the DSP’s ILAT/IMASK/PMASK registers.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Auxiliary I/O Card LEDs and Connectors*

There are nine LEDs on the front of the auxiliary I/O card. Starting near the power connector and at the position closest to the circuit board, the LEDs indicate:

+4 Volt DC Power OK

+5 Volt DC Power OK

+12 Volt DC Power OK

Turn Marker Detected

-5 Volt DC Power OK

+2 Volt DC Power OK

DSP Activity Detected

Xbus Activity Detected

ColdFire (Ethernet) Activity Detected

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Local Bus Masters*

There are several devices which can be “bus master” on the local address and data bus. Arbitration logic within the XILINX chip, along with the /HBR and /HBG signals on the DSP, is used to determine which device has bus ownership at any instant. The potential bus masters are:

DSP

XBUS Controller

Ethernet ColdFire controller

FLASH Programmer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *ColdFire Programming*

 Base address = 0x80000000

The ColdFire 'dimm' is a 1-byte oriented machine, while the BPM/BSM/FLM modules are 4-byte (32-bit word) oriented machines. The two LSBs of the 'dimm\_adr' bus will specify the byte within a 4-byte word. The coldfire is "big-endian", so when a 32-bit word is moved over the 8-bit bus, the MSB goes with byte address '00' and the LSB goes with byte address '11'. The address map seen by the ColdFire is:

Adr BinAdr Register

00 0000xx fix\_adr32

04 0001xx fix\_dat32

08 0010xx inc\_adr32

0C 0011xx inc\_dat32

10 0100xx fix\_adr16

14 0101xx fix\_dat16

18 0110xx inc\_adr16

1C 0111xx inc\_dat16

20-3C reserved

The programmer needs to write the address of the BPM/BSM/FLM location that he wants to access to one of the 4 "adr" registers. He then reads from or writes to the associated "dat" register to transfer data. The address to use is that of the 4-byte entity (use the same address that would be used by the XBUS or the DSP)

If the programmer is using a register with the "fix\_dat" name, the BPM/BSM/FLM address will remain at the value that he last programmed. If the programmer is using a register with the "inc\_dat" name, the BPM/BSM/FLM address will be incremented by 1 at the end of the operation. So to read from a series of consecutive BPM/BSM/FLM locations, the programmer would write the initial address to the "inc\_adr" register, then repeatedly access the "inc\_dat" register to get the data. To read over and over again from the same address, like for testing a status bit, the programmer would write the desired address to the "fix\_adr" register, then repeatedly access the "fix\_dat" register to get the data.

The registers that end with "32" are used to transfer 4-byte data to and from BPM/BSM/FLM locations. Registers that end with "16" are used to transfer the 2 lower bytes of a BPM/BSM/FLM location. Remember that ALL data in the BPM/BSM/FLM is 32-bit data, but if you are just reading raw ADC values, there is no need to waste bandwidth moving a bunch of zeros around.

The DIMM module needs to be configured for TCP/IP applications. The TCP/IP parameters are initialized with environment variables and then used by the 'rtems\_nvram' function. The board must be configured by using appropriate values in the following set of commands:

setenv HOSTNAME klybpm08

setenv IPADDR0 192.168.7.118

setenv GATEWAY 192.168.7.1 ! use 172.17.0.1 for ERL subnet

setenv NETMASK 255.255.255.0 ! use 255.255.0.0 for ERL subnet

setenv SERVER 192.168.1.80 ! BOOTP server?

setenv NAMESERVER 128.84.47.200 ! or 128.84.46.26

setenv NTPSERVER 128.84.46.17 ! or 128.84.46.26 or 128.84.46.181

setenv NFSMOUNT lnx180c:/mnt/instr:/mnt/instr ! Server:Path:MountPoint

And if you are supporting EPICS:

 setenv CMDLINE /mnt/instr/epics/example/iocBoot/ioctest/st.cmd

 setenv BOOTFILE /mnt/instr/epics/example/bin/RTEMS-uC5282/test.boot

To compile and download a program to the ColdFire, see the Twiki entry at:

https://wiki.lepp.cornell.edu/lepp/bin/view/CESR/Bunch/RTEMS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Accumulator Programming*

ANALOG CARD 0

ADR 0x0A000000 (dec=167772160)

CSR 0x0A000001 (dec=167772161)

LO\_DAT 0x0A000002 (dec=167772162)

HI\_DAT 0x0A000003 (dec=167772163)

ANALOG CARD 1

 ADR 0x0A400000 (dec=171966464)

 CSR 0x0A400001 (dec=171966465)

 LO\_DAT 0x0A400002 (dec=171966466)

 HI\_DAT 0x0A400003 (dec=171966467)

ANALOG CARD 2

ADR 0x0A800000 (dec=176160768)

CSR 0x0A800001 (dec=176160769)

LO\_DAT 0x0A800002 (dec=176160770)

HI\_DAT 0x0A800003 (dec=176160771)

ANALOG CARD 3

ADR 0x0AC00000 (dec=180355072)

CSR 0x0AC00001 (dec=180355073)

LO\_DAT 0x0AC00002 (dec=180355074)

HI\_DAT 0x0AC00003 (dec=180355075)

## Each accumulator card has 4 registers. The register names and the offset from the base address for each board are;

0 = ADR

1 = CSR

2 = LO\_DAT

3 = HI\_DAT

The accumulator boards use indirect addressing to access data within the accumulator chip. The address must be written to the ADR register, after which data can be written to the LO\_DAT register or read from the LO\_DAT and HI\_DAT registers. All data transfers are 16-bit. All write operations are only 16 bits. Some read operations are 16 bit while others are 32 bits. For the 32 bit cases, the high 16 bits will be latched when the low 16 bits are read, and can be accessed from the HI\_DAT register.

NOTE: AUTO-INCREMENT HAS NOT BEEN IMPLEMENTED YET. When using the auto-incrementing mode, one first writes the initial address to the ADR register, then repeatedly reads or writes the LO\_DAT register. The address \*always\* gets incremented after access to the low data register. If 32-bit data is being read, the HI\_DAT register should be read after the LO\_DAT register.

Up to 4 accumulator boards can be used in a system. Each board has a 2 pin ID header that is used to set the board address. When writing to the ADR register, the write operation happens on **all** accumulator boards. All other operations (read ADR, read/write CSR, LO\_DAT, HI\_DAT) involve only one board. The two MSBs written to the ADR register must match the ID header to select a particular board. The remaining 14 bits in the ADR register are use to access some resource on the selected board. This results in a 16k address space per board.

The accumulator chip can be read from or written to while data is being collected in the ADC memory. Readout of the ADC memory or the accumulator can be interspersed.

CSR offset = 1

This register controls or monitors overall operation of the accumulator chip. It is a 16-bit register that can be written or read.

bit 0 = Nreset: This bit will power up low (asserted). It must have a '1' written to it to take the chip out of the ‘reset’ mode.

bit 1 = Nsetup: This bit will power up low (asserted). It must have a '1' written to it to take the chip out of the ‘setup’ mode.

The intended use of these bits is that they will initially both be zero. The programmer will set the ‘Nreset’ bit to one and then proceed to configure the chip by programming the lookup table and setting the various registers. When the programmer is ready to start collecting data, the ‘Nsetup’ bit will also be set to one. This causes all of the various state machines and counters to start together.

ACCUMULATOR ADDRESS MAP

Block 0: 1 sector of 4096 addresses

Decode ADR[13:12] Pass ADR[11:0]

Start End Size Function

0x0000 {dec=0} 0x0fff {dec=4095) 4096 Hit Lookup Table (Read/Write)

Block 0 contains the 4k-by-2 bit Hit Lookup table. It is implemented as a dual-port memory, where one port provides read/write access for initializing and verifying the data and the other port is used to determine how many photons have hit the detector based on a combination of high and low threshold signal comparisons. When writing to it, data bits [1..0] contain the data to be stored. Data bits [15..2] are discarded. When reading, data bits [1..0] will contain the stored value. Data bits [15..2] will always read as zeroes.

The address used for looking up the hit count is derived from 6 ‘lo\_hit’ channels (address bits [11..6]) and 6 ‘hi\_hit’ channels (address bits [5..0]). If one wishes to only use the first 6 'hi\_hit' channels, then the lookup table only needs to have the first 64 locations programmed. The ‘Hit Mask Register’ would be programmed as 0x007d. Refer to the discussion about the Hit Mask Register later in this document to see the details about the mapping between ADC channels and lookup table address bits.

Block 1: 16 sectors of 256 addresses

Decode ADR[13:8] Pass ADR[7:0]

Start End Size Function

0x1000 {dec=4096} 0x1fff {dec=8191} 4096 16-bit R/W Control Registers

0x1000 {dec=4096} ADC 0 Pedestal

0x1001 {dec=4097} ADC 0 Low Threshold

0x1002 {dec=4098} ADC 0 High Threshold

0x1003 {dec=4099} Unused

0x1004 {dec=4100} ADC 1 Pedestal

0x1005 {dec=4101} ADC 1 Low Threshold

0x1006 {dec=4102} ADC 1 High Threshold

0x1007 {dec=4103} Unused

0x1008 {dec=4104} ADC 2 Pedestal

0x1009 {dec=4105} ADC 2 Low Threshold

0x100A {dec=4106} ADC 2 High Threshold

0x100B {dec=4107} Unused

0x100C {dec=4108} ADC 3 Pedestal

0x100D {dec=4109} ADC 3 Low Threshold

0x100E {dec=4110} ADC 3 High Threshold

0x100F {dec=4111} Unused

0x1010 {dec=4112} ADC 4 Pedestal

0x1011 {dec=4113} ADC 4 Low Threshold

0x1012 {dec=4114} ADC 4 High Threshold

0x1013 {dec=4115} Unused

0x1014 {dec=4116} ADC 5 Pedestal

0x1015 {dec=4117} ADC 5 Low Threshold

0x1016 {dec=4118} ADC 5 High Threshold

0x1017 {dec=4119} Unused

0x1018 {dec=4120} ADC 6 Pedestal

0x1019 {dec=4121} ADC 6 Low Threshold

0x101A {dec=4122} ADC 6 High Threshold

0x101B {dec=4123} Unused

0x101C {dec=4124} ADC 7 Pedestal

0x101D {dec=4125} ADC 7 Low Threshold

0x101E {dec=4126} ADC 7 High Threshold

0x101F {dec=4127} Unused

0x1100 {dec=4352} Bunch Pattern Register, (bunch\_pat[15:0])

0x1101 {dec=4353} Bunch Pattern Register, (bunch\_pat[31:16])

0x1102 {dec=4354} Bunch Pattern Register, (bunch\_pat[47:32])

0x1103 {dec=4355} Bunch Pattern Register, (bunch\_pat[63:48])

0x1104 {dec=4356} Bunch Pattern Register, (bunch\_pat[79:64])

0x1105 {dec=4357} Bunch Pattern Register, (bunch\_pat[95:80])

0x1106 {dec=4358} Bunch Pattern Register, (bunch\_pat[111:96])

0x1107 {dec=4359} Bunch Pattern Register, (bunch\_pat[127:112])

0x1108 {dec=4360} Bunch Pattern Register, (bunch\_pat[143:128])

0x1109 {dec=4361} Bunch Pattern Register, (bunch\_pat[159:144])

0x110A {dec=4362} Bunch Pattern Register, (bunch\_pat[175:160])

0x110B {dec=4363} Bunch Pattern Register, (bunch\_pat[191:176])

0x1200 {dec=4608} Hit Mask Register

0x1201 {dec=4609} Force Hi Hit Register

0x1300 {dec=4864} Bunch Rate Low Period Register

0x1301 {dec=4865} Bunch Rate High Period Register

0x1400 {dec=5120} Fast Global Rate Low Period Register

0x1401 {dec=5121} Fast Global Rate High Period Register

0x1500 {dec=5376} Slow Global Rate Low Period Register

0x1501 {dec=5377} Slow Global Rate High Period Register

0x1600 {dec=5632} Geo Global Rate Low Period Register

0x1601 {dec=5633} Geo Global Rate High Period Register

0x1700 {dec=5888} Geo Bunch Rate Low Period Register

0x1701 {dec=5889} Geo Bunch Rate High Period Register

Block 1 contains 16-bit control registers. Each register can be read or written.

NOTE: The data from the ADCs is first inverted so that all numbers are positive, a small amplitude is near zero, and a large amplitude is near 65000. The 12-bit ADC data is left-shifted to occupy bits [15:4]. Pedestals and thresholds are treated similarly.

ADC Pedestal registers contain a number that is subtracted from the ADC data. It is a 16-bit unsigned number, but only the 12 MSBs are used. If the result would be less than zero, it is clipped at zero. All bits are cleared to ‘0’ when the ‘Nreset’ bit in the CSR is zero.

ADC Low Threshold and ADC High Threshold registers contain numbers that the pedestal corrected ADC data is compared with. It is a 16-bit unsigned number, but only the 12 MSBs are used. The comparison is “greater than or equal to”. All bits are cleared to ‘0’ when the ‘Nreset’ bit in the CSR is zero.

Bunch Pattern Registers are used to enable or disable processing of every 7th RF bucket (14 nsec.). It is a 16-bit number, but the last register only uses the 7 LSBs. There are 183 possible bunches specified in bunch\_pat[182:0]. A ‘1’ in a bit position enables processing of that bunch while a ‘0’ disables it. All bits are preset to ‘1’ when the ‘Nreset’ bit in the CSR is zero.

The Hit Mask Register is used to specify which ADCs are producing valid data to be used in forming the address for the Hit Lookup Table and for counting geometric channel rates. Only ADC channels 1 thru 6 are used to generate the address. The other 2 channels (0 and 7) have signals that are not part of the lookup process. They are still part of the 'hit\_mask'. Bits [7..0] enable use of the High Threshold comparison while bits [15..8] enable use of the Low Threshold comparison. All bits are preset to ‘1’ when the ‘Nreset’ bit in the CSR is zero. The logic for creating the lookup table address is:

 always @(posedge clk) begin

 if (bunch\_en) begin

 hit\_lkup\_adr[0] <= hi\_hit[3] & hit\_mask[3];

 hit\_lkup\_adr[1] <= hi\_hit[6] & hit\_mask[6];

 hit\_lkup\_adr[2] <= hi\_hit[2] & hit\_mask[2];

 hit\_lkup\_adr[3] <= hi\_hit[5] & hit\_mask[5];

 hit\_lkup\_adr[4] <= hi\_hit[1] & hit\_mask[1];

 hit\_lkup\_adr[5] <= hi\_hit[4] & hit\_mask[4];

 hit\_lkup\_adr[6] <= lo\_hit[3] & hit\_mask[11];

 hit\_lkup\_adr[7] <= lo\_hit[6] & hit\_mask[14];

 hit\_lkup\_adr[8] <= lo\_hit[2] & hit\_mask[10];

 hit\_lkup\_adr[9] <= lo\_hit[5] & hit\_mask[13];

 hit\_lkup\_adr[10] <= lo\_hit[1] & hit\_mask[9];

 hit\_lkup\_adr[11] <= lo\_hit[4] & hit\_mask[12];

 end

 else

 hit\_lkup\_adr <= 12'h000;

 end

The Force Hi Hit Register allows one to always force the result of the High Threshold comparison to be true ONLY for Geo Global Rate data and Geo Channel Rate data. It is useful for testing. Bits [7..0] force the corresponding channel. Bits [15..8] are unused. All bits are cleared to ‘0’ when the ‘Nreset’ bit in the CSR is zero.

The various Rate Period Registers control the duration of the sample period for a rate measurement. The period is specified in units of ‘cesr turn time’ (2.56 usec). The underlying counters are 22-bit counters, so the maximum setting is 2\*\*22 x 2.56 usec = 10.7 seconds. The 16 LSBs are specified in the various Low Period Registers, while the 6 MSBs are specified in the various High Period Registers. The number of turns written to the rate registers must be 1 less than the number of turns in the desired period. To get a period of 1 second (390320 turns) program a value of ‘5’ in the High Period Register and a value of ‘62639’ in the Low Period Register. [390320 – 1 = 390319 = 0x5f4af, therefore the High Period Register = 0x05 = 5 and the Low Period Register = 0xf4af = 62639]

Block 2: 16 sectors of 256 addresses

Decode ADR[13:8] Pass ADR[7:0]

Start End Size Function

0x2000 {dec=8192} 0x2fff {dec=12287} 4096 32-bit Rate Registers (Read Only)

0x2000 {dec=8192} 183 Bunch Rate Data

0x2100 {dec=8448} 1 Fast Global Rate Data

0x2200 {dec=8704} 1 Slow Global Rate Data

0x2300 {dec=8960} 8 Geo Global Rate Data (8 chan, all bunches)

0x2800 {dec=10240} 183 Geo Channel 0 Rate Data

0x2900 {dec=10496} 183 Geo Channel 1 Rate Data

0x2A00 {dec=10752} 183 Geo Channel 2 Rate Data

0x2B00 {dec=11008} 183 Geo Channel 3 Rate Data

0x2C00 {dec=11264} 183 Geo Channel 4 Rate Data

0x2D00 {dec=11520} 183 Geo Channel 5 Rate Data

0x2E00 {dec=11776} 183 Geo Channel 6 Rate Data

0x2F00 {dec=12032} 183 Geo Channel 7 Rate Data

Block 2 contains 32-bit Rate Data registers. Each register is read-only. To read any given Rate Data register, setup the ADR register with the address of the desired Rate Data register. Read from the LO\_DAT register to get the 16 LSBs. Then read from the HI\_DAT register to get the 16 MSBs. The resulting 32-bit number contains the count in the lower 20 bits and a sample tag in the upper 12 bits. Analysis code will need to separate these two numbers.

The 12-bit sample tag is incremented at the end of each sample period. The value will wrap around to zero when it exceeds 4095. Each of the 4 Rate Period Registers drives its own sample tag, so they do not necessarily all count in lockstep order. The sample tags will be reset to zero when either ‘Nreset’ or ‘Nsetup’ in the CSR is a zero.

Block 3: 1 sector of 4096 addresses

Start End Size Function

0x3000 {12288} 0x3fff {16383} 4096 Reserved

Nothing is implemented in block 3.

#### Sample command files to access the accumulator board

[cesr.cesrbpm.6048\_113]accum\_csr\_wr.com

$! Write to the accumulator CSR and then read back (ADC card 0 only)

$! do accum\_csr\_wr accum\_csr\_dat

$! accum\_csr\_dat in decimal

$! crs 12/22/05

$!

$ wo = "write sys$output"

$! set up the address register

$ fff vxputn cbpm adr tst 1 1 167772160

$! write to the csr register

$ fff vxputn cbpm adr tst 1 1 167772161

$ fff vxputn cbpm dat tst 1 1 'p1

$! read the CSR register

$ fff vxgetn cbpm dat tst 1 1

 [cesr.cesrbpm.6048\_113]accum\_wr.com

$! Write a 16-bit word to an accumulator location (ADC card 0 only)

$! do accum\_wr accum\_adr accum\_dat

$! accum\_adr and accum\_dat in decimal

$! crs 12/22/05

$!

$ wo = "write sys$output"

$! set up the address register

$ fff vxputn cbpm adr tst 1 1 167772160

$ fff vxputn cbpm dat tst 1 1 'p1

$! write to the lo\_dat register

$ fff vxputn cbpm adr tst 1 1 167772162

$ fff vxputn cbpm dat tst 1 1 'p2

[cesr.cesrbpm.6048\_113]accum\_rd.com

$! Read a 16-bit word from an accumulator location (ADC card 0 only)

$! do accum\_rd accum\_adr

$! accum\_adr in decimal

$! crs 12/22/05

$!

$ wo = "write sys$output"

$! set up the address register

$ fff vxputn cbpm adr tst 1 1 167772160

$ fff vxputn cbpm dat tst 1 1 'p1

$! read from the lo\_dat register

$ fff vxputn cbpm adr tst 1 1 167772162

$ fff vxgetn cbpm dat tst 1 1

[cesr.cesrbpm.6048\_113]accum\_rd2.com

$! Read two 16-bit words from an accumulator address (ADC card 0 only)

$! do accum\_rd2 accum\_adr

$! accum\_adr in decimal

$! crs 12/22/05

$!

$ wo = "write sys$output"

$! set up the address register

$ fff vxputn cbpm adr tst 1 1 167772160

$ fff vxputn cbpm dat tst 1 1 'p1

$loop:

$! read from the lo\_dat register

$ fff vxputn cbpm adr tst 1 1 167772162

$ fff vxgetn cbpm dat tst 1 1

$! read from the hi\_dat register

$ fff vxputn cbpm adr tst 1 1 167772163

$ fff vxgetn cbpm dat tst 1 1

$ goto loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *xBSM4 Current Monitor Programming*

##  *(see next section for the older vBSM current monitor)*

*Add section here about the signal levels, similar to vBSM info*

Current Monitor Address Map (\*=disabled)

Current RAM 0x00-0x1f (dec=0-31)

Pedestal RAM 0x20-0x3f (dec=32-63)

\*Limit RAM 0x40-0x5f (dec=64-95)

CSR (Control and Status Register) 0x60 (dec=96)

DAC0 Setting 0x61 (dec=97)

DAC1 Setting 0x62 (dec=98)

DAC1 Setting 0x63 (dec=99)

AIN0 reading 0x64 (dec=100)

AIN1 reading 0x65 (dec=101)

Total Current 0x66 (dec=102)

Supervisor State Machine 0x67 (dec=103)

Current RAM 0x00-0x1f (dec=0-31)

This is a 16-bit by 32 dual-port memory. If the PROGRAM/RUN bit of the CSR is at the PROGRAM value, readout of the ADCs is stopped and no current updates are available. If the PROGRAM/RUN bit of the CSR is at the RUN value, then new ADC data is fetched from all active boards. If the RAW/PED bit of the CSR is at the RAW value, it contains the raw ADC value of the current for each of the 32 channels. If the RAW/PED bit of the CSR is at the PED value, it contains the corrected (pedestal subtracted) value. The data is 16-bit 2’s-complement format. Data is written to this memory by the hardware and is read-only from the DSP or from external access.

Pedestal RAM 0x20-0x3f (dec=32-63)

This is a 16-bit by 32 dual-port memory. It contains the pedestal (or zero current) value to be subtracted from the measured current for each of the 32 channels. The data is 16-bit 2’s-complement format. Data is written to this memory by the DSP or external program. If the PROGRAM/RUN bit of the CSR is in the PROGRAM state, then the DSP can read back the memory contents. If the bit is in the RUN state, then the hardware has read access and the DSP cannot read the data.

\*Limit RAM 0x40-0x5f (dec=64-95)

This is currently unused.

CSR (Control and Status Register) 0x60 (dec=96)

This register is divided into the following bit fields:

bit 0 = PROGRAM/RUN:

This read/write bit controls the mode of operation. If the bit is a ‘0’, the system is in the PROGRAM mode. If the bit is a ‘1’, the system is in the RUN mode. In PROGRAM mode, the readout of the channel ADCs is disabled, and the state machines that control the readout are held in a “reset” state. This can be useful to deal with a hung current monitor board, or to disable the digital activity if it induces noise.

bit 1 = RAW/PED:

This read/write bit controls pedestal subtraction. If the bit is a ‘0’, the system is in the RAW mode and the contents of the “Current RAM” is data directly from the ADCs. If the bit is a ‘1’, the system is in the PEDestal mode and the contents of the “Current RAM” have had the contents of the “Pedestal RAM” subtracted.

bit 5:2 = CARD ENABLE[3..0]:

These read/write bits are used to disable/enable readout of individual carrier boards. If a bit is a zero, then data from the corresponding carrier board will not be considered. If a bit is a one, then the corresponding carrier board is assumed to be active.

Normally, all four bits will be “one” and all four carrier boards will be active. However, when testing carrier boards, there may be fewer installed. The boards that are not present must be disabled, or else the readout will hang.

bit 6 = MONITOR\_MODE:

This read/write bit controls the source of data for the front panel monitor BNC connector. If the bit is ’0’ then the total current value is output. If the bit is ‘1’, then the value from a single channel is output. The channel is selected by the MONITOR\_CHANNEL bits of the CSR.

bits 11:7 = MONITOR\_CHANNEL[4..0]:

These read/write bits select which channel is used as the source for the front panel monitor BNC when the MONITOR\_MODE bit is in the CHANNEL mode. The physical mapping needs to be taken into account in specifying a particular detector element. Bits [11..10] select 1 of 4 cards, and bits [9..7] select 1 of 8 ADCs on a card.

Here are some typical settings for the CSR:

0x003d = 0b0000000000111101 = monitor total current, use cards 0,1,2,3, raw mode, run mode

0x03ff = 0b0000001111111111 = monitor card 0 chan 7, use cards 0,1,2,3, ped mode, run mode

DAC0 Setting 0x61 (dec=97)

DAC1 Setting 0x62 (dec=98)

DAC2 Readback 0x63 (dec=99)

These are 14-bit unsigned registers, using bits [13..0]. They drive a Linear Technologies LTC1658 DAC, which provides a 0 to 4.5 volt output. DAC0 is connected to pin 7 on the front-panel DB25 connector. DAC1 is connected to pin 20 on the front-panel DB25 connector. DAC2 is connected to the front panel BNC connector.

DAC0 and DAC1 can be written to. DAC2 is driven by the current reading circuit. It is controlled by the MONITOR\_MODE and MONITOR\_CHANNEL bits of the CSR.

When read from, the last value that was written to the DACs is returned.

The DAC2 setting is subject to rollover and incorrect data in the event of very large current values. If the average channel has a current reading of more than +/- 1000 units, the data in this register would be suspect.

AIN0 Readback 0x64 (dec=100)

AIN1 Readback 0x65 (dec=101)

These are 16-bit registers. They contain the value measured across pins 5 and 18 for AIN0, or 9 and 22 for AIN1. There is no pedestal correction. The data is 16-bit 2’s-complement format. Data can be read at any time. The value is updated approximately 6.8 times per second.

Total Current 0x66 (dec=102)

This read-only register contains the total current of all 32 channels. If the RAW/PED bit of the CSR is at the RAW value, it contains the sum of the raw ADC value of the current for all of the 32 channels. If the RAW/PED bit of the CSR is at the PED value, it contains the sum of the corrected (pedestal subtracted) values. The data is 16-bit 2’s-complement format.

This register is subject to rollover and incorrect data in the event of very large current values. If the average channel has a current reading of more than +/- 1000 units, the data in this register would be suspect.

Supervisor State Machine 0x67 (dec=103)

This read-only register provides access to the supervisor state machine. It shows the level of each state bit. The correlation between bits in this register and the state of the supervisor is:

 supv\_state[0] = (mon\_sm == init);

 supv\_state[1] = (mon\_sm == read1);

 supv\_state[2] = (mon\_sm == read2);

 supv\_state[3] = (mon\_sm == wait1);

 supv\_state[4] = (mon\_sm == wait2);

 supv\_state[5] = (mon\_sm == check1);

 supv\_state[6] = (mon\_sm == check2);

 supv\_state[7] = (mon\_sm == check3);

 supv\_state[8] = (mon\_sm == check4);

 supv\_state[9] = (mon\_sm == check5);

 supv\_state[10] = run\_mode;

 supv\_state[11] = kill\_hv\_ff;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *vBSM Current Monitor Programming*

##  *(see previous section for the newer xBSM4 current monitor)*

The amplifiers on the BSM ADC boards have an input bias current between 20 and 40 uA. This corresponds to a signal between 1 and 2 mV at the input of the Current Monitor Board ADC which has a range of +/- 50mV. This 1 to 2 mV is the DC offset (temperature drift ~ 0.1% per degree). The signal we want to measure is 1 to 10 uA, or 0.05 to 0.5 mV at the input of the Current Monitor Board ADC (a fraction of the pedestal value). There is no specific hardware limit on the current signal, however, only a broken channel should saturate the ADC. It is unlikely that over/under range testing is required.

Full scale: +/- 50 mV = +/- 1 mA = 16 bits (1 LSB = 1.526 uV = 30.52 nA)

Pedestal: +/- 1 to 2 mV = +/- 20 to 40 uA (ADC dat = 655 to 1311)

Signal: 0.05 to 0.5 mV = 1 to 10 uA (ADC dat = 32 to 328)

Current Monitor Address Map

Current RAM 0x00-0x1f (dec=0-31)

Pedestal RAM 0x20-0x3f (dec=32-63)

Limit RAM 0x40-0x5f (dec=64-95)

Total Current Limit Setting 0x60 (dec=96)

DAC0 Setting 0x61 (dec=97)

DAC1 Setting 0x62 (dec=98)

HV Readback 0x63 (dec=99)

Shutter Time Setting/Readback 0x64 (dec=100)

CSR (Control and Status Register) 0x65 (dec=101)

Shutter OverCurrent Cycles 0x66 (dec=102)

High Voltage OverCurrent Cycles 0x67 (dec=103)

Total Current 0x68 (dec=104)

Supervisor State Machine 0x69 (dec=105)

Current RAM 0x00-0x1f (dec=0-31)

This is a 16-bit by 32 dual-port memory. If the PROGRAM/RUN bit of the CSR is at the PROGRAM value, it contains the raw ADC value of the current for each of the 32 channels. If the PROGRAM/RUN bit of the CSR is at the RUN value, it contains the corrected (pedestal subtracted) value. The data is 16-bit 2’s-complement format. Data is written to this memory by the hardware and is read-only from the DSP. If a High Voltage trip occurs, the data in this memory will be frozen at the values that caused the trip.

Pedestal RAM 0x20-0x3f (dec=32-63)

This is a 16-bit by 32 dual-port memory. It contains the pedestal (or zero current) value to be subtracted from the measured current for each of the 32 channels. The data is 16-bit 2’s-complement format. Data is written to this memory by the DSP. If the PROGRAM/RUN bit of the CSR is in the PROGRAM state, then the DSP can read back the memory contents. If the bit is in the RUN state, then the hardware has read access and the DSP cannot read the data.

Limit RAM 0x40-0x5f (dec=64-95)

This is a 16-bit by 32 dual-port memory. It contains the limit value to be compared with the measured and corrected current for each of the 32 channels. The data is 16-bit 2’s-complement format. Data is written to this memory by the DSP. If the PROGRAM/RUN bit of the CSR is in the PROGRAM state, then the DSP can read back the memory contents. If the bit is in the RUN state, then the hardware has read access and the DSP cannot read the data.

Total Current Limit Setting 0x60 (dec=96)

This is a 16-bit register. It contains the limit value to be compared with the total measured and corrected current for all of the 32 channels. The data is 16-bit 2’s-complement format. Data can be written or read at any time by the DSP.

DAC0 Setting 0x61 (dec=97)

DAC1 Setting 0x62 (dec=98)

These are 14-bit unsigned registers, using bits [13..0]. They drive a Linear Technologies LTC1658 DAC, which provides a 0 to 4.5 volt output. DAC0 is used to set the amplitude of the high voltage supply. DAC1 is currently unused. When read from, the last value that was written to the DACs is returned.

If a High Voltage trip occurs, the DAC0 register will be set to zero and that value will repeatedly be written to the DAC. Readback of the DAC0 Setting register will give a value of zero. The PROGRAM/RUN bit of the CSR needs to be set back to the PROGRAM state before new data that is written to the DAC0 register will take effect.

HV Readback 0x63 (dec=99)

This is a 16-bit register. It contains the value measured by the HighVoltage ADC. There is no pedestal correction. The data is 16-bit 2’s-complement format. Data can be read at any time by the DSP. The value in the HV register is updated approximately 6.8 times per second.

Shutter Time Register 0x64 (dec=100)

This register provides a timer for BSM shutter opening. When written to, the input is a 16-bit unsigned number that specifies the number of 10-turn periods that the shutter is open. A value of 1000 will cause the shutter to be open for 10000 turns (actually for 10,132 with roundup error). The 10-turn counter is running asynchronously to the loading of data, so it is possible that the counter will decrement almost immediately after the data is loaded. This would give a runt period of less than 10 turns. An input of "8" would give between 70 and 80 turns. The maximum time that the shutter can be open is 655k turns. When read from, the remaining time in the counter will be returned.

The shutter can be closed immediately by writing a value of zero. The shutter will also be closed when the overcurrent detector finds an overcurrent condition.

Since this register uses an unsigned 16-bit number, a negative number will be read back if the current value is between 32,768 and 65,535. This is due to the sign bit extension that the hardware supplies. Values of 32,767 or less will be read correctly.

CSR (Control and Status Register) 0x65 (dec=101)

This register is divided into the following bit fields:

bit 0 = PROGRAM/RUN:

This read/write bit controls the mode of operation. If the bit is a ‘0’, the system is in the PROGRAM mode. If the bit is a ‘1’, the system is in the RUN mode.

bit 5:1 = RESERVED:

These read/write bits are reserved for future use. Whatever is written to them will be read back.

bit 6 = SHUTTER\_OPEN:

This read-only bit shows the state of the control line that opens the shutter. If the bit is ’0’ then the shutter is supposed to be closed. If the bit is ‘1’, then the shutter is supposed to be open.

bit 7 = SHUTTER\_OVERCURRENT:

This read-only bit indicates if the overcurrent protection circuit closed the shutter. If the bit is ’0’ then no overcurrent situation has been detected. If the bit is ‘1’, then an overcurrent situation has been detected and the shutter has been closed.

Once an overcurrent situation has occurred, the PROGRAM/RUN bit must be set back to the PROGRAM state to clear the error, and then to the RUN state to enable the protection.

bit 8 = SHUTTER\_OVERCURRENT\_CAUSE:

This read-only bit what type of overcurrent situation closed the shutter. If the bit is ’0’, then the shutter was closed due to an overcurrent situation on one of the channels. If the bit is ‘1’, then the shutter was closed due to an overcurrent situation on the total sum of all the channels.

bit 9 = HIGH\_VOLTAGE\_OVERCURRENT:

This read-only bit indicates if the overcurrent protection circuit turned off the High Voltage by setting the DAC to zero. If the bit is ’0’ then no overcurrent situation has been detected. If the bit is ‘1’, then an overcurrent situation has been detected and the High Voltage has been turned off.

Once an overcurrent situation has occurred, the PROGRAM/RUN bit must be set back to the PROGRAM state to clear the error and enable writing a new value to the DAC0 register. The PROGRAM/RUN bit must then be set to the RUN state to enable the protection.

bit 10 = HIGH\_VOLTAGE\_OVERCURRENT\_CAUSE:

This read-only bit what type of overcurrent situation killed the High Voltage. If the bit is ’0’, then the voltage was turned off due to an overcurrent situation on one of the channels. If the bit is ‘1’, then the voltage was turned off due to an overcurrent situation on the total sum of all the channels.

bits 15:11 = OVERCURRENT\_CHANNEL\_NUM:

These 5 read-only bits contain the channel number of the ADC that caused a SHUTTER\_OVERCURRENT trip. The data is only valid if bit 7 is a ‘1’. If bit 8 is a ‘0’, then this is the first channel that had too much current. If bit 8 is a ‘1’, then this is the channel that caused the accumulated total current to exceed the value of the Total Current Limit Setting.

Shutter OverCurrent Cycles 0x66 (dec=102)

This read/write register is used to specify how many consecutive cycles of an overcurrent situation must occur before a SHUTTER\_OVERCURRENT trip occurs and causes the shutter to close. Each cycle takes about 140 milliseconds. The maximum value is 255 cycles.

High Voltage OverCurrent Cycles 0x67 (dec=103)

This read/write register is used to specify how many consecutive cycles of an overcurrent situation must occur before a HIGH\_VOLTAGE\_OVERCURRENT trip occurs and causes the High Voltage to be turned off. Each cycle takes about 140 milliseconds. The maximum value is 255 cycles.

Generally one would program the High Voltage OverCurrent Cycles register with a value that is larger than the value programmed in the Shutter OverCurrent Cycles register. That will cause the shutter to close first, and the high voltage would be turned off later if the overcurrent situation persisted.

Total Current 0x68 (dec=104)

This read-only register contains the total current of all 32 channels. If the PROGRAM/RUN bit of the CSR is at the PROGRAM value, it contains the sum of the raw ADC value of the current for all of the 32 channels. If the PROGRAM/RUN bit of the CSR is at the RUN value, it contains the sum of the corrected (pedestal subtracted) values. The data is 16-bit 2’s-complement format. If a High Voltage trip occurs, the data in this register will be frozen at the values that caused the trip.

This register is subject to rollover and incorrect data in the event of very large current values. If the average channel has a current reading of more than +/- 1000 units, the data in this register would be suspect.

Supervisor State Machine 0x69 (dec=105)

This read-only register provides access to the supervisor state machine. It shows the level of each state bit. The correlation between bits in this register and the state of the supervisor is:

 supv\_state[0] = (mon\_sm == init);

 supv\_state[1] = (mon\_sm == read1);

 supv\_state[2] = (mon\_sm == read2);

 supv\_state[3] = (mon\_sm == wait1);

 supv\_state[4] = (mon\_sm == wait2);

 supv\_state[5] = (mon\_sm == check1);

 supv\_state[6] = (mon\_sm == check2);

 supv\_state[7] = (mon\_sm == check3);

 supv\_state[8] = (mon\_sm == check4);

 supv\_state[9] = (mon\_sm == check5);

 supv\_state[10] = run\_mode;

 supv\_state[11] = kill\_hv\_ff;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *DSP Wait State Programming*

The DSP can address 3 regions of external memory. They are called “MS0”, “MS1”, and “MSH”. The linker “LDF” file defines the boundaries of each region. Each region can have a specific number of wait states for DSP operations. Reading and writing must both use the same number of wait states. A logic analyzer was used to examine all valid operations. The peripherals that the DSP can access and the minimum number of wait states are:

Region “/MS0”

rd ANA\_MEM min waitstates = 1

wr ANA\_MEM min waitstates = 0

wr ANA\_REG min waitstates = 0

Region “/MS1”

rd SRAM min waitstates = 1

wr SRAM min waitstates = 1

rd FLASH min waitstates = 4 !!! TOO SLOW

wr FLASH min waitstates = 10 !!! TOO SLOW

Region “/MSH”

rd XIL\_MEM min waitstates = 3

wr XIL\_MEM min waitstates = 0

rd XIL\_REG min waitstates = 2

wr XIL\_REG min waitstates = 1

wr TIM min waitstates = 1

The DSP initially uses 3 wait states for all operations. Without an external circuit to negate the “ACK” signal on the DSP and add wait states, 3 wait states is the maximum. As a result, the DSP cannot access the FLASH memory from a program. It can access the FLASH correctly when booting, because it uses special internal logic that inserts 16 wait states. The number of wait states can be changed by writing to the SYSCON register. The address of the SYSCON register is defined in “defts101.h”. A snippet of code that changes the number of waitstates for “MS0” and “MS1” from 3 to 1 is:

#include <defts101.h>

/\* Set up the SYSCON register \*/

/\* Configure 1 wait state for MS0 and MS1, 3 wait states for MSH \*/

/\* Configure slow protocol with idle cycles for all segments. \*/

/\* ‘OR’ of all the bits yields 0x000278E3 \*/

int \*syscon\_reg = (int \*)SYSCON\_LOC;

\*syscon\_reg =

SYSCON\_MSH\_SLOW | SYSCON\_MSH\_PIPE1 | SYSCON\_MSH\_WT3 | SYSCON\_MSH\_IDLE |

SYSCON\_MS1\_SLOW | SYSCON\_MS1\_PIPE1 | SYSCON\_MS1\_WT1 | SYSCON\_MS1\_IDLE |

SYSCON\_MS0\_SLOW | SYSCON\_MS0\_PIPE1 | SYSCON\_MS0\_WT1 | SYSCON\_MS0\_IDLE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *Matlab Programming using CBI\_NET*

This section contains some details about using Matlab to communicate with a system that has a ColdFire running CBI\_net.

The Matlab interface for Windows is implemented as a set of DLLs. These need to be in a directory that is in the Matlab search path. The functions currently available are:

matlab\_cbi\_net\_initialize\_sockets() :

called once before other socket functions

matlab\_cbi\_net\_init\_node('node\_name') :

node-name is target name as string, sets the active target

matlab\_cbi\_net\_net\_open() :

opens a socket for reading and/or writing

returns: socket\_number (Matlab type int32)

matlab\_cbi\_net\_close\_socket(socket\_number)

closes a previously opened socket

matlab\_cbi\_net\_rd\_mem(socket\_number, address, number\_of\_words, data\_type)

reads data from address on previously opened socket\_number

address is a decimal number

data\_type is one of:

mxSINGLE\_CLASS  = 7  ( 32 bit float = Matlab single)

mxINT16\_CLASS   = 10 ( Matlab int16 )

mxUINT16\_CLASS  = 11 ( Matlab uint16 )

mxINT32\_CLASS   = 12 ( Matlab int32 )

mxUINT32\_CLASS  = 13 ( Matlab uint32 )

The data read is interpreted and returned as an array of the indicated type. For 16 bit type you are implicitly setting the cbi\_net transfer size to 2 bytes.

returns: Matlab array of type data\_type

matlab\_cbi\_net\_wr\_mem(socket\_number, address, data\_array )

writes data\_array to socket\_number and address

data\_array is a Matlab array of one of the above mentioned types

matlab\_cbi\_net\_clear\_dsp\_flash(socket\_number)

Not yet tested!
clear the dsp flash at previously opened socket\_number

matlab\_cbi\_net\_wr\_dsp\_flash()

Not yet tested!

programs the dsp flash at previously opened socket\_number

prompts for file name,

opens socket to current target,

clears dsp flash

writes dsp flash

Here is an example Matlab session that writes to memory and reads back memory.

>> matlab\_cbi\_net\_initialize\_sockets();

>> matlab\_cbi\_net\_init\_node('klybpm06');

>> data=int32([99,98,97,96]);

>> sock = matlab\_cbi\_net\_net\_open()

>> matlab\_cbi\_net\_wr\_mem(sock, hex2dec('30200000'), data)

sock = 2300

>> matlab\_cbi\_net\_close\_socket(sock)

>> sock = matlab\_cbi\_net\_net\_open()

sock = 2300

>> matlab\_cbi\_net\_rd\_mem(sock, hex2dec('30200000'), 4, 12)

ans = 99          98          97          96

>> matlab\_cbi\_net\_close\_socket(sock)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## *A BSM Test Program in Matlab using CBI\_NET*

This section contains some details about testing a BSM module. The first goal is to read the pedestals (offsets) from the 32 ADCs.

1. Setup the cbi\_net interface

2. Disable the DSP.

 DSP\_RESET = 0

3. Disable data acquisition.

 DATA\_ACQ = 0

4. Set the channel delays to mid scale.

 CARD0\_DELAY = 512

 CARD1\_DELAY = 512

 CARD2\_DELAY = 512

 CARD3\_DELAY = 512

5. Set the global delays to minimum.

 GLOBAL0\_DELAY = 0

 GLOBAL1\_DELAY = 0

6. Enable all bunches.

 BUNCH\_PATTERN\_A = 0xffffffff

 BUNCH\_PATTERN\_B = 0xffffffff

 BUNCH\_PATTERN\_C = 0xffffffff

 BUNCH\_PATTERN\_D = 0xffffffff

 BUNCH\_PATTERN\_E = 0xffffffff

 BUNCH\_PATTERN\_F = 0xffffffff

7. Set the first memory location to zero.

 ADC\_MEM\_FIRST = 0

8. Set the turn request to 3. There is suspicion that the first and last turns may be corrupted!

 ACQ\_TURN\_REQ = 3

9. Enable acquisition.

 DATA\_ACQ = 1

10. Check for completion of acquisition.

 Wait till bit 2 in DATA\_ACQ is set

11. Disable acquisition

 DATA\_ACQ = 0

12. Read the ‘middle’ turn from each ADC. Add 183 to the base address of each ADC and read 183 data words starting at that address.

 Read 183 words from CARD0\_CHAN0 starting at (0x08000000 + 0xb7)

 Read 183 words from CARD0\_CHAN1 starting at (0x08080000 + 0xb7)

 .

 .

 Read 183 words from CARD3\_CHAN7 starting at (0x08f80000 + 0xb7)