1. Clock and Control Board CCB2004

This User’s Guide should be used together with the CCB2004 Specification [1] and the TTCrx Reference Manual [2]. The TTCrx ASIC can be programmed from the TTC source (four main registers, write only, see Section 1.1) and over I²C serial bus using VME accesses in the CCB2004 address space (write and read, see Section 1.2). JTAG access to TTCrx ASIC has not been implemented on the CCB2004 board.

1.1 Initialization

After power cycling make sure that the four green LEDs on the front panel indicating active powers as well as “DONE” LED (FPGA was successfully configured from its EPROM) are “on”. When optical connection between the TTCrq and the source of the TTC clock and commands (TTCvi [3] or TTCci modules) is established, the “TTCRDY” and “QLOCK” LEDs of the front panel of the CCB2004 must be “on”. They just repeat the state of the respective LEDs on TTCrq mezzanine board (also visible through the CCB2004 front panel). If an optical fiber is plugged in from both sides, and both source and destination operate properly, the connection will be established automatically after power cycling. Make sure the “CLK40” LED on the front panel is blinking (~7Hz). Then:

1. Program CSRA1 (write data = Fh for the “Discrete Logic” mode, write data = Eh for the “FPGA” mode). The only source of L1A for all boards in the crate in a “Discrete Logic” mode will be the TTCrx ASIC.
2. Program CSRB1. CSRB1[0]=0 for the “FPGA External” and CSRB1[0]=1 for the “FPGA Internal” modes. Use appropriate values to mask/unmask the L1A sources.
3. Write any data into CSRA3 (generate “Soft_Reset” for the FPGA). “Soft_Reset” command enables propagation of selected L1A and “external trigger” signals to custom backplane, disables 32-bit L1A counter and resets the CSRB11[10..8] bits. It does not affect the content of all other CSRB[1..18] registers. A “Hard_Reset” command is intended for recovery from possible Single Event Upsets in the LHC environment. It is not necessary to send a “Hard_Reset” during initialization.
4. Write appropriate delay values into CSRB5 (“FPGA” mode only).
5. (Optional) Read CSRB17 to verify the date of the firmware revision.
All the CCB2004 boards are equipped with the TTCrq mezzanines [2]. The jumpers on TTCrq board (see Chapter 8 in [2]) select the unique 14-bit ID address of the mezzanine that is required for the individually addressed TTC commands as well as for the I2C accesses to TTCrx. This address must be latched into the ASIC during initialization at the rising edge of the Reset_b signal. To do this the following steps are needed:

6. Send “Reset TTCrx ASIC” command (write any data to base+5C address of the CCB2004).
7. Wait at least 60 microseconds.
8. Read CSRB18. CSRB18[15..8]=00000001 for all the TTCrq mezzanines intended for the CSC system. Bits CSRB18[7..0] are unique for each CCB2004 and equal to a serial number (1..99) of the main CCB2004 board. This number is labeled on the front panel and on the main board as well. Out of these 8 bits, the lowest 6 bits are used to calculate the base I2C address, see Chapter 7 in [2]. The 14-bit ID address can also be read back and verified directly from the TTCrx ASIC over I2C bus, registers 16-18, see Chapter 3 in [2].
9. Read CSRA2 and CSRA3. Make sure CSRA3[13..12] = 1 and CSRA3[14] = 0. Check the other bits in CSRA2 and CSRA3 to verify that all the installed peripheral boards were configured successfully.
10. (Optional) Read the unique 64-bit board ID (see Section 1.3).

Initialization of the TTCrx ASIC requires the following steps (examples below are referred to TTCvi module. That same procedures can be performed over I2C bus as well):

11. Write fine delay values “xx” into “Fine Delay” registers 1 and 2 in the TTCrx ASIC (see Appendix A in [2] for more details):
   - write data = 8000h into base+C0 address on TTCvi board
   - write “xx” into base+C2 address on TTCvi board
   - write data = 100h into base+C2 address on TTCvi board.

12. Write coarse delay values “xx” into “Coarse Delay” register in the TTCrx ASIC (note bits [3:0] should be the same as bits[7:4]):
   - write data = 8000h into base+C0 address on TTCvi board
   - write data = 2xxh into base+C2 address on TTCvi board.

13. Program Control Register in the TTCrx ASIC:
   - write data = 8000h into base+C0 address on TTCvi board
   - write data = 3b3h into base+C2 address on TTCvi board (enable parallel output bus of the TTCrx, enable bunch counter and event counter operation, so the Trigger mode “11” is selected, see Chapter 6 in [2]).

14. Clear the discrete logic decoder on CCB2004 board with the long-format broadcast command = 0:
   - write data = 8001h into base+C0 register on TTCvi board
   - write data = 0 into base+C2 address on TTCvi board.
1.2. I²C Interface

The I²C is a 2-wire interface used on CCB2004 board for communications with the TTCrx ASIC residing on a TTCrq mezzanine board. General description of the I²C protocol can be found in [4], while the TTCrx specific implementation of the I²C functions can be found in Chapter 7 of the TTCrx Reference Manual [2]. I²C interface provides access to all the internal registers of the TTCrx ASIC from the VME through the CSR1[4..1] bits, see Section 6 of the CCB2004 specification [1].

Note that the correct operation of the I²C bus requires the TTCrx to be locked to the TTC signal (“TTC_Ready”=1), see Chapter 7 in [2].

Each TTCrq board has a unique 14-bit ID number encoded with the soldered resistors, as shown on Fig.9 [2]. A 6-bit portion Dout<5..0> of this 14-bit ID number defines the base I²C address as described in Table 12 [2]. On all production CCB2004 boards this 6-bit address is equal to serial number of the boards which is labeled on its front panel. For example, Dout<5..0>=32h on TTCrq which is installed on CCB2004 #50. This number must be loaded into the TTCrx ASIC at the rising edge of the Reset_b signal. A reset of the TTCrx (by writing any data to 68005Ch address of the CCB2004) should be done only once during CCB2004 initialization. The duration of the reset procedure should be ~60 microseconds (as described in steps 1-3 in Section 1.1. above), so the next operation should be delayed for that period. The following steps are required to perform a write (read) operation after that:

1. Generate “Start” condition (“high-to-low” transition on SDA line when SCL is “high”):
   - write data = Eh into CSRA1 (set both SDA and SCL lines “high”, i.e. inactive)
   - write data = Ah into CSRA1 (set SDA “low”)
   - write data = 2 into CSRA1 (set SCL “low”).

2. Write 7-bit address of the pointer register (MSB should be written first, LSB last). This address is calculated as Dout<5..0> * 2. During all write cycled the CSRA1[1] should be “1”. Data is written on “low-to-high” transition of SCL while SDA is valid. To load “1” the following steps are required:
   - write data = 6 into CSRA1 (set SDA = 1 while CSRA1[1]=0 and CSRA1[3]=0)
   - write data = Eh into CSRA1 (set SCL = 1)
   - write data = 6 into CSRA1 (set SCL = 0)
   - write data = 2 into CSRA1 (release SDA)

To load “0” the following steps are required:
   - write data = 2 into CSRA1 (set SDA = 0 while CSRA1[1]=0 and CSRA1[3]=0)
   - write data = Ah into CSRA1 (set SCL = 1)
   - write data = 2 into CSRA1 (set SCL = 0, SDA = 0)
Load all 7 bits of the pointer address as described above.

3. Write the 8th bit called “write” = 0.

4. Check the “acknowledgement” signal:
   - write data = 4 into CSRA1 (enable “read” operation, set SDA “high”)
   - write data = Ch into CSRA1 (set SCL “high” while SDA is “high”)
   - read CSRA1 and make sure the CSRA1[4]=0 (i.e. the cycle was acknowledged).

5. Generate “stop” condition (“low-to-high” transition on SDA while SCL is “high”).
   - write data = Eh into CSRA1 (set SDA = SCL = “high”)
   - write data = Ah into CSRA1 (set SDA “low”)
   - write data = 2 into CSRA1 (set SCL “low”).

6. Write the 8-bit number of the register to be addressed, MSB first. This could be any TTCrx internal register listed in Table 3 of [2]. The sequence is similar to one described in step 2 above.

7. Check the “acknowledgement” as described in step 4 above.

8. Generate “stop” condition.

9. Write data to the register selected by the pointer (or read data; see 9.1 below):
   - generate “start” condition
   - write 7-bit address of the data register as described in step 2. The address of the data register = address of the pointer + 1
   - check the “acknowledgement”
   - write 8-bit data into data register, similarly to step 6 above
   - check the “acknowledgement”
   - generate “stop” condition.

9.1. Read data from the register selected by the pointer
   - generate “start” condition
   - write 7-bit address of the data register to read data from. The address of the data register = address of the pointer + 1
   - check the “acknowledgement”
   - read 8-bit data. Each read cycle consists of the following steps:
     1. write data = 4 to CSRA1 (enable read, set SDA “high” or inactive)
     2. write data = Ch to CSRA1 (set SCL “high”)
     3. read CSRA1 and get the expected read value from CSRA1[4]
       - check the “no acknowledgement” state, i.e. read CSRA1 and verify that the CSRA1[4]=1
       - generate “stop” condition.

An example of the C++ program to perform I2C accesses to TTCrx ASIC is given in [5].

1.3. 1-Wire Interface

1-Wire is a proprietary Maxim – Dallas Semiconductor interface for communication with the Silicon Serial Number DS2401 chip that consists of a factory-lasered 64-bit ROM that includes a unique 48-bit serial number, an 8-bit CRC, and an 8-bit Family Code (01h). Data is transferred serially via the 1-Wire protocol, read and write least significant
The protocol details and timing diagrams are given in [6]. Access to serial number chip consists of three phases: Initialization, ROM Function Command, and Read Data.

The Initialization sequence consists of a “Reset pulse” transmitted by the master followed by a “Presence pulse” transmitted by the DS2401. The “Presence pulse” lets the bus master know that the DS2401 is on the bus and ready to operate. For the Initialization, the “Reset pulse” should be sent, then CSRB9[2] should be checked, and, when the CSRB9[2]=1, the CSRB9[0] should be read out. If CSRB9[0]=0 at this moment, that means that the “Presence pulse” was sent and the next step can be performed.

The ROM Function Command phase consists of sending a Read ROM command [33h] or [0Fh] to DS2401. The first bit (“Write-one”) should be sent, then CSRB9[4] scanned, and, when CSRB9[4]=1, the next bit of command should be sent. Since all commands are 8-bit long, eight write operations are necessary.

The Read Data phase consists of 64 read cycles. Each cycle starts with sending a “Read pulse”, then CSRB9[3] is scanned, and, when CSRB9[3]=1, the valid data bit should be received from CSRB9[1]. Note the first data bit should be “1” and the next seven bits should be “0” (they represent the Family Code 01h). Bits 49-56 are also “0” and bits 57-64 represent the CRC code.

An example of the C++ program to read the DS2401 serial ID can be found in [7].

1.4. JTAG Access to FPGA and EPROM and Firmware Upgrade

One Xilinx XC2V250-4FG456 FPGA and one XC18V02 EPROM are used on CCB2004 board. Both devices can be accessed over JTAG bus. JTAG protocol can be emulated using write and read operations directed to CSRA1[8..5]. The other optional access is possible using Xilinx Parallel Cable IV over the front panel connector. An on-board switch S10-1 defines which of these two options is selected. The EPROM is the 1st device in a JTAG chain, and the FPGA is the 2nd one. Files with the .mcs and .svf extension (produced by Xilinx ISE development system) are needed to reprogram the EPROM with the Xilinx downloading cable and VME path respectively. The most recent version of the downloading file can be found in [8].

Four bits of CSRA1[8..5] (bit 5 for TDI, bit 6 for TMS, bit 7 for TCK, bit 8 for TDO) are used to implement the JTAG protocol. These four Test Access Point (TAP) pins control how data is scanned into the various registers. The state of the TMS pin at a rising edge of TCK determines the sequence of state transitions. There are two main sequences, one for shifting data into the data register and the other for shifting an instruction into the instruction register (see State Diagram for the TAP Controller on Fig.1). Below is an example of how to read the 32-bit IDCODE code from Xilinx XC18V02 EPROM over JTAG using VME accesses. Datasheet [9] is essential for understanding of the JTAG access to Xilinx EPROM.
1. Set **Test-Logic-Reset** mode 5 times: repeat 5 times the following sequence:
   - write data = **4Fh** into CSRA1 (set TMS =1)
   - write data = **CFh** into CSRA1 (set TCK =1)
   - write data = **4Fh** into CSRA1 (set TCK = 0)
   - write data = **0Fh** into CSRA1 (set TMS = 0).

2. Set **Run-Test/Idle** mode:
   - write data = **0Fh** into CSRA1 (set TMS = TCK = TDI =0)
   - write data = **8Fh** into CSRA1 (set TCK =1)
   - write data = **0Fh** into CSRA1 (set TCK =0).

3. Set **Select-DR-Scan** mode:
   - write data = **4Fh** into CSRA1 (set TMS =1)
   - write data = **CFh** into CSRA1 (set TCK =1)
   - write data = **4Fh** into CSRA1 (set TCK = 0)
   - write data = **0Fh** into CSRA1 (set TMS = 0).

4. Set **Select-IR-Scan** mode:
   - write data = **4Fh** into CSRA1 (set TMS =1)
   - write data = **CFh** into CSRA1 (set TCK =1)
   - write data = **4Fh** into CSRA1 (set TCK = 0)
   - write data = **0Fh** into CSRA1 (set TMS = 0).
5. Set **Capture IR-Scan** mode:
   - write data = 0Fh into CSRA1 (set TMS = TCK = TDI = 0)
   - write data = 8Fh into CSRA1 (set TCK = 1)
   - write data = 0Fh into CSRA1 (set TCK = 0).

6. Set **Shift-IR-Scan** mode:
   - write data = 0Fh into CSRA1 (set TMS = TCK = TDI = 0)
   - write data = 8Fh into CSRA1 (set TCK = 1)
   - write data = 0Fh into CSRA1 (set TCK = 0).

7. Send 6-bit **Bypass** instruction “111111” to XC2V250 FPGA. To do this, repeat 6 times the following sequence:
   - write data = 2Fh into CSRA1 (set TDI = 1)
   - write data = AFh into CSRA1 (set TCK = 1)
   - write data = 2Fh into CSRA1 (set TCK = 0)
   - write data = 0Fh into CSRA1 (set TDI = 0).

8. Send 8-bit **IDCODE** instruction “11111110” (FEh) to EPROM. Repeat 8 times a process similar to step 7. LSB sent first, MSB sent last along with the TMS bit.

9. Set **Update-IR** mode:
   - write data = 4Fh into CSRA1 (set TMS = 1)
   - write data = CFh into CSRA1 (set TCK = 1)
   - write data = 4Fh into CSRA1 (set TCK = 0)
   - write data = 0Fh into CSRA1 (set TMS = 0).

10. Set **Select-DR-Scan** mode:
    - write data = 4Fh into CSRA1 (set TMS = 1)
    - write data = CFh into CSRA1 (set TCK = 1)
    - write data = 4Fh into CSRA1 (set TCK = 0)
    - write data = 0Fh into CSRA1 (set TMS = 0).

11. Set **Capture-DR** mode:
    - write data = 0Fh into CSRA1 (set TMS = TCK = TDI = 0)
    - write data = 8Fh into CSRA1 (set TCK = 1)
    - write data = 0Fh into CSRA1 (set TCK = 0).

12. Set **Shift-DR** mode:
    - write data = 0Fh into CSRA1 (set TMS = TCK = TDI = 0)
    - write data = 8Fh into CSRA1 (set TCK = 1)
    - write data = 0Fh into CSRA1 (set TCK = 0)
    - read CSRA1[8]. This is bit 0 of the 32-bit IDCODE.

13. **Repeat step 12 31 times** to get the other 1..31 bits of IDCODE.

14. Set **Update-DR** mode
    - write data = 4Fh into CSRA1 (set TMS = 1)
    - write data = CFh into CSRA1 (set TCK = 1)
    - write data = 4Fh into CSRA1 (set TCK = 0)
    - write data = 0Fh into CSRA1 (set TMS = 0).

The IDCODE assigned to XC18V02 EPROM is 05025093h. The IDCODE from the FPGA can be read out in a similar way. An example of the C++ program to reprogram the EPROM over VME can be found in [10].
1.5. CCB2004 Test with the TTCvi/TTCvx and TMB2005

The CCB2004 resides in the peripheral crate with at least one TMB2005 board. Optical connection to TTC source (a pair of TTCvi [3] and TTCvx [11] boards) is established. The test allows to verify internal functionality of the CCB2004 as well as distribution of the TTC broadcast commands and L1A signal over custom peripheral backplane to one or more TMB2005 [12] boards. The following procedures are required:

1. Verify that the TTC link is established ("TTCRDY" and "QLOCK" LED on the front panel are “on” and corresponding bits from CSRA3[13]=CSRA3[14]=1). Also make sure the FPGA was configured successfully (CSRA3[12]=1).
2. Make sure the “CLK40” LED on the front panel is blinking at a frequency of ~7Hz. If not, there is a problem with clock recovery/distribution from the TTCrx.
3. (Optional) read CSRB17 and verify the date of the firmware revision.
4. Run VME access test: write, read and verify the content of CSRB1…CSRB8.
5. Initialize the CCB2004 as described in Section 1.1. Set “Discrete Logic” mode. Read CSRB18 and verify the value returned on CSRB18[7:0] against the serial number of the CCB2004 board (labeled on the front panel).
6. Write data = DFF4h into CSRB1 (TTCrx is a source of broadcast commands, L1A from the TTCrx is enabled).
7. Write data = 4 into CSR1 of TTCvi (address $80) to enable L1A generation on VME command.
8. Write any data to address Base+94h on CCB2004 (reset L1ACC counter)
9. Write any data to address Base+96h on CCB2004 (enable L1ACC counter to count).
10. Write data = 1 into address $C4h of TTCvi (broadcast command = BCntRes). Make sure the “BCRES” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking.
11. Write data = 2 into address $C4h of TTCvi (broadcast command = EvCntRes). Make sure the “EVCRES” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking.
12. Write data = 4 into address $C4h of TTCvi (broadcast command = 1, or BC0). Make sure the “BC0” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Read back the CSRB15 and verify that the returned value = 4. Read back the ADR_CCB_STAT from available TMB2005 board(s) and verify that the returned value is “1” (note there is no 2-bit shift to the left in the ADR_CCB_STAT register).
13. Write data = Ch into address $C4h of TTCvi (broadcast command = 3, or L1Reset). Make sure the “L1RES” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Read back the CSRB15 and verify that the returned value = Ch. Read back the ADR_CCB_STAT from available TMB2005 board(s) and verify that the returned value is “3” (note there is no 2-bit shift to the left in the ADR_CCB_STAT register).
14. Write data = 10h into address $C4h of TTCvi (broadcast command = 4, or Hard_Reset). Make sure the “HRESET” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Read back the CSRB15 and verify that the
returned value = 10h. Make sure the TMB2005 board(s) react to this command (LEDs on the front panel).

15. Write data = 3Ch into address $C4h of TTCvi (broadcast command = Fh, or CCB_Hard_Reset). Make sure the “CCBHR” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Read back the CSRB15 and verify that the returned value = 3Ch.

16. Write data = 40h into address $C4h of TTCvi (broadcast command = 10h, or TMB_Hard_Reset). Make sure the “HRESET” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Make sure the TMB2005 board(s) react to this command (LEDs on the front panel).

17. Write data = 4Ch into address $C4h of TTCvi (broadcast command = 13h, or MPC_Hard_Reset). Make sure the “HRESET” and “CMDSTR” LEDs on the front panel of the CCB2004 are blinking. Make sure the MPC2004 reacts to this command (LEDs on the front panel).

18. Generate the other short broadcast commands by writing corresponding codes into address $C4h of TTCvi and verifying the returned values from CSRB15 (CCB2004) and ADR_CCB_STAT (TMB2005) registers.

19. Check propagation of the L1A from the TTC source down to TMB2005 board(s):
   - write data = 1, then data = 0 into ADR_CNT_CTRL of TMB2005 (clear all counters)
   - write any data to address $86h of TTCvi (N=1..100000) times to generate (N) L1A pulses. Make sure the “L1A” LED on the front panel of the CCB2004 is blinking.
   - read back the data from Base+90h and Base+92h addresses of CCB2004 and verify the returned values against the number (N) of transmitted L1A
   - write data = 2 into ADR_CNT_CTRL of TMB2005 (take snapshot of current counter state)
   - write data = 2800h into ADR_CNT_CTRL of TMB2005
   - read back the content of ADR_CNT_RDATA and verify against the number of L1A transmitted (lowest 16 bits)
   - write data = 2900h into ADR_CNT_CTRL of TMB2005
   - read back the content of ADR_CNT_RDATA and verify against the number of L1A transmitted (highest 16 bits)
   - (Optional, only in “Discrete Logic” or “FPGA External” mode). The 12-bit Bunch and 24-bit Event counters are multiplexed on the BCnt<11:0> counter output bus of the TTCrx ASIC. Upon reception of an L1A signal, the TTCrx makes the content of these counters (depending on “Trigger Mode” as defined by bits 0 and 1 in the control register of the TTCrx) available on BCnt<11:0> bus with the respective strobes. These values are latched into the three registers CSRB12-CSRB14 on the CCB2004 board and available for read. The following steps are required to test this functionality:
     a. send BCRES and EVCRES broadcast commands from the TTC source (load data = 3 into $C4h address on TTCvi)
     b. send (N) L1A pulses from the TTC source
c. read **CSRB12** and make sure the returned value is changing (bunch counter counts permanently)
d. read **CSRB13** and **CSRB14** and verify the returned value against the values read out from the CCB2004 and TMB2005 (above). Note that the first event will be marked as event number zero, so the value in CSRB13 and CSRB14 should always be \((N-1)\) while the content of L1A counters available from the CCB2004 and TMB2005 should be \((N)\).

20. Check propagation of the long-format asynchronous cycles from the TTC:
- write data = **8001h** into address **SC0h** of TTCvi (broadcast command with TTCrx=0)
- write data = **0…FFFFh** into address **SC2h** of TTCvi (send command)
- make sure the “DATSTR” LED on the front panel of CCB2004 is blinking
- read **CSRB16** and verify the returned value against the data in register **SC2h**.

21. Switch to “**FPGA External**” mode:
- write data = **Eh** into **CSRA1**
- write data = **DFF4h** into **CSRB1**

Repeat steps 7-21.

22. Switch to “**FPGA Internal**” mode:
- write data = **Eh** into **CSRA1**
- write data = **DFE9h** into **CSRB1** (CSRB2 is a source of short broadcast commands, CSRB3 is a source of long commands, L1A will be generated on VME command).

23. Repeat steps 8-20 with the **CSRB2** as a source of short broadcast commands (instead of register **SC4h**) and **CSRB3** as a source of long commands instead of register **SC2h**. Write any data to address **Base+54h** on CCB2004 to generate the L1A pulse.

24. (Optional). Check propagation of the TMB_L1A_Request signal from the TMB2005 to CCB2004. This signal causes generation of L1A, when enabled. The following steps are required:
- write data = **DFDEh** into **CSRB1** (Enable TMB_L1A_Request)
- write data = 1, then data = 0 into **ADR_CNT_CTRL** of TMB2005 (clear all counters)
- initialize the TMB_L1A_Request generator on TMB2005 board. The generator starts generating 25 ns L1A_Request pulses (at ~3MHz frequency) on “Start Trigger” and stops generating on “Stop Trigger” commands:
  a. write data = **3dh** into **ADR_CCB_CFG** of TMB2005
  b. write data = **7204h** into **ADR_CCB_TRIG**
  c. write data = 0 into **ADR_ALCT_INJ**
  d. write data = **5h** into **ADR_ALCT_INJ**
  e. write data = 85h into **ADR_ALCT_INJ**
  f. write data = 3 into **ADR_TMBTIM**
g. write data = FFE0h into ADR_CFEB_INJ
h. write data = 3FFh into ADR_SEQ_TRIG_EN
i. write data = 7C00h into ADR_TRIG_EN
j. write data = FFFh into ADR_SEQ_L1A
k. write data = 1 into ADR_TRIG_EN
l. write data = FFFDh into ADR_ALCT_INJ
m. write data = 1 into ADR_CCB_CMD
n. write data = 603h into ADR_CCB_CMD (“Start Trigger”) 
o. write data = 1 into ADR_CCB_CMD
p. write data = 103h into ADR_CCB_CMD (“BC0”) 
q. write data = 1 into ADR_CCB_CMD
r. write data = 7FFh into ADR_CFEB_INJ_ADR
s. write data = 8000h into ADR_CFEB_INJ_ADR
t. write data = 7FFh into ADR_CFEB_INJ_ADR
u. write data = 1 into ADR_CCB_CMD
v. write data = 703h into ADR_CCB_CMD (“Stop Trigger”) 
w. write data = 1 into ADR_CCB_CMD

- read back the content of Base+90h and Base+92h registers of CCB2004
- write data = 2 into ADR_CNT_CTRL of TMB2005 (take snapshot of current counter state)
- write data = 2800h into ADR_CNT_CTRL of TMB2005
- read back the content of ADR_CNT_RDATA and verify it against the value that was read out from the Base+90h address (lowest 16 bits)
- write data = 2900h into ADR_CNT_CTRL of TMB2005
- read back the content of ADR_CNT_RDATA and verify it against the value that was read out from the Base+92h address (highest 16 bits).

1.6. CCB2004 Test in the Track Finder Crate

Repeat steps 1-5 described in Section 1.5. Make sure the “NO LOCK” LED on the front panel of all SP05 boards in the Track Finder crate is “off” (the QPLL on SP05 board must be locked to the backplane clock provided by CCB2004).

The SP05 firmware allows to analyze timing of the CCB2004 command strobe with respect to the SP05 system clock, as well as the CCB2004 command itself, in the CCB analyzer STS_ANA [13]. The analyzer is 64 words deep and available for read from VME. To check command transmission from the CCB2004 to SP05 the following steps are required:

- set SP05 under the CCB fast control (write “0” into CSR_FCC)
- reset CCB analyzer (write any data into STS_ANA)
- send 1..64 sample commands from the CCB2004
- read 1..64 words from STS_ANA and verify them against expected values. Make sure the VME_FPGA on SP05 board samples the CCB command in the middle of the valid sample. See Section STS_ANA in [13] for more details.
References

2. Muon Port Card  MPC2004

This User’s Guide should be used together with the MPC2004 Specification [1].

2.1. Initialization

After power cycling make sure that the six green LEDs on the front panel indicating active on-board powers as well as the “DONE” LED (FPGA was successfully configured from its EPROM) are “on”. Make sure that the “CLK40” LED on the front panel is blinking (~7Hz). This means that the main 40Mhz clock on MPC2004 board is active. Then:

1. Read CSR0. Make sure the CSR0[12]=1 (which means the FPGA was successfully reloaded from the EPROM).
2. (Optional). Read CSR1 and check the date of the firmware revision.
3. Program CSR0 with the Board_ID[5..0], FPGA_Mode, CSR0[13]=1 (by default, set the clock in the middle of the “safe window”), CSR0[14]=1 (enable all serializers), CSR0[15]=0 (normal mode of operation) or CSR0[15]=1 (to run a PRBS test of optical links to Sector Processor).

Note: Bit CSR0[13] is effective only if the on-board switch S2-2 is “on” and S2-1 is “off”. Switch S2 selects the source of the 40MHz clock (fixed clock from the CCB2004 or adjustable clock with the delay element) for the FPGA. If S2-2 is “off” and S2-1 is “on”, the fixed CCB2004 clock is chosen. This clock is set approximately in the middle of the “safe window”.

4. Send “Soft_Reset” (write any data to address 600004h). This command resets all the FIFO buffers in the FPGA. It does not affect any CSR registers. “Hard_Reset” command is intended for recovery from possible Single Event Upsets in the LHC environment. It is not necessary to send the “Hard_Reset” during initialization.
5. (Optional). Load CSR2 if needed. By default (after power cycling or “Hard_Reset”), CSR2=”0”.
6. (Optional). Load CSR4 if the “transparent” mode will be used. By default (after power cycling or “Hard_Reset”) CSR4=0, and the MPC2004 is in a “sorter” mode.

2.2. 1-Wire Interface

1-Wire bus is intended for obtaining a unique 64-bit serial ID number from the DS2401 chip. The CSR6 should be used. Its bit assignment is identical to CSRB9 on CCB2004 (see Section 1.3 for more details), so the same program can be used to read the ID from both the CCB2004 and MPC2004 boards.
2.3. JTAG Access to FPGA and EPROM and Firmware Upgrade

One Xilinx XCV600E-8FG680 FPGA and one XC18V04 EPROM are located on the mezzanine board. Both devices can be accessed over JTAG bus. JTAG protocol can be emulated using write and read operations directed to CSR0[8..5]. Xilinx Parallel Cable IV can be used as well. An on-board switch S8-1 defines which of these two options is set. The FPGA is the 1st device in a JTAG chain, and the EPROM is the 2nd one. Files with the .mcs and .svf extension (produced by Xilinx ISE development system) are needed to reprogram the EPROM with the Xilinx downloading cable or VME path respectively. The most recent versions of downloading files can be found in [2].

An IDCODE from XC18V04 EPROM can be obtained over JTAG as described in Section 1.4. Note that the BYPASS instruction for Virtex-E FPGA is a 5-bit “11111”. The IDCODE assigned to XC18V04 EPROM is 05026093h.

An example of the code to reprogram the XC18V04 over VME can be found in [3].

2.4. MPC2004 Self-Test

The MPC2004 is located in the peripheral crate (slot 12) and set to a “Test” mode. Data patterns are loaded into FIFO_A, sent through the sorter unit and checked out from the FIFO_B. All FIFO buffers are 511 words deep. The procedures are the following:

1. Write any data to address 600004h of MPC2004 (generate “Soft_Reset” to the FPGA).
2. Initialize the CCB2004 board in the peripheral crate in “Discrete Logic” or “FPGA External” mode and generate a “Soft_Reset” for the CCB2004 (Section 1.1 above).
3. Write data = 6A01h into CSR0 (set “Test” mode).
4. Read CSR3 and make sure the returned value is A(hex) (both FIFO_A and FIFO_B buffers are empty).
5. Load 255 32-bit words (510 frames) of data into FIFO_A[1..9]. They will represent the incoming LCT’s for the sorter unit.
6. Load the last word (2 frames) = 0 into all FIFA_A[1..9] buffers.
7. Send broadcast command = 30h from the TTC source to inject data from FIFO_A into the sorter unit.
8. Read CSR3 and make sure that FIFO_A is empty and FIFO_B is not empty (if the patterns in FIFO_A were representing valid LCT’s).
9. Read 510 words from FIFO_B[1..3] and verify they are equal to expected values according to FIFO_A content and the sorting algorithm.
10. Read CSR3 and verify the returned value is Ah (both FIFO_A and FIFO_B buffers are empty).
2.5. TMB2005 – to – MPC2004 Data Transmission Test

This test involves the CCB2004, MPC2004 and 1..9 TMB2005 boards residing in the EMU peripheral crate. Understanding of the TMB2005 functionality, and its internal registers is essential for this test, see [4] for details. This test also allows to measure the “safe window” (i.e. the fraction of the 80Mhz clock period within which the data from all nine TMB2005 boards can be safely latched in into the MPC2004). The following procedures are needed:

1. Program CCB2004 in the peripheral crate (slot 13):
   - write data = 0 into CSRA1 (set “FPGA” mode)
   - write data = FFFFh into CSRB1 (set “Internal” mode)
   - write any data to CSRA3 (generate “Soft_Reset”)

2. (Optional; required only if the CCB2004 is in “Discrete Logic” mode) Initialize the TTCrx ASIC from the TTC source as described in Section 1.1 above.

3. Program and initialize the MPC2004 (slot 12):
   - write any data into address 600002h (generate “Soft_Reset” to the FPGA)
   - (Optional) read back CSR3 and make sure the returned value is Ah (both FIFO_A and FIFO_B buffers are empty)
   - write data = 4A1Eh into CSR0 (set “Trigger” mode)
   - write data = 3000h into CSR2 (adjust input clock in the middle of “safe window”).

4. Initialize every TMB2005 board in the crate:
   - write data = 1 into ADR_ALCT_CFG (Blank ALCT received data)
   - write data = 7C00h into ADR_CFEB_INJ (turn off all CFEBn inputs)
   - write data = 00E0h into ADR_TMB_TRIG (set mpc_rx_delay[3:0]= 7)
   - write data = 2 into ADR_CCB_CFG (disable CCB backplane outputs)
   - write data = 02FFh into ADR_MPC_INJ (enable injector start by TTC command; number of LCT pairs to inject = 255, or 510 frames).

5. Program LCT patterns to be injected into MPC2004 on every TMB2005 board:
   a. Load the 1st frame of LCT0 into ADR_MPC_RAM_WDATA
      - write data = 0 into ADR_MPC_RAM_ADR
      - write data = 1 into ADR_MPC_RAM_ADR
      - write data = 0 into ADR_MPC_RAM_ADR
   b. Load the 2nd frame of LCT0 into ADR_MPC_RAM_WDATA
      - write data = 0 into ADR_MPC_RAM_ADR
      - write data = 2 into ADR_MPC_RAM_ADR
      - write data = 0 into ADR_MPC_RAM_ADR
   c. Load the 1st frame of LCT1 into ADR_MPC_RAM_WDATA
      - write data = 0 into ADR_MPC_RAM_ADR
      - write data = 4 into ADR_MPC_RAM_ADR
      - write data = 0 into ADR_MPC_RAM_ADR
   d. Load the 2nd frame of LCT1 into ADR_MPC_RAM_WDATA
      - write data = 0 into ADR_MPC_RAM_ADR
      - write data = 8 into ADR_MPC_RAM_ADR
- write data = 0 into ADR_MPC_RAM_ADR

6. Repeat step 6 for other LCT patterns (maximum 255 pairs of LCT0+LCT1). Note the mpc_adr[7:0] in ADR_MPC_RAM_INJ should increase for every pair (four frames) of LCTs. For example, the second pair of LCT0+LCT1 should be loaded as shown below:

   a. Load the 1st frame of LCT0 into ADR_MPC_RAM_WDATA
      - write data = 0100h into ADR_MPC_RAM_ADR
      - write data = 0101h into ADR_MPC_RAM_ADR
      - write data = 0100h into ADR_MPC_RAM_ADR

   b. Load the 2nd frame of LCT0 into ADR_MPC_RAM_WDATA
      - write data = 0100h into ADR_MPC_RAM_ADR
      - write data = 0102h into ADR_MPC_RAM_ADR
      - write data = 0100h into ADR_MPC_RAM_ADR

   c. Load the 1st frame of LCT1 into ADR_MPC_RAM_WDATA
      - write data = 0100h into ADR_MPC_RAM_ADR
      - write data = 0104h into ADR_MPC_RAM_ADR
      - write data = 0100h into ADR_MPC_RAM_ADR

   d. Load the 2nd frame of LCT1 into ADR_MPC_RAM_WDATA
      - write data = 0100h into ADR_MPC_RAM_ADR
      - write data = 0108h into ADR_MPC_RAM_ADR
      - write data = 0100h into ADR_MPC_RAM_ADR.


8. Send the TTC broadcast command = 24h (“Inject test patterns from the TMB”)

9. (Optional) Read CSR3 of MPC2004 and make sure FIFO_B is not empty (if valid patterns from TMB2005 boards are expected).

10. Read FIFO_B[1..3] from MPC2004 (255 words = 510 frames) and verify the returned values against expected, according to number of participating TMB2005 boards, content of their MPC_RAM, and sorting algorithm of the MPC2004.

11. (Optional) Read CSR3 of MPC2004 and verify the FIFO_B is empty.

12. Read “winner bits” (called mpc_accept[1:0] in the TMB manual [4]) from the injector RAM of all participating TMB2005 boards. To do this:

    a. Load ADR_MPC_RAM_ADR to be read out, starting from address 0 for mpc_adr[7:0]: mpc_wen[3:0]=mpc_ren[3:0]=0

    b. Read ADR_MPC_INJ, check bits mpc_accept[1:0] and verify that they correspond to results of sorting for specific patterns loaded into this TMB2005.

Note 1: Due to TMB-to-MPC-to-TMB propagation delays, the first several (usually, nine) words in the injector RAM will have the “MPC accept bits” [11:10] (“winners”) = “0” and only the tenth and further words will represent the real MPC2004 “winner” responses.

Note 2: The four “MPC accept response delay” bits in the ADR_TMB_TRIG register do not affect the content of the
injector RAM with “winner” responses. These four bits specify the delay (by default = 7) to latch the two “mpc_accept[1:0]” bits into the ADR_TMB_TRIG register.

c. Increase mpc_adr[7:0] (up to FFh) and go to step b) above.

To measure the “safe window”, the value loaded into the CSR2 on Step 3, should vary typically between 2000h to 4000h (use 2100h, 2200h… 4000h); each step corresponds to 0.25 ns. For each step we recommend to run 300..500 iterations (up to 1000 iterations on the boundaries of the “safe window”); each iteration comprises 255 random data patterns to be loaded into every TMB2005. The “safe window” corresponds to error-free data transmission. Based on our experience, the average “safe window” is ~5.5 ns (CSR2[15:8]=25h..3Bh). Note the S2-1 on MPC2004 board should be “off” and S2-2 “on” to allow this test. If S2-1 is “on” and S2-2 “off”, the clock will be automatically set approximately in the middle of the “safe window” and the test should show no errors for any value in CSR2. It is essential that all the nine TMB2005 boards participate in this measurement.

2.6. Test of optical links with the PRBS patterns

The TLK2501 transceiver has an embedded pseudo-random bit stream (PRBS) generator that makes testing of optical links between the MPC2004 and SP05 boards [5] quite simple. The following steps are required:

1. Establish connection (all three links) between the MPC2004 under the test and SP05.
2. Write data = CA00h into CSR0 of MPC2004 (program PRBS mode).
3. Write data = 50h into CSR_LNK register of SP05 for all three links under test.
4. Write data = 1Fh into ACT_LCR to reset all error counters on SP05 board for selected links.
5. Make sure that three yellow LEDs on the front panel of the SP05 for selected links are “on”. At this point the PRBS is running continuously.
6. Read three CSR_LNK registers of SP05 for selected links in a loop and make sure the returned values are 750h (all error counters = 0). See more details on CSR_LNK register in the SP05 manual [5].

2.7. Test of optical links with the programmable data patterns

The MPC2004 is located in the peripheral crate, and the SP05 board is located in the Track Finder crate. The CCB2004 boards in both crates are connected to the TTC clock and command source (TTCvi or TTCci board) in one of these two crates or in a separate crate. The TMB boards are not involved in this test, so the MPC2004 is in a “Test” mode.

1. Program CCB2004 in both crates:
   - Initialize both CCB2004 boards in the peripheral and Track Finder crates as described in Section 1.1 of this Guide.
   - Set both CCB2004 into “Discrete Logic” mode (load “1” into CSRA1)
   - Send “Soft_Reset” to both CCB2004 (write any data to Base+4 address)
- Make sure the TTCrx on both CCB2004 boards are “Ready” and OPLL are “Locked” (check corresponding LEDs on the front panel or/and read CSRA3).

2. Establish connection (all three links) between the MPC2004 under test and the SP05 (use only one triple link out of F1..F5, typically the bottom F1). Optical fibers can be of different length.

3. Program MPC2004 in the peripheral crate:
   - send “Soft_Reset” to FPGA on MPC2004 (write any data into 600004h)
   - write data = 6A1Fh into CSR0 (note: Board_ID[5:0] can be any)
   - load 255 programmable LCT patterns (510 frames) into FIFO_A[9..1]
   - load the last pattern = 0 into all FIFO_A[9..1] buffers.

4. Program SP05 in the Track Finder crate:
   - write data = 0 into CSR_FCC VM (set SP05 to CCB control)
   - write data = 10h into CSR_LNK FA MA (enable all three links)
   - write data = 9050h into CSR_SFC VM (set delay of spy FIFO write)
   - write data = 11FFh into CSR_SFC FA MA (set how many words to expect)
   - write data = 31FFh into CSR_SFC SP (set spy FIFO window)
   - write data = 4h into ACT_XFR FA MA (reset spy FIFO)
   - write data = 4h into ACT_XFR SP (reset spy FIFO)
   - write data = 6Dh into CSR_AFD FA MA (set alignment FIFO read delay)

5. Send L1Reset broadcast command (=3h) from the TTC source to both CCB2004 boards.

6. Send “Inject patterns from MPC” command (=30h) from the TTC source to both CCB2004 boards.

7. Read 255 patterns (510 frames) from FIFO_B[3..1] on MPC2004 and make sure they satisfy with the sorting criteria.

8. Read 255 patterns (510 frames) from DAT_SF F1 M1/M2/M3 (for each muon) and make sure they satisfy with the sorting criteria and coincide with the data read out of FIFO_B[3..1] on MPC2004.

9. Read CSR_SF spy FIFO status and make sure the spy FIFO is empty (SFEF=1).

10. Read CSR_BID and make sure the MPC_LINK_ID[7:0] is equal to expected number (MPC_ID[5:0] as pre-programmed into the CSR0 of MPC2004 and LINK_ID[1:0] as hard-wired on SP05).

Note: You will probably need to adjust the delay of spy FIFO write (in CSR_SFC VM) and the delay of alignment FIFO read (in CSR_AFD FA).

References

See also http://www.phys.ufl.edu/~uvarov/SP05/SP05.htm
3. Muon Sorter MS2005

This User’s Guide should be used together with the MS2005 Specification [1].

3.1. Initialization

After power cycling make sure that the four green LEDs on the front panel indicating active on-board powers as well as the “DONE” LED (FPGA was successfully configured from its EPROM) are “on”. Make sure that the “CLK40” LED on the front panel is blinking (~7Hz). This means that the main 40MHz clock on MS2005 board is active. Then:

1. Write data = 12A0h into CSR0 (set “Trigger” mode).
2. (Optional) Read CSR4 and check the date of the firmware revision.
3. Make sure the CSR3[0]="1" (FPGA was configured and locked successfully).
4. Send “Soft_Reset” (write any data to address 700018h). This command resets all the FIFO buffers in the FPGA. It does not affect any CSR registers. “Hard_Reset” command is intended for recovery from possible Single Event Upsets in the LHC environment. It is not necessary to send ‘Hard_Reset’ during initialization. After power cycling the “Hard_Reset” is disabled. To enable it, the CSR4[1] should be set to “1”.
5. Program CSR9[11..0] to enable or disable input data streams from a specific Sector Processor(s) in the Track Finder crate. After power cycling all the SP’s are enabled.
6. Program CSR8. All the 12 data streams from SP1..SP12 are latched into the main Xilinx FPGA on a common 80Mhz clock that is derived from the Master 40Mhz provided by the CCB2004 board. This clock is produced inside the FPGA using its Digital Control Module (DCM, see [2]) and must be set in the middle of the “safe window” for all 12 SP sources. Fine clock adjustments within the 12.5 ns clock period can be done using two independent methods. The first (A) method is preferred.

   A. The Master 40Mhz clock from the CCB2004 passes through the 3D7408-025 [3] delay chip before it reaches the FPGA. The delay is programmable with the CSR8[7..0] and the minimal step of adjustment is 250 ps. This delay value can be programmed in such a way that the 80Mhz clock produced from the delayed 40Mhz Master clock will be set exactly in the middle of the “safe window”. It was measured that the “safe window” corresponds to CSR8=78..101(dec), so the recommended setting is CSR8=89(dec)=59h. Then it is not necessary to program the DCM1 which delay is set to “0” automatically on power cycling.

   B. For a fixed value of CSR8, the resulting 80Mhz clock within the FPGA can be delayed using the DCM1 with a 100 ps precision and set in the middle of the “safe window”. It is recommended to program CSR8[7..0]=0 (this is also a default value after power cycling). Then,
according to our measurements, the “safe window” corresponds to DCM1 settings between –66 and +7. The middle of the “safe window” corresponds to DCM1 setting of –30. The procedure to program the DCM1 with value = “-30” is the following:

B.1. Write “0” to address 700160h (set up fine phase adjustment for DCM1)
B.2. Read CSR5 and make sure the CSR5[0]=1 (clock de-skew was done)
B.3. Write any data to address 700164h (reset PSDONE status bit)
B.4. Repeat B.1.-B.3. 29 times.

7. Write any data to 70016Ah address (set “winner bit” mode).
8. Load all Rank and Phi LUT RAM’s and (optional) read them back for verification.

3.2. JTAG Access to FPGA and EPROM and Firmware Upgrade

The Xilinx XCR3128 PLD on the main MS2005 board performs the VME A24D16 slave functions and controls the operation of the Fairchild SCANPSC100 controller [4]. This controller supports VME accesses to the Xilinx XC2V4000-5FF1152C FPGA and four XC18V04 EPROM’s located on the mezzanine board.

The PLD can be programmed over Xilinx Parallel Cable IV only using an on-board 14-pin connector P11. File with the .jed extension for the PLD can be found in [5]. Four EPROM’s can be programmed over Parallel Cable IV as well using another on-board 14-pin connector P10. Note that CSR4[0] should be set to “1” to enable the SCANPSC100 controller. By default the CSR4[0]=0 and the cable connection is enabled.

Files with the .mcs and .svf extensions (produced by Xilinx ISE development system) are needed to reprogram the EPROM with the Xilinx Parallel Cable and VME path respectively. The most recent versions can be found in [5].

To reprogram the EPROMs over VME the following steps are needed:

1. Write data = 1 into CSR4.
2. Use LoadFPGAconsole.exe, which is a UF program to load the SP firmware, with the following flags:

   LoadFPGAconsole.exe -s14 -j0 -x[SP xml file] [MS svf file]

3.3. MS2005 Self-Test

The MS2005 is set to “Test” mode. Data patterns are loaded into FIFO_A, sent through the sorter unit and checked out from FIFO_C and FIFO_B. All FIFO buffers are 511 words deep. An external cable can be connected between one of the output connectors on the front panel and the P12 connector on the main board. Then the results of sorting after LUT conversion will be stored in FIFO_D. The required procedures are the following:

1. Write any data into address 700018h (generate “Soft_Reset” for the FPGA).
2. Initialize the CCB2004 board in the Track Finder crate in “Discrete Logic” or “FPGA External” mode and generate a “Soft_Reset” for the CCB2004.
3. Write data = A2A1h into CSR0 (address 700158h).
4. Load the Rank+Phi LUT content for all four muons (see Section 3.1 in [1]):
   - use addresses 700400h..7007FEh for LUT corresponding to muon1
   - use addresses 700800h..7007BEh for LUT corresponding to muon2
   - use addresses 700C00h..7007FEh for LUT corresponding to muon3
   - use addresses 701000h..7013FEh for LUT corresponding to muon4
5. Read CSR1 and make sure the returned value is AAh (all FIFO buffers are empty).
6. Load 255 32-bit words (510 frames) of data into FIFO_A[1..12]. They will
   represent the 255 incoming muon patterns for the sorter unit.
7. Load the last word (2 frames) “0” into all FIFO_A[1..12] buffers.
8. Send the “Inject patterns from MS” broadcast command (=31h) from the TTC
   source (or write any data to address 70015Eh to emulate this command ).
9. Read CSR1 and make sure that FIFO_A is empty and the other FIFO’s are not
   empty (if the patterns in FIFO_A were representing valid muons).
10. Read 510 32-bit words of data from FIFO_C[1..4] and make sure they are equal
    to expected values according to FIFO_A content and the sorting algorithm.
11. Read 510 32-bit words of data from FIFO_B[1..4] and make sure they are equal
    to expected values according to FIFO_A content, the sorting algorithm and LUT
    content.
12. Read 510 32-bit words of data from FIFO_D and make sure they are equal to
    expected values according to FIFO_A content, the sorting algorithm, LUT
    content and required data format conversion (see Notes to Table 11 in [1]) for the
    selected connection between the front panel output and P12 connector.
13. Read CSR1 and make sure the returned value is AAh (all the FIFO buffers are
    empty).

### 3.4. MS2005 – to – GMT Data Transmission Test

The test involves the MS2005 board residing in the CSC Track Finder crate and the
Global Muon Trigger (GMT) receiver board residing in the GMT/GT crate [6]. The clock
and commands should arrive to both crates from the same TTC source. The simplest
option would be to establish a cable link connection (4 SCSI cables) between two boards,
run a MS2005 self-test as described above in Section 3.3 and check the patterns from
the input buffers of the GMT receiver against expected values. However, to be able to do
this, the GMT receiver must recognize the “Inject patterns from MS” broadcast command
and synchronize its operation with it. To simplify the MS2005 – to – GMT testing
procedures, we have implemented a dedicated RAM buffers in the main MS2005 FPGA.
These buffers keep patterns representing the four output muons being sent to the GMT
receiver (see Section 6 in [1]). The procedures to run a MS2005 – to – GMT test using
these RAM buffers are the following:

1. Write any data into address 700018h (generate “Soft_Reset” for the FPGA).
2. Initialize the CCB2004 board in the TF crate in “Discrete Logic” or “FPGA
   External” mode and generate a “Soft_Reset” for the CCB2004.
3. Initialize the GMT receiver board as needed.
4. Write data = \textbf{A4A1h} into \textbf{CSR0} address \textbf{700158h} (set CSR0[10]=1 to use the RAM buffers as data sources for the GMT receiver).
5. Write data = 0 into \textbf{CSR6}[7..0] to select the RAM buffer.
6. Write data = 0 into RAM address counter (address \textbf{700178h}).
7. Write 16-bit test pattern into address \textbf{70017Ch}.
8. Increment (up to \textbf{1FFh}) the address, load it into address counter and go to step 5 above.
9. Increment the number of selected RAM buffers (use values \textbf{CSR6} = 0, 1, 2, 4, 8, 16, 32, 64, 128) and go to step 4 above.
10. (Optional) Read back all \textbf{RAM buffers} for verification.
11. Send \textbf{BCntRes} broadcast command (=1) from the TTC source. Then all the 512 patterns from all RAM buffers (with the exception of bits [31..30]) will be sent out (without conversion) to GMT receiver. The input buffers on GMT receiver board are expected to be synchronized with this command.
12. Verify the content of input buffers on GMT receiver board against expected values.

3.5. \textbf{SP05 - to - MS2005 Data Transmission Test}

The simplest option is to test the connection between one SP05 board and the MS2005. The most complete test would involve all 12 SP05 boards residing in the TF crate. The data is sent from the Test FIFOs on the SP05 board (main FPGA) and checked from the FIFO C of the MS2005 and (optionally) three spy buffers on SP05 board: DAT_SF, DAT_SFE and DAT_SFM. The CCB2004 receives the clock and broadcast commands from the TTC source. Understanding of the SP05 functionality, external interfaces and internal registers is essential for this test, see [7] for details. This test also allows to measure the “safe window” (i.e. the fraction of the 80Mhz clock period within which the data from all 12 SP05 boards can be safely latched in into the M2005). The following procedures are required to run the test:

1. Program CCB2004 in the TF crate (slot 12):
   - write data = 1 into \textbf{CSRA1} (set “Discrete Logic” mode)
   - write any data into \textbf{CSRA3} (generate “\textbf{Soft_Reset}”)
2. Initialize the TTCrx from the TTC source as described in Section 1.1 above.
3. Program MS2005 in the TF crate (slot 14):
   - write any data into address \textbf{700018h} (generate “\textbf{Soft_Reset}” to the FPGA)
   - read CSR1 (FIFO status) and make sure the returned value is \textbf{AAh} (all FIFO buffers are empty)
   - write data = \textbf{12A1h} into \textbf{CSR0} (set “Test” mode temporary for the duration of the SP initialization procedures)
   - (Optional) write data = 0 into \textbf{CSR9} (enable inputs from all SP05 boards)
   - write data = \textbf{59h} into \textbf{CSR8} (set input clock for latching data from all SP05 boards in the middle of the “safe window”)
   - write any data to address \textbf{70016Ah} (set “winner bit” mode).
4. Program SP05 board(s) in the TF crate (slots 6..11 and 16..21):
   - write data = \textbf{1Fh} into \textbf{ACT_XFR FA MA} (reset all FIFO buffers, all
muons, all front FPGA’s)
- write data = 1Fh into ACT_XFR SP MA (reset all FIFO buffers, all muons)
- write data = 100h into CSR_FCC VM MA (set Fast Control Mode to VME)
- write data = 00C8h into ACT_FCC VM MA (reset bunch crossing counter with the command = 32h)
- Load Pt LUT content into DAT_PT (data = 0 for address = 0)
- (Optional) read back DAT_PT for verification
- (Optional) write data = FFFFh into ACT_ACR SP MA (reset various Local/Global Phi/Eta/DT LUT address counters)
- write data = 11FFh into CSR_SFC SP MA (spy FIFO window = 512 bunch crossings)
- write data = 39FFh into CSR_TFC SP MA (will inject test patterns for 512 bunch crossings from the main FPGA on the next FC_TFRUN command, enable injecting EMU test data from DAT_TFE, enable injecting timing bits BXN0/BC0 along with data patterns)
- (Optional) read back CSR_TFC SP MA for verification
- write data = A000h into CSR_SFC VM MA (persistent mode to spy on CCB_TPSP; spy FIFO starts writing data immediately after the requested event)
- write data = A000h into CSR_TFC VM MA (persistent mode to inject data on all events that follow; test FIFO starts injecting data immediately after the requested event).

5. Begin test iteration:
- load 512 test patterns (2 frames each) into DAT_TF SP M1
- load 512 test patterns (2 frames each) into DAT_TF SP M2
- load 512 test patterns (2 frames each) into DAT_TF SP M3
- load 512 test patterns (2 frames each) into DAT_TFE SP MA
- (Optional) read back CSR_TF SP and make sure the returned value is 8400h (Test FIFO is full)
- write data = 0 into CSR_FCC VM MA (set Fast Control Mode to CCB)
- write data = 12A0h into CSR0 of MS (set “Trigger” mode)
- write data = 2 into CSR_MWC SP MA (enable MS interface outputs, all SP’s).

6. Send “Start trigger” broadcast command (=6) from the TTC source.
7. Send BC0 broadcast command (=1) from the TTC source.
8. Send “Inject data from SP” broadcast command (=2Fh) from the TTC source.
9. Send “Stop trigger” broadcast command (=7) from the TTC source.
10. Write data = 100h into CSR_FCC VM MA of SP (set Fast Control Mode to VME).
11. Write data = 12A1h into CSR0 of MS (return to “Test” mode).
12. Write data = 8002h into CSR_MWC SP MA (disable MS output drivers, all SP’s).
13. (Optional) Read CSR1 from the MS2005 and verify that the FIFO_C is not empty (if the SP’s were expecting to send valid muon patterns)
14. (Optional) Read \textbf{CSR\_TF SP} and make sure the returned value is \textbf{4000h} (test FIFO is empty).

15. (Optional) Read \textbf{CSR\_SFM SP M1/2/3} and make sure the MS spy FIFO is not empty (if the SP05 was sending valid patterns) or full (returned value = \textbf{8400h}, if all 512 patterns were valid).

16. Read 512 words (two frames each) from \textbf{DAT\_SF M[1..3]} and verify them against the data loaded into test FIFO \textbf{DAT\_TF} for every muon.

17. Read 512 words from \textbf{FIFO\_C[1..4]} and compare with expected values, taking into account the number of SP05 sources in the TF crate, Pt LUT content, sorting algorithm. Note the BC0 and BXN0 bits are treated separately (see description of \textbf{DAT\_TF} in the SP05 manual).

18. Read 512 words (two frames each) from \textbf{DAT\_SFE SP MA} and verify the EMU spy FIFO content against the data preloaded into EMU test FIFO \textbf{DAT\_TFE SP MA}.

19. Read 512 words (two frames each) from \textbf{DAT\_SFM M[1..3]} and make sure the MS\_ID[3..1] bits in the first frame correspond to expected results of sorting. Verify the BXN0 and BC0 bits according to content of the source FIFO \textbf{DAT\_TFE}.

Note: you will probably need to adjust the CSR0[15:12] bits (MS2005; delay of winner bits on MS board before they are sent to SP).

20. Go to step 5, load another set of test patterns into \textbf{DAT\_TF} and \textbf{DAT\_TFE} and repeat the iteration.

To measure the “safe window”, the value loaded into the CSR8 on Step 3, should vary typically between 44h and 70h (use 44h, 45h... 70h); each step corresponds to 0.25 ns. For each step we recommend to run 300..500 iterations (up to 1000 iterations on the boundaries of the “safe window’’); each iteration comprises 512 random data patterns to be loaded into every SP05. The “safe window” corresponds to error-free data transmission. Based on our experience, the average “safe window” is ~6.5 ns (CSR8=4Ch..66h). It is essential that all the twelve SP05 boards participate in this measurement.

4. Data Transmission Chain Tests

The chain tests involve the following boards:
- CCB2004, MPC2004, up to 9 TMB2005 residing in the peripheral crate
- CCB2004, MS2005 and up to 12 SP05 in the Track Finder crate
- TTC source boards (for example, TTCvi/TTCvx, or TTCci) residing in the TF or separate VME crate
- (Optional) GMT receiver board residing in the GMT/GT crate

The most typical hardware configuration consists of two crates: peripheral with the CCB2004, MPC2004 and nine TMB2005 boards, and the Track Finder with the CCB2004, MS2005 and one SP05 board (Fig.2). The TTCvi and TTCvx modules are located in the TF crate. Three optical fibers connect the outputs of the MPC2004 with one triple muon input (usually, F1, the lowest part) of the SP05. The chain test may be divided into three stages:
(1) from the TMB2005 through MPC2004 to the input FIFO on SP05;
(2) from the input of the SP05 to output of the SP05
(3) from the output of SP05 to the output of MS2005 (input of GMT).

Figure 2: Chain Test in the Peripheral and Track Finder crates

At a first stage of the chain test, the testing patterns are loaded into all TMB2005 boards, sent to MPC2004 and verified from the output FIFO_B buffer of MPC2004 and from input spy FIFO buffers DAT_SFM FA on SP05. The steps required to perform this test are the following (see Sections 2.4 and 2.6 for more details):

1. Program and initialize the CCB2004 boards in both crates in “Discrete Logic” or “FPGA External” mode as described in Section 1.1.
2. Program and initialize the SP05 as described in Step 4, Section 2.6. Note the delay values SFD[9:0] in CSR_SFC VM and AFD[6:0] in CSR_AFD FA MA may be different from those listed in Section 2.7. Also note the bits SFM and SFRM in CSR_SFC VM should be “1” and SFRM=0 to allow data to be captured on the next CCB_TPTMB broadcast command.
3. Program and initialize the MPC2004 board (Step 3, Section 2.5).
4. Program and initialize all TMB2005 board(s) (Steps 4-7, Section 2.5; load 255 pairs of LCT0+LCT1 into MPC RAM).
5. Send L1Reset broadcast command (=3h) from the TTC source to both CCB2004 boards for timing adjustments on SP05 inputs.
6. Send “Inject patterns from TMB” command (=24h) from the TTC source to both CCB2004 boards.
7. Read 255 words (510 frames) from FIFO_B[3..1] on MPC2004 and make sure they satisfy with the sorting criteria.
8. Read 255 words (510 frames) from DAT_SF F1 M1/M2/M3 (for each muon) and make sure they satisfy with the sorting criteria and are equal to ones read out from FIFO_B[3..1] on MPC2004.
9. Read CSR_SF spy FIFO status and verify that the spy FIFO is empty (SFEF=1).
10. Read **CSR_BID** and verify that the **MPC_LINK_ID[7:0]** is equal to expected number (**MPC_ID[5:0]** as pre-programmed into the CSR0 of MPC2004 and **LINK_ID[1:0]** as hard-wired on SP05).

11. Read “**winner bits**” from participating TMB2005 boards as described in Step 12, Section 2.5.

The second stage involves the SP05 only. The testing of SP05 internal functionality is out of scope of this Guide. The third stage is described in Section 3.5. Some examples of the **C++** programs to run the first and third stages of the chain test can be found in [1].

**References**


**History**

**12/06/2006.** Section 2.3 was added.
**01/10/2007.** Major additions to Sections 2 and 3.
**01/19/2007.** Minor additions to Section 3.5. Sections 1.2 and 1.4 were expanded, Sections 1.5 and 4 added.
**02/06/2007.** Appendix was added.
**02/27/2007.** Minor changes in Section 1.1. Section 1.6 was added.
Appendix

1. Peripheral Backplane

The picture of the EMU custom peripheral 9U backplane designed at the University of Florida, Gainesville (revision 3, production version, front view) is shown on Fig.1. The upper part complies with the VME64x protocol that uses a unique geographical address for every slot 1..21. The base 24-bit addresses according to geographical addressing scheme board are listed in Table 1 for every slot. All the other connectors are female type metric (2 mm) 5- or 7-row Zpack (Example is shown on Fig.2). Pin assignment of the CCB, MPC, TMB and DMB backplane connectors is given in Tables 2-5 respectively.

Figure 1: EMU Custom Peripheral Backplane, front view
Table 1: Geographical Addressing in the EMU Peripheral Crate

<table>
<thead>
<tr>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CC</td>
<td>8</td>
<td>TMB4</td>
<td>15</td>
<td>DMB6</td>
<td>780000h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TMB1</td>
<td>100000h</td>
<td>9</td>
<td>DMB4</td>
<td>480000h</td>
<td>16</td>
<td>TMB7</td>
<td>800000h</td>
</tr>
<tr>
<td>3</td>
<td>DMB1</td>
<td>180000h</td>
<td>10</td>
<td>TMB5</td>
<td>500000h</td>
<td>17</td>
<td>DMB7</td>
<td>880000h</td>
</tr>
<tr>
<td>4</td>
<td>TMB2</td>
<td>200000h</td>
<td>11</td>
<td>DMB5</td>
<td>580000h</td>
<td>18</td>
<td>TMB8</td>
<td>900000h</td>
</tr>
<tr>
<td>5</td>
<td>DMB2</td>
<td>280000h</td>
<td>12</td>
<td>MPC</td>
<td>600000h</td>
<td>19</td>
<td>DMB8</td>
<td>980000h</td>
</tr>
<tr>
<td>6</td>
<td>TMB3</td>
<td>300000h</td>
<td>13</td>
<td>CCB</td>
<td>680000h</td>
<td>20</td>
<td>TMB9</td>
<td>A000000h</td>
</tr>
<tr>
<td>7</td>
<td>DMB3</td>
<td>380000h</td>
<td>14</td>
<td>TMB6</td>
<td>700000h</td>
<td>21</td>
<td>DMB9</td>
<td>A800000h</td>
</tr>
</tbody>
</table>

Power distribution in the EMU peripheral backplane is different from standard VME backplane. Only +3.3V (as specified in the VME64x document for pins D12, D14, D16, D18, D20, D22, D24, D26, D28, D30) and +5.0V (pins A32, B32, C32) are provided. +12V and –12V powers are not provided. The power comes from the Crate Regulator Board (CRB) located behind the J1/P1 portion of the custom backplane. The CRB provides +3.3V and +5.0V powers individually to each slot in the crate. In addition, the CRB provides two sources of +1.5V required for the GTLP terminators: one for the MPC and another for all other boards (CCB, TMB1-9, DMB1-9). The +3.3V power for RAT1-9 cards comes from the corresponding TMB board (Table 4, connector 4). The CRB output voltages can be monitored over CAN bus. Some of the voltages can be monitored on a TMB2005 board accessing the ADR_ADC register. An example of the program is given in [2].

---

Figure 2: Pin assignment of the Zpack77 connector
All bussed GTLP signals (full list is given in Table 3A of the CCB Specification [1]) are terminated on both ends of the peripheral backplane. All 40MHz and 80MHz point-to-point signals are terminated on receiving boards: 100 Ohm to +1.5V for GTLP, 100 Ohm between the complementary signals for LVDS (40MHz and 80MHz clocks from the CCB).
Table 3: MPC Slot
Table 4: TMB Slot

<table>
<thead>
<tr>
<th>XIA</th>
<th>XIB</th>
<th>XID</th>
<th>XIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>C1</td>
<td>D1</td>
</tr>
<tr>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
</tr>
<tr>
<td>A3</td>
<td>B3</td>
<td>C3</td>
<td>D3</td>
</tr>
<tr>
<td>A4</td>
<td>B4</td>
<td>C4</td>
<td>D4</td>
</tr>
<tr>
<td>A5</td>
<td>B5</td>
<td>C5</td>
<td>D5</td>
</tr>
<tr>
<td>A6</td>
<td>B6</td>
<td>C6</td>
<td>D6</td>
</tr>
<tr>
<td>A7</td>
<td>B7</td>
<td>C7</td>
<td>D7</td>
</tr>
<tr>
<td>A8</td>
<td>B8</td>
<td>C8</td>
<td>D8</td>
</tr>
<tr>
<td>A9</td>
<td>B9</td>
<td>C9</td>
<td>D9</td>
</tr>
<tr>
<td>A10</td>
<td>B10</td>
<td>C10</td>
<td>D10</td>
</tr>
<tr>
<td>A11</td>
<td>B11</td>
<td>C11</td>
<td>D11</td>
</tr>
<tr>
<td>A12</td>
<td>B12</td>
<td>C12</td>
<td>D12</td>
</tr>
<tr>
<td>A13</td>
<td>B13</td>
<td>C13</td>
<td>D13</td>
</tr>
<tr>
<td>A14</td>
<td>B14</td>
<td>C14</td>
<td>D14</td>
</tr>
<tr>
<td>A15</td>
<td>B15</td>
<td>C15</td>
<td>D15</td>
</tr>
<tr>
<td>A16</td>
<td>B16</td>
<td>C16</td>
<td>D16</td>
</tr>
<tr>
<td>A17</td>
<td>B17</td>
<td>C17</td>
<td>D17</td>
</tr>
<tr>
<td>A18</td>
<td>B18</td>
<td>C18</td>
<td>D18</td>
</tr>
<tr>
<td>A19</td>
<td>B19</td>
<td>C19</td>
<td>D19</td>
</tr>
<tr>
<td>A20</td>
<td>B20</td>
<td>C20</td>
<td>D20</td>
</tr>
<tr>
<td>A21</td>
<td>B21</td>
<td>C21</td>
<td>D21</td>
</tr>
<tr>
<td>A22</td>
<td>B22</td>
<td>C22</td>
<td>D22</td>
</tr>
<tr>
<td>A23</td>
<td>B23</td>
<td>C23</td>
<td>D23</td>
</tr>
<tr>
<td>A24</td>
<td>B24</td>
<td>C24</td>
<td>D24</td>
</tr>
<tr>
<td>A25</td>
<td>B25</td>
<td>C25</td>
<td>D25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZPA154</th>
<th>ZPA154</th>
<th>ZPA154</th>
<th>ZPA154</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>A1</td>
<td>Z15</td>
<td>B15</td>
</tr>
<tr>
<td>Z2</td>
<td>A2</td>
<td>Z16</td>
<td>B16</td>
</tr>
<tr>
<td>Z3</td>
<td>A3</td>
<td>Z17</td>
<td>B17</td>
</tr>
<tr>
<td>Z4</td>
<td>A4</td>
<td>Z18</td>
<td>B18</td>
</tr>
<tr>
<td>Z5</td>
<td>A5</td>
<td>Z19</td>
<td>B19</td>
</tr>
<tr>
<td>Z6</td>
<td>A6</td>
<td>Z20</td>
<td>B20</td>
</tr>
<tr>
<td>Z7</td>
<td>A7</td>
<td>Z21</td>
<td>B21</td>
</tr>
<tr>
<td>Z8</td>
<td>A8</td>
<td>Z22</td>
<td>B22</td>
</tr>
<tr>
<td>Z9</td>
<td>A9</td>
<td>Z23</td>
<td>B23</td>
</tr>
<tr>
<td>Z10</td>
<td>A10</td>
<td>Z24</td>
<td>B24</td>
</tr>
<tr>
<td>Z11</td>
<td>A11</td>
<td>Z25</td>
<td>B25</td>
</tr>
<tr>
<td>Z12</td>
<td>A12</td>
<td>Z26</td>
<td>B26</td>
</tr>
<tr>
<td>Z13</td>
<td>A13</td>
<td>Z27</td>
<td>B27</td>
</tr>
<tr>
<td>Z14</td>
<td>A14</td>
<td>Z28</td>
<td>B28</td>
</tr>
<tr>
<td>Z15</td>
<td>A15</td>
<td>Z29</td>
<td>B29</td>
</tr>
<tr>
<td>Z16</td>
<td>A16</td>
<td>Z30</td>
<td>B30</td>
</tr>
<tr>
<td>Z17</td>
<td>A17</td>
<td>Z31</td>
<td>B31</td>
</tr>
</tbody>
</table>

3.3V

Table continues with additional columns and rows representing different slots and connections, with various labels and connections denoted by numbers and letters.
2. Track Finder Backplane

The picture of the CSC Track Finder custom 6U backplane designed at the University of Florida, Gainesville (revision 2, production version, front view) is shown on Fig.3. This backplane is located in the Track Finder (TF) crate below commercial 3U VME64x backplane which utilizes the geographical addressing scheme as described in Table 6. The TF crate is a customized Wiener 6023 [3] part with the power supply that provides the +5.0V, +3.3V and (optionally) +12V and –12V. The Wiener power supply is connected to 3U VME64x backplane with several wires as specified by Wiener. Three thick wires for the GND, +5.0V and +3.3V are required for connection between the VME64x and custom 6U backplanes. An additional custom mezzanine card (Fig.4) available from the University of Florida should be mounted on custom backplane to provide +1.5V for the GTLP terminators. Pin assignment of all female type metric (2 mm) 5- or 7-row Zpack connectors for the CCB, MS, SP, DDU and MPC slots is shown in Tables 7-11 respectively. Note the MPC slots are implemented only partially (compare with Table 2). The MPC may only receive the CCB 40MHz and 80MHz clocks and commands. The main goal of putting the MPC into the TF crate is to provide a source(s) of data for the SP boards for testing purposes.

Table 6: Geographical Addressing in the TF Crate

<table>
<thead>
<tr>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
<th>VME Slot</th>
<th>Board</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CC</td>
<td>100000h</td>
<td>8 SP3 400000h</td>
<td>9 SP4 480000h</td>
<td>15 None 780000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 DDU 180000h</td>
<td>10 SP5 500000h</td>
<td>16 SP7 800000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 MPC1 200000h</td>
<td>11 SP6 580000h</td>
<td>17 SP8 880000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 MPC2 280000h</td>
<td>12 CCB 600000h</td>
<td>19 SP10 980000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 MPC3 300000h</td>
<td>13 None 680000h</td>
<td>20 SP11 A00000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 SP1 380000h</td>
<td>14 MS 700000h</td>
<td>21 SP12 A80000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All bussed GTLP signals (full list is given in Table 3B of the CCB Specification [1]) are terminated on both ends of the TF custom backplane. All 40MHz and 80MHz point-to-point signals are terminated on receiving boards: 100 Ohm to +1.5V for GTLP, 100 Ohm between the complementary signals for LVDS (40MHz and 80MHz clocks from the CCB).
Figure 3: Track Finder custom 6U backplane, front view

Figure 4: Track Finder crate, rear view
### Table 7: CCB Slot

<table>
<thead>
<tr>
<th>X1A</th>
<th>X1B</th>
<th>X1C</th>
<th>X1D</th>
<th>X1E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>C1</td>
<td>D1</td>
<td>E1</td>
</tr>
<tr>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
<td>E2</td>
</tr>
<tr>
<td>A3</td>
<td>B3</td>
<td>C3</td>
<td>D3</td>
<td>E3</td>
</tr>
<tr>
<td>A4</td>
<td>B4</td>
<td>C4</td>
<td>D4</td>
<td>E4</td>
</tr>
<tr>
<td>A5</td>
<td>B5</td>
<td>C5</td>
<td>D5</td>
<td>E5</td>
</tr>
<tr>
<td>A6</td>
<td>B6</td>
<td>C6</td>
<td>D6</td>
<td>E6</td>
</tr>
<tr>
<td>A7</td>
<td>B7</td>
<td>C7</td>
<td>D7</td>
<td>E7</td>
</tr>
<tr>
<td>A8</td>
<td>B8</td>
<td>C8</td>
<td>D8</td>
<td>E8</td>
</tr>
<tr>
<td>A9</td>
<td>B9</td>
<td>C9</td>
<td>D9</td>
<td>E9</td>
</tr>
<tr>
<td>A10</td>
<td>B10</td>
<td>C10</td>
<td>D10</td>
<td>E10</td>
</tr>
<tr>
<td>A11</td>
<td>B11</td>
<td>C11</td>
<td>D11</td>
<td>E11</td>
</tr>
<tr>
<td>A12</td>
<td>B12</td>
<td>C12</td>
<td>D12</td>
<td>E12</td>
</tr>
<tr>
<td>A13</td>
<td>B13</td>
<td>C13</td>
<td>D13</td>
<td>E13</td>
</tr>
<tr>
<td>A14</td>
<td>B14</td>
<td>C14</td>
<td>D14</td>
<td>E14</td>
</tr>
<tr>
<td>A15</td>
<td>B15</td>
<td>C15</td>
<td>D15</td>
<td>E15</td>
</tr>
<tr>
<td>A16</td>
<td>B16</td>
<td>C16</td>
<td>D16</td>
<td>E16</td>
</tr>
<tr>
<td>A17</td>
<td>B17</td>
<td>C17</td>
<td>D17</td>
<td>E17</td>
</tr>
<tr>
<td>A18</td>
<td>B18</td>
<td>C18</td>
<td>D18</td>
<td>E18</td>
</tr>
<tr>
<td>A19</td>
<td>B19</td>
<td>C19</td>
<td>D19</td>
<td>E19</td>
</tr>
<tr>
<td>A20</td>
<td>B20</td>
<td>C20</td>
<td>D20</td>
<td>E20</td>
</tr>
<tr>
<td>A21</td>
<td>B21</td>
<td>C21</td>
<td>D21</td>
<td>E21</td>
</tr>
<tr>
<td>A22</td>
<td>B22</td>
<td>C22</td>
<td>D22</td>
<td>E22</td>
</tr>
<tr>
<td>A23</td>
<td>B23</td>
<td>C23</td>
<td>D23</td>
<td>E23</td>
</tr>
<tr>
<td>A24</td>
<td>B24</td>
<td>C24</td>
<td>D24</td>
<td>E24</td>
</tr>
</tbody>
</table>

### Diagram

![Diagram of CCB Slot]
Table 8: MS Slot

<table>
<thead>
<tr>
<th>X1A</th>
<th>X1B</th>
<th>X1C</th>
<th>X1D</th>
<th>X1E</th>
<th>X1F</th>
<th>X1G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
</tbody>
</table>

**Table Data:**

- **X1A:** Various signals or components
- **X1B:** Various signals or components
- **X1C:** Various signals or components
- **X1D:** Various signals or components
- **X1E:** Various signals or components
- **X1F:** Various signals or components
- **X1G:** Various signals or components
Table 9: SP Slot
Table 10: DDU Slot

Table 11: MPC Slot
3. VME Backplanes

<table>
<thead>
<tr>
<th>Pin</th>
<th>Row Z (2)</th>
<th>Row A</th>
<th>Row B</th>
<th>Row C</th>
<th>Row D (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPR</td>
<td>D00</td>
<td>BBSY*</td>
<td>D08</td>
<td>VPC (3)</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>D01</td>
<td>BCLR*</td>
<td>D09</td>
<td>GND (3)</td>
</tr>
<tr>
<td>3</td>
<td>MCLK</td>
<td>D02</td>
<td>ACFAIL*</td>
<td>D10</td>
<td>+V1</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>D03</td>
<td>BG0IN*</td>
<td>D11</td>
<td>+V2</td>
</tr>
<tr>
<td>5</td>
<td>MSD</td>
<td>D04</td>
<td>BG0OUT*</td>
<td>D12</td>
<td>RsvU</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>D05</td>
<td>BG1IN*</td>
<td>D13</td>
<td>-V1</td>
</tr>
<tr>
<td>7</td>
<td>MMD</td>
<td>D06</td>
<td>BG1OUT*</td>
<td>D14</td>
<td>-V2</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>D07</td>
<td>BG2IN*</td>
<td>D15</td>
<td>RsvU</td>
</tr>
<tr>
<td>9</td>
<td>MCTL</td>
<td>GND</td>
<td>BG2OUT*</td>
<td>GND</td>
<td>GAP*</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>SYSCLK</td>
<td>BG3IN*</td>
<td>SYSFAIL*</td>
<td>GA0*</td>
</tr>
<tr>
<td>11</td>
<td>RESP*</td>
<td>GND</td>
<td>BG3OUT*</td>
<td>BERR*</td>
<td>GA1*</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>DS1*</td>
<td>BR0*</td>
<td>SYSRESET*</td>
<td>+3.3V</td>
</tr>
<tr>
<td>13</td>
<td>RsvBus</td>
<td>DS0*</td>
<td>BR1*</td>
<td>LWORD*</td>
<td>GA2*</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>WRITE*</td>
<td>BR2*</td>
<td>AM5</td>
<td>+3.3V</td>
</tr>
<tr>
<td>15</td>
<td>RsvBus</td>
<td>GND</td>
<td>BR3*</td>
<td>A23</td>
<td>GA3*</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>DTACK*</td>
<td>AM0</td>
<td>A22</td>
<td>+3.3V</td>
</tr>
<tr>
<td>17</td>
<td>RsvBus</td>
<td>GND</td>
<td>AM1</td>
<td>A21</td>
<td>GA4*</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>AS*</td>
<td>AM2</td>
<td>A20</td>
<td>+3.3V</td>
</tr>
<tr>
<td>19</td>
<td>RsvBus</td>
<td>GND</td>
<td>AM3</td>
<td>A19</td>
<td>RsvBus</td>
</tr>
<tr>
<td>20</td>
<td>GND</td>
<td>IACK*</td>
<td>GND</td>
<td>A18</td>
<td>+3.3V</td>
</tr>
<tr>
<td>21</td>
<td>RsvBus</td>
<td>IACKIN*</td>
<td>SERA (1)</td>
<td>A17</td>
<td>RsvBus</td>
</tr>
<tr>
<td>22</td>
<td>GND</td>
<td>IACKOUT*</td>
<td>SERB (1)</td>
<td>A16</td>
<td>+3.3V</td>
</tr>
<tr>
<td>23</td>
<td>RsvBus</td>
<td>AM4</td>
<td>GND</td>
<td>A15</td>
<td>RsvBus</td>
</tr>
<tr>
<td>24</td>
<td>GND</td>
<td>A07</td>
<td>IRQ7*</td>
<td>A14</td>
<td>+3.3V</td>
</tr>
<tr>
<td>25</td>
<td>RsvBus</td>
<td>A06</td>
<td>IRQ6*</td>
<td>A13</td>
<td>RsvBus</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
<td>A05</td>
<td>IRQ5*</td>
<td>A12</td>
<td>+3.3V</td>
</tr>
<tr>
<td>27</td>
<td>RsvBus</td>
<td>A04</td>
<td>IRQ4*</td>
<td>A11</td>
<td>LI/I*</td>
</tr>
<tr>
<td>28</td>
<td>GND</td>
<td>A03</td>
<td>IRQ3*</td>
<td>A10</td>
<td>+3.3V</td>
</tr>
<tr>
<td>29</td>
<td>RsvBus</td>
<td>A02</td>
<td>IRQ2*</td>
<td>A09</td>
<td>LI/O*</td>
</tr>
<tr>
<td>30</td>
<td>GND</td>
<td>A01</td>
<td>IRQ1*</td>
<td>A08</td>
<td>+3.3V</td>
</tr>
<tr>
<td>31</td>
<td>RsvBus</td>
<td>-12 VDC</td>
<td>+5VSTDBY</td>
<td>+12 VDC</td>
<td>GND (3)</td>
</tr>
<tr>
<td>32</td>
<td>GND</td>
<td>+5 VDC</td>
<td>+5 VDC</td>
<td>+5 VDC</td>
<td>VPC (3)</td>
</tr>
</tbody>
</table>

(1) Pin(s) redefined under the VME64 specification.
(2) Pin(s) redefined under the VME64x specification.
(3) Elongated (mate first, break last) connector contact.
### P2/J2 Pin Assignment

<table>
<thead>
<tr>
<th>Pin</th>
<th>Row Z (2)</th>
<th>Row A</th>
<th>Row B</th>
<th>Row C</th>
<th>Row D (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>+5 VDC</td>
<td>UsrDef</td>
<td>UsrDef (3)</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>UsrDef</td>
<td>GND</td>
<td>UsrDef</td>
<td>UsrDef (3)</td>
</tr>
<tr>
<td>3</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>RETRY* (1)</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>UsrDef</td>
<td>A24</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>5</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>A25</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>UsrDef</td>
<td>A26</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>7</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>A27</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>UsrDef</td>
<td>A28</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>9</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>A29</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>UsrDef</td>
<td>A30</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>11</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>A31</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>UsrDef</td>
<td>GND</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>13</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>+5 VDC</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>14</td>
<td>GND</td>
<td>UsrDef</td>
<td>D16</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>15</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D17</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>UsrDef</td>
<td>D18</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>17</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D19</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>UsrDef</td>
<td>D20</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>19</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D21</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>20</td>
<td>GND</td>
<td>UsrDef</td>
<td>D22</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>21</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D23</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>22</td>
<td>GND</td>
<td>UsrDef</td>
<td>GND</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>23</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D24</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>24</td>
<td>GND</td>
<td>UsrDef</td>
<td>D25</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>25</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D26</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
<td>UsrDef</td>
<td>D27</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>27</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D28</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>28</td>
<td>GND</td>
<td>UsrDef</td>
<td>D29</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>29</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>D30</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>30</td>
<td>GND</td>
<td>UsrDef</td>
<td>D31</td>
<td>UsrDef</td>
<td>UsrDef</td>
</tr>
<tr>
<td>31</td>
<td>UsrDef</td>
<td>UsrDef</td>
<td>GND</td>
<td>UsrDef</td>
<td>GND (3)</td>
</tr>
<tr>
<td>32</td>
<td>GND</td>
<td>UsrDef</td>
<td>+5 VDC</td>
<td>UsrDef</td>
<td>VPC (3)</td>
</tr>
</tbody>
</table>

(1) Pin(s) redefined under the VME64 specification.
(2) Pin(s) redefined under the VME64x specification.
(3) Elongated (mate first, break last) connector contact.

### References

[2] [http://bonner-ntserver.rice.edu/cms/tlc2543.txt](http://bonner-ntserver.rice.edu/cms/tlc2543.txt)