Jason Leake and Bob Hatley.
Version 1.8, 8th July 1996

Programming model for the Trigger Timing Control Module used in the Phase 2 Atlas Bit-Parallel Calorimeter Trigger Demonstrator

The Timing Control Module (TCM) generates and distributes the following signals in the Phase 2 demonstrator:

- *Clock* signals for the FADC, Transmitter and Cluster Processing modules, the Level 2 trigger demonstrator and general test outputs.
- A *Dead Time* signal.
- An *Electron* signal, indicating a Cluster Processing Module hit.
- *Stop* signals for the Transmitter and Cluster Processing modules.
- A *Digitisation Gate* for the Heidelberg FADC modules.

  TDC START and *TDC STOP* signals for an extertnal TDC

The module is a double width 6U VME module.
Provision has been made for a TDC, measuring the time interval between a beam trigger and the next clock edge, to be added later via a daughter board if required.
The daughter board connection is via a 40-way header. The signals available via this header include the 16 bit VME data bus and bits A5—A1[1] of the VME address bus. If the TCM is selected, the daughter board logic will respond when bit 5 is 1, and the mother board when bit 5 is zero. Thus, the address range 20—3F is reserved for use by the daughter board.

1. The notation used in this document

The notation adopted in this document is for registers and bit fields to be written in *italic*. The names of signals are written in *underlined italic*.

Setting a bit means writing a 1 to it, and clearing it means writing a 0. A byte is 8 bits wide, and a word is 16 bits wide. The least significant bit is labelled bit 0.

Suggested abbreviated names for registers and bit fields are given in section headings, where they are enclosed in square brackets—[]. These names can be used in the software and thus the descriptions of them can be easily found in this document. Of course, the names may be written in mixed or lower case in the software.

2. General considerations in the design of the programming model

The general guidelines for the programming model are listed below.

- The usual layout adopted by the collaboration, with the lowest numbered register being a unique module identification register and followed by the main module control and status registers.
- All registers are readable. Bits that are writeable by the computer cannot be modified by the internal circuitry on the card and vice versa.

[1] VME has no A0 because it addresses by words. For byte accesses, which are not implemented on this card, additional logic indicates which byte is to be transferred.

- Writeable bit fields return the last value written except in the case of those bits which produce pulsed signals by writing a 1 to them (such as *Reset* in the *Control Register* which resets the module to is initial power-up state when written to). Such bits always return 0 when read. This is so that there is no effect if a register is read and then written back with the value just read.

- Unused bits in those registers which the computer can modify (i.e. RW registers) are always 0 on read.

- Unused bits in registers which are read-only by the computer (i.e. RO registers) are read back by the computer as zeros.

- No field of any register can be modified by both the computer and the module, except to be reset to an initial value by the module on power-up or on receipt of a *Reset* from the computer. The initial value of all registers is 0000.

- The module supports non-block mode A24/D16 operations only (i.e. the J2 connector is not used). All registers are 16 bits wide and must be accessed as such—byte accesses will cause an undefined response from the card. Similarly, the module will only respond to A24 addressing, not A16 or A32. The reason for these restrictions is to eliminate unnecessary circuitry as these functions are not required.

3. Register Layout

The 15 VME registers are laid out in the following order. One word in the address space is unused and there are a further 16 words which are reserved for the daughter board. The computer should not access any of these as these as the card will not assert *DTACK**, and the eventual result will be a SIGBUS software signal.

Offset from module base address	Name	Abbreviation	Access	Function
0	Module ID register	MID	RO	Uniquely identifies module
2	Control register	CR	RW	Controls operation of module
4	Status register	SR	RO	Communicates module status
6	Interrupt register		RO	Bit mask indicating reason for VME interrupt generated by module
8	Unused			
A	TXM clock 1 fine delay	TXM_CLK1_FINE	RW	Fine delay to be added to TXM clock 1
C	CPM clock 1 fine delay	CPM_CLK1_FINE	RW	Fine delay to be added to *CPM Clock 1*
E	TXM clock 2 fine delay	TXM_CLK2_FINE	RW	Fine delay to be added to *TXM Clock 2*
10	TRIG IP tick delay	TRIG_IP_TICK	RW	Tick delay to be added to *Trigger Input*
12	TRIG IP fine delay	TRIG_IP_FINE	RW	Fine delay to be added to *Trigger Input*
14	FADC Gate tick delay	DIG_GATE_TICK	RW	Tick delay to be added to *Data Gate* signal sent to the Heidelberg FADCs
16	FADC Gate fine delay	DIG_GATE_FINE	RW	Fine delay to be added to *Data Gate* signal sent to the Heidelberg FADCs
18	TXM Stop tick delay	TXM_S_TICK	RW	Tick delay added to the *Stop* signal sent to the TXM
1A	TXM Stop fine delay	TXM_S_FINE	RW	Fine delay added to the *Stop* signal sent to the TXM
1C	CPM Stop tick delay	CPM_S_TICK	RW	Tick delay added to the *Stop* signal sent to the CPM
1E	CPM Stop fine delay	CPM_S_FINE	RW	Fine delay added to the *Stop* signal sent to the CPM
20—3F	Reserved for TDC daughter board		None	

The meanings of abbreviations in the table are as follows:

RW	Register which can be both read and modified (i.e. written) by the computer

RO	Register which are read only—i.e. they can be read but not modified by the computer. Attempting to write to such a register from the computer has no effect upon its contents on the module's state
CPM	Cluster Processing Module
FADC	Heidelberg Flash ADC cards
TRIG IP	Trigger Input signal
TXM	Transmitter Module

The module only responds to non-block mode A24/D16 accesses, with address modifiers 39 and 3D. The upper 16 bits of the card address are set using DIP switches. Because the card will respond only when the next two bits of the address are zero, it be allocated any 32 word address space which starts on a 128 word boundary.

4. Module ID Register [MID]

This word contains a hardwired bit pattern that represents the type, revision and serial number of the card. It is a modification of Eric Eisenhandler's scheme, to allow a daughter board serial number to be incorporated.

Bits	**Function**	**Abbreviation**	**Notes**
0-3	*Serial Number* Serial number of module	SERIAL	First module is 1
4-7	*Daughter Board ID* Serial number of daughter board	DID	First daughter board is 1. If no daughter board, then 0.
8-10	*Revision Number* Revision number of module	REV	Initial revision is 1
11-15	*Type Number* Type number of module	TYPE	Value for this module is always 4

Because it is not clear whether a daughter board will ever be produced, it was not felt worthwhile to have a special daughter board ID register. As very few examples of the TCM will be built, there is sufficient room in the normal "serial number" field used by the collaboration to also hold a daughter board serial number.

If there is no daughter board fitted to the module, then the *Daughter Board ID* field contains zero. Otherwise, it contains the serial number of the daughter board. The first daughter board built has serial number 1. To minimise the number of connections to the daughter board, there are no type or revision number fields.

The daughter board serial number is physically derived using four signals in the header for it on the TCM. When unconnected, these float to the 0 state to provide a serial number of zero if no board is present. Thus, this scheme has capacity for up to 15 different daughter boards.

5. Control Register [CR]

This register is used by the computer to configure the card. It can be modified by the computer but not by the module except when the computer requests a reset to the module's power-up state.

Most bit fields (including unused bits) are read back by the computer as they were last written by it. The initial power-up condition is for it to contain 0000. This is important as in this state the generation of clock ticks (i.e. pulses) to other modules is suppressed, as is the generation of VME interrupts.

Bits	**Function**	**Abbreviation**	**Notes**
0 (LSB)	Reset	RESET	Always read as 0
1-2	Clock Signal Source	CLK_SSRCE	
3	Clock Speed Select	CLK_SPEED	
4-6	Trigger Select	TRIG_SEL	
7	Set Live	SET_LIVE	Always read as 0
8	Set Dead	SET_DEAD	Always read as 0
9	Set Digitisation	SET_DIGIT	Always read as 0
10	Set Hold	SET_HOLD	Always read as 0
11	Enable Interrupts	EN_INT	
12-15	Unused		Always 0 on read

Reset [RESET] Bit 0

Setting this bit causes the card to revert to its power-up state. This takes less than one VME cycle to complete, so that the computer does not have to wait or poll the card to see if the operation is completed. If any other bits in the *Control Register* are set at the same time as this bit, they are cleared to zero again by the reset (although the effect during the reset is undefined, for example the might emit some clock pulses if a signal source was enabled).

This means that good practice dictates that the computer should not set other bits at the same time as this bit. For this reason it is allocated to bit 0, so that the computer writes 0001 to the register to reset the card rather than some slightly less obvious pattern.

0	Read value. Writing 0 has no effect
1	On writing, causes the module to instantly (as far as the computer is concerned) reset to its power-up state

Clock Signal Source [CLK_SSRCE] Bits 1-2

This bit selects the clock source as listed below. The power-up state of 0 ensures that the module does not output any clock pulses until this function is enabled. The "normal" state is for both bits to be set so that the clock source is the internal 80 MHz oscillator.

Bit 2	**Bit 1**	**Effect**
0	0	No clock pulses output
0	1	Clock source is External Lemo socket
1	0	Clock source is External Oscillator Module
1	1	Clock source is internal 80 MHz oscillator

NB, System Clock frequency is half the Clock source frequency.

Clock 2 Speed Select [CLK_SPEED] Bit 3

This bit selects whether or not the Clock 2 speed is to be halved. If the internal oscillator is selected because the *Clock Signal Source* field is 11_2, then it selects between 40 and 80 MHz (because the internal oscillator runs at 80 MHz).

0	Clk2 = 80 MHz, same as source
1	Clk2 = 40 MHz, source / 2

Trigger Select [TRIG_SEL] Bits 4-6

This is a 3-bit mask which selects the trigger source to cause the demonstrator to be stopped for the computer to read it out. Normally only one bit is set to select a single source but a bit pattern is used instead of a 2-bit code to slightly simplify the circuitry.

All bits 0	Stop signals will not be generated
Bit 6 = 1	Coincident *Beam Trigger* and electron trigger (i.e. any of *ASIC hits*)
Bit 5 = 1	Electron trigger (i.e. any of *ASIC hits*)
Bit 4 = 1	*Beam Trigger*
Other values	See sepate table

Full Trigger Select codes are, (the number is the hex value of bits 4 to 6):-

0 = No trigger, 1 = Beam Trigger only, 2 = Hit A only, 3 = Hit B only,
4 = Hit A or Hit B, 5 = Beam Trigger and Hit A, 6 = BeamTrigger and Hit B,
7 = Beam Trigger and (Hit A or Hit B).

Set Live [SET_LIVE] Bit 7

Writing a 1 to this bit-field immediately clears the dead time flip-flop, thus enabling "dead-timed" triggers. Its read value is always zero.

0	Read value, no effect on writing
1	Clear dead-time flip-flop

Set Dead [SET_DEAD] Bit 8

Writing a 1 to this bit-field immediately sets the dead time flip-flop, thus disabling "dead-timed triggers". Two bits (i.e. *Set Live* and *Set Dead*) are needed to allow the state of the dead time flip-flop to be pre-set an cleared during normal running. The module will automatically enable dead-timing of triggers after receipt of a valid trigger. Its read value is always zero.

0	Read value, no effect on writing
1	Set dead-time flip-flop

The combined effects of the *Set Live* and *Set Dead* are as follows:

SET_LIVE	*SET_DEAD*	*Dead Time Status*
0	0	Normal operation, cleared until set by trigger
0	1	Set
1	0	Cleared
1	1	Undefined—Do not use

Set Digitisation [SET_DIGIT] Bit 9

This bit field and *Set Hold* are used to immediately set and clear the flip-flop which drives the FADC Digitisation Gate (DIG_GATE) signal to the FADCs. They work in a similar fashion to *Set Live* and *Set Dead*. Writing a 1 to Set *Digitisation* pre-sets the flip-flop which allows the FADCs to digitise until the flip-flop is cleared by a Beam Trigger or *Set Hold* being set. Its read value is always zero.

0	Read value, no effect on writing
1	Clear FADC digitisation gate flip-flop

Set Hold [SET_HOLD] Bit 10

Setting *Set Hold* causes the FADCs to cease digitising input and be available for interrogation by the computer. Its read value is always zero.

0	Read value, no effect on writing
1	Set FADC digitisation gate flip-flop

The ordering of this pair of bit-fields, and the *Set Live*/*Set Dead* pair is the same—setting the first bit-field in each pair puts the flip-flop into its “normal” running state whilst setting the second bit-field sets the “system halted” state.

SET_DIGIT	*SET_HOLD*	*DIG_GATE*
0	0	Set until cleared by trigger
1	0	Set
0	1	Cleared
1	1	Undefined—Do not use

Enable Interrupts [EN_INT] Bit 11

When this bit is set, the module will issue VME interrupts on receipt of pulses on the Begin Burst, End Burst and Beam Trig inputs on the front panel. This allows the computer to initialise the interrupt handling software and ensure that the rest of the system is ready before it starts to receive interrupts from the TCM. All interrupts are disabled on board power up or reset.

0	Suppress generation of VME interrupts
1	Enable generation of VME interrupts

Further details on interrupt generation are described in the section on the *Interrupt Register*.

6. Interrupt Register [IR]

If a pulse is received on any of the Start Burst, End Burst or Beam Trig inputs, and interrupts are enabled (via the *Enable Interrupts* bit), then a D08(O) VME interrupt will be generated. The IRQ and status/ID byte are specified using jumpers.

The *Interrupt Register* holds a set of three counters representing the number of interrupts of each different kind seen since the register was last read out. Reading the register clears it.

If more than three interrupts of a particular kind are seen before the register is read (and thus cleared), the relevant counter limits at three. Thus, it indicates when 0, 1, 2 or more than 2 interrupts of each kind have been seen.

In all cases, a signal on any of the three inputs will always cause an interrupt unless one is already outstanding because of a prior signal on these inputs.

Bits	Function	Notes
0-1(LSB)	Count of *Start Burst* signals received	Cleared on reading
2-3	Count of *End Burst* signals received	
4-5	Count of *Beam Trig*signals received	
6-15	Unused	Always read as 0

A simple LynxOS device driver will be needed to handle TCM interrupts.

7. Status Register [SR]

This register can be read but not modified by the computer. It provides status information about the module.

Bit	Function	Abbreviation	Notes
0 (LSB)	Dead Time Enabled	DT_STATUS	
1	Digitisation Gate Enabled	DIG_GATE	
2-15	Unused	-	Always read as 0

Dead Time Enabled [DT_STATUS] Bit 0

This bit is set if the card is generating a *Dead Time* signal.

Digitisation Gate Enabled [DIG_GATE] Bit 1

This bit is set if the card is producing a *Digitisation Gate* signal for the FADC modules.

8. Tick Delay Registers [TRIG_IP_TICK, DIG_GATE_TICK, TXM_S_TICK, CPM_S_TICK]

These registers control the latency in ticks (a 'tick' is one System Clock period) on the *Beam Trigger* inputs and *Stop* outputs from the module. Further adjustment is possible using the *Fine Delay Registers*. The TRIG_IP_TICK Delay Register is 8 bits long and the DIG_GATE_TICK, TXS_S_TICK and CPM_S_TICK Delay Registers are each 6 bits long with unused most significant bits being always read as 0. They specify the delay in clock ticks. Internally, a counter is used to delay the signals so that the output signal appears after the specified number of clock ticks have been received since the signal was input into the counters. A precision of 8 bits allows a range of 0 to 255 ticks to be specified and 6 bits allows a range of 0 to 63 ticks to be specified. Minimum value is 1.

9. Fine Delay Registers [TXM_CLK1_FINE, CPM_CLK1_FINE, TXM_CLK2_FINE, DIG_GATE_FINE, TXM_S_FINE, CPM_S_FINE]

These seven registers control the latency on the *Beam Trigger,* inputs and*Clock* and *Stop* outputs from the module. Each register uses the least significant 5 Bits—the remaining most significant 11 bits being unchangeable 0s. The fine delays are in units of 1 nanosecond, so these registers allow values in the range 0 to 31 nanoseconds to be specified. Minimum value is 0.

10. The Front Panel LEDs

The LEDs are coloured according the following general code. Thus it is possible to get a rough indication of the health of the module from a single glance. Coloured LEDs also make them slightly easier to identify from across a room.

Green	Should generally be on—status OK type condition
Red	Should generally be off—error type condition
Yellow	Should be periodically on

The proposed LEDs are as follows, with the proposed labels which will be used for them if there is room on the front panel of the module.

Colour	**Interpretation**	**Label**
Green	Card is receiving power	POWER, 2 off
Green	Card has produced a clock pulse within approximately the last 0.7 seconds	CLK
Yellow	Interrupts are being generated. Card has received a *Start of Burst*, *End of Burst* or *Beam Trigger* pulse within the last 0.7 second or so	INTS
Yellow	Stop signals active	TRIG
Yellow	*Dead Time* (*DT_STATS*) active	DEAD TIME

11. An outline of how the card will be operated by the computer

Accesses to control registers are done by reading the register, changing any bits which need changing and then writing the new word back to the register.

Perhaps he only exception is in resetting the card to an initial known state which will usually be the first operation that the computer performs. It does this by writing to the *Control Register* with just the *Reset* bit set. This will have the important effects of disabling the generation of output signals from the module, as well as disabling interrupts.

Once this has been done, the computer should set the various delay registers in the TCM to values which have been previously determined by a mixture of experimentation and calculation. The *Control Register* should then be written to configure the control module. The following bit fields in should be set:

Clock Speed Select [CLK_SPEED]		Normally want 40 MHz clocks if we are going to use the internal oscillator
Trigger Select [TRIG_SEL]		Normally want coincident electron/beam triggers, but sometimes we may just want beam triggers if testing a module other than the CPM
Set Dead [SET_DEAD]		Select dead time to prevent any triggers for the moment

Once this has been done, the clock outputs can be enabled by setting *Clock Signal Source* (*CLK_SSRCE*). Generally, the internal oscillator will be selected by writing 11 to this bit field. At the same time interrupts can be enabled by setting the *Enable Interrupts* field.

Finally, triggers can be enabled by setting *Set Live*.

When the system is running, it is essentially interrupt driven. On receipt of an interrupt the computer should write burst or event records. If it resulted in a *Stop* signal (which can be found by interrogating the CPM), then the CPM and TXM can be read out.

During this time, there is the possibility of further interrupts—for example start or end of burst during event readout.. These will be serviced by the device driver, but an outstanding signal (of some sort) will be left for the DAQ software to act upon once processing of the current event is complete.

Once readout is complete, the system can be re-enabled by setting the *Digitisation Gate Enable* and *Set Live* bit fields simultaneously.

12. Information on setting the interrupt level for the TCM.

The level at which the TCM can assert an interrupt onto the VME bus is selectable by jumpers and may be any one of the seven levels available. Two sets of jumpers arre required. The first, a single jumper, routes the internal interrupt signal onto one of the interrupt request bus lines, IRQ1* to IRQ7*. The second set of jumpers configures the logic to recognise INT ACK cycles intended for the TCM.

The following jumpers set the interrupt request level.

A jumper on PL20 sets the IRQ = 7
A jumper on PL21 sets the IRQ = 6
A jumper on PL22 sets the IRQ = 5
A jumper on PL23 sets the IRQ = 4
A jumper on PL24 sets the IRQ = 3
A jumper on PL25 sets the IRQ = 2
A jumper on PL26 sets the IRQ = 1

We currently are using IRQ3 hence PL24 is jumpered

The second set of jumpers configures the logic to recognise an INT ACK cycle which is responding to an interrupt generated by the TCM. (An INT ACK cycle uses address lines A1, A2 & A3 to encode the level of interrupt being serviced.)
The jumpers must match the interrupt level in use by the TCM therefore jumpers PL17, PL18 and PL19 are set up accordingly. The circuitry has pull ups holding the 3 lines high (to logic 1) and jumpers, if fitted, would pull these lines low (to logic 0).

If using IRQ1 fit links to PL17 and PL18
If using IRQ2 fit links to PL17 and PL19
If using IRQ3 fit links to PL17
If using IRQ4 fit links to PL18 and PL19
If using IRQ5 fit links to PL18
If using IRQ6 fit links to PL19
If using IRQ7 fit no links, ie as current.

Since we currently use level 3 a link should be fitted to PL17.

THE RELEVANT SCHEMATIC IS SHEET 2 BUT THE SIGNAL LABELS LA1, LA3, BCD1 & BCD3 HAVE BEEN SWAPPED.

13.SETTING THE INTERRUPT VECTOR/ID.

In addition the VECTOR/ID value has to to set. This is a byte gated onto the LS 8 bits of the data bus. It is specified by jumpers or wiring on PL14 as follows:-

PIN 1 is a logic 1, PIN 10 is a logic 0, PINS 2 to 9 are connected to data bits 0 to 7 respectively. Hence any pin requiring a logic 0 is connected to pin 10 and any pin requiring a logic 1 is connected to pin 1. By default any pin left open circuit will be at logic 1 so in practice it is only necessary to wire the logic zeros. We currently use a VECTOR/ID OF 123 dec (7B hex) so links from pins 4 and 9 to pin 10 are required.