Local Trigger Unit

SOFTWARE MODEL

Revision 0.1 23 August 2004

DOCUMENT HISTORY

Version	Date	Editor	Comments
Draft 0.1	23.08.2004	P. Jovanović	First draft

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Preface

This document presents the *software model* of the LTU board, with inevitable gory details and added bits of “hardware wisdom” about the way to handle a plethora of LTU settings. Most of it is a compilation of notes produced during the development of the FPGA firmware to guide and help the writing of the first-level application and testing software.

The *Users* (see section **3.4.1**) are, by no means, encouraged to embark on the arduous (and frustrating) task of fully comprehending its content. The document is essential to the group of capable people (in Birmingham) involved in the development of a comprehensive, robust and (very) user-friendly software suite that shall guide the user - expertly and gently, step by step, with on-line comments, explanations and help - through the maize of the LTU options. That dream-piece of software is already well on its way - so watch this space!

This is a *working document*, currently in a draft form. As always, comments and suggestions from the ALICE community are invited and encouraged. The document shall be regularly updated, with changes explained in section ***Document History***.

1.2 Document overview

The present section (***Introduction***) gives a brief account of the purpose and scope of the document, explains its structure and lists cited documents. The table in the following section (***The VME Address Map***) includes all the VME words, with references to detailed descriptions, given in the final part (***Software Notes***).

1.3 References

[1] *ALICE Local Trigger Unit - Preliminary Design Review,* current version available on the ALICE CTP web site [3].

[2] *ALICE Central Trigger Processor - Preliminary Design Review,* current version available on the ALICE CTP web site [3].

[3] ALICE CTP web site: (ALICE→Projects→Trigger), or, directly, http://www.ep.ph.bham.ac.uk/user/pedja/alice/.

[4] Ph. Farthouat et al., *TTC-VMEbus Interface (TTCvi-MkII),* current version on the TTC web site: http://www.cern.ch/TTC/intro.html.

2.0 THE VME ADDRESS MAP OF THE LTU BOARD

Address	Mnemonic	Description	Wr	Rd	Section
		***VME Controller* FPGA**			
H"01"	CODE_ADD	LTU *Board Type* code		+	**3.2.1**
H"02"	SERIAL_NUMBER	LTU board *Serial Number*		+	**3.2.2**
H"03"	VERSION_ADD	*VME controller FPGA Firmware Version*		+	**3.2.3**
H"0A"	SOFT_RESET	LTU software reset (dummy data)	+		**3.2.1**
H"10"	FLASHACCESS_INCR	Flash memory access with add. increment	+	+	**3.5**
H"11"	FLASHACCESS_NOINCR	Flash memory access, no add. increment	+	+	**3.5**
H"12"	FLASHADD_CLEAR	Clear *Flash Memory Address*	+		**3.5**
H"13"	FLASH_STATUS	*Flash memory Status* word		+	**3.5**
H"14"	CONFIG_STATUS	*Configuration Status* word		+	**3.6.1**
H"15"	CONFIG_START	*Start Configuration* command	+		**3.6.1**
H"16"	TEMP_START	*Start Temperature Read* command	+		**3.6.2**
H"17"	TEMP_STATUS	*Temperature Read Status*		+	**3.6.2**
H"18"	TEMP_READ	*Temperature Read Data*		+	**3.6.2**
H"19"	MASTER_MODE	Master write destination (TTCvi or LTU)	+	+	**3.6.3**
		***LTU Logic* FPGA**			
H"20"	VERSION_ADD	*LTU logic FPGA Firmware Version*		+	**3.2.3**
H"30"	TEST_ADD	*LED Test* flag	+	+	**3.3.1**
H"31"	BC_STATUS	*BC Status* word		+	**3.20**
H"32"	BC_DELAY_ADD	BC delay register (5 bits, 1ns/bit)	+	+	**3.7.2**
H"33"	ADC_START	*ADC Start* command (dummy data)	+		**3.7.1**
H"34"	ADC_DATA	*ADC Data* (including **ADC BUSY** status)		+	**3.7.1**
H"35"	ERROR_ENABLE	*Enable Error Emulation* command	+		**3.15**
H"36"	ERROR_STATUS	*Error Emulation Status* word		+	**3.15**
H"37"	ERROR_SELECTOR	Error selector (*Set Error* mode)	+		**3.15**
H"38"	ERROR_DEMAND	*Error on Demand* request	+		**3.15**
H"39"	ERROR_RATE	Error rate setting	+	+	**3.15**
H"3A"	ORBIT_CLEAR	Clear emulator's **ORBIT** counter	+		**3.13**
H"3B"	EMULATION_START	*Start Emulation* command (dummy data)	+		**3.13**
H"3C"	QUIT_SET	*Set QUIT* command (dummy data)	+		**3.13**
H"3D"	BREAK_SET	*Set BREAK* command (dummy data)	+		**3.13**
H"3E"	LAST_BC	Number of the last BC in the LHC orbit	+	+	**3.8**
H"3F"	PREPULSE_BC	BC number of the **PREPULSE**	+	+	**3.8**
H"40"	CALIBRATION_BC	BC number of the calibration trigger	+	+	**3.8**
H"41"	GAP_BC	First BC of the LHC *Large Gap*	+	+	**3.8**
H"42"	ORBIT_BC	BC number of the emulator's **ORBIT**	+	+	**3.8**
H"43"	L1_DELAY	Emulator's **L0** to **L1** delay (in BCs)	+	+	**3.8**
H"44"	L2_DELAY	Emulator's **L0** to **L2** delay (in BCs)	+	+	**3.8**
H"45"	SLM_ADD_CLEAR	Clear *SLM Read* and *Write Addresses*	+		**3.13**
H"46"	SLM_DATA	*SLM Data* word		+	**3.13**
H"47"	EMU_STATUS	*Emulator BUSY* status		+	**3.13**
H"48"	START_SET	Settings for the **START** signal	+	+	**3.14**
H"49"	SOFT_TRIGGER	Software generated **START**	+		**3.14**
H"4A"	RUNDOM_NUMBER	Rate of the *Random* **START**	+	+	**3.14**
H"4B"	COUNT_PERIOD	Period of the *Scaled-down BC* **START**	+	+	**3.14**
H"4C"	TTC_INTERFACE	**L1** and **PREPULSE** polarity and delay	+	+	**3.9**
H"4D"	MODE	Setting of *Global/Stand-alone* mode	+	+	**3.22**
H"4E"	BUSY_ENABLE	**BUSY 1/BUSY2** input enable	+		**3.10**
H"4F"	BYSY_STATUS	**BUSY** status word		+	**3.10**
H"50"	SOFT_BUSY	Software generated **BUSY**	+		**3.10**

Address	Mnemonic	Description	Wr	Rd	Section
H”51”	L1_FORMAT	Setting of the *L1 Message* format	+	+	**3.23**
H”52”	L1MAX_CLEAR	Clear *L1 FIFO Maximum Occupancy*	+		**3.11**
H”53”	L2MAX_CLEAR	Clear *L2 FIFO Maximum Occupancy*	+		**3.11**
H”54”	SCOPE_SELECT	Selection of ScopeProbe **A/B** signals	+	+	**3.18**
H”55”	BACKPLANE_EN	Backplane output enable	+	+	**3.12**
H”56”	FIFO_MAX	*L1/L2 FIFO Maximum Occupancy*		+	**3.11**
H”57”	SOFTWARE_LED	Software driven LED (spare)		+	**3.3.1**
H”58”	COUNTER_CLEAR	Clear counters and timers (dummy data)	+		**3.16**
H”59”	L1FIFO_COUNTER	*L1 FIFO Nearly Full* counter		+	**3.16**
H”5A”	L2FIFO_COUNTER	*L2 FIFO Nearly Full* counter		+	**3.16**
H”5B”	PP_COUNTER	**PREPULSE** counter		+	**3.16**
H”5C”	L0_COUNTER	**L0** counter		+	**3.16**
H”5D”	L1_COUNTER	**L1** counter		+	**3.16**
H”5E”	L1S_COUNTER	**L1 STROBE** counter		+	**3.16**
H”5F”	L2A_COUNTER	**L2a** counter		+	**3.16**
H”60”	L2R_COUNTER	**L2r** counter		+	**3.16**
H”61”	START_COUNTER	**START** signal counter		+	**3.16**
H”62”	BUSY_COUNTER	**BUSY** counter		+	**3.16**
H”63”	BUSY_TIMER	**BUSY** timer		+	**3.16**
H”64”	SUBBUSY_TIMER	Sub-detector **BUSY** timer		+	**3.16**
H”65”	COUNTER_MASK	Counter/timer clear mask	+	+	**3.16**
H”66”	PIPELINE_CLEAR	*Pipeline Clear* command (dummy data)	+		**3.13**
H”67”	SSM_COMMAND	*SSM Command* word	+		**3.17**
H”68”	SSM_START	*SSM Start* command (dummy data)	+		**3.17**
H”69”	SSM_STOP	*SSM Stop* command (dummy data)	+		**3.17**
H”6A”	SSM_ADDRESS	*SSM Address* word	+	+	**3.17**
H”6B”	SSM_DATA	*SSM Data* word	+	+	**3.17**
H”6C”	SSM_STATUS	*SSM Status* word		+	**3.17**
H”6E”	TIMING_TEST	*Timing Test* word	+		**3.19**
H”80”	TEST_ADDRESS	*Test* word			

Address is the *local address* (see section **3.1.6**).

Mnemonic is the symbolic name used in the LTU FPGA firmware documentation.

Wr and **Rd** refer to the VME write and VME read access.

Very brief **Description** is expanded in the corresponding **Section**.

3.0 SOFTWARE NOTES

3.1 VME interface

The LTU board is VME64 compatible (J1 and J2 connectors with 96 pins).

3.1.1 *VME slave*

A24 (performs 24-bit address transfers across the backplane);
D32 (performs 32-bit data transfers across the backplane);
Responds to the following Address Modifier codes (AM[5..0]):

- H”39” - nonprivileged data access,
- H”3A” - nonprivileged program access,
- H”3D” - supervisory data access,
- H”3E” - supervisory program access.

VME base address:

- A[23..16] - H”81”;
- A[15..12] - set by the 4-bit *Board Address* hexadecimal dial;
- A[11..2] - 10 bits used to decode up to 1k of *local addresses*;
- A[1..0] - 0.

3.1.2 *VME requester*

The VME bus request **DWB** (*Device Wants Bus*) is set when either the *L1 FIFO* or the *L2 FIFO* is *not empty* (transmission of the *L1 Message*/*L2a Message*/*L2r Word*); if both FIFOs have data, the transmission from the *L1 FIFO* is given priority

Request priority level:	BR2*/BG2*.
Requester type:	RWD (release-when-done).

3.1.3 *VME master*

In normal operation, the LTU as the *VME master* always writes to the *FIFO2* of the corresponding TTCvi board [4]:

A24 (performs 24-bit address transfers across the backplane);
D32 (performs 32-bit data transfers across the backplane);
Address Modifier code (AM[5..0]): H”39” - nonprivileged data access;
VME write address:

- A[23..16] - H”80”;
- A[15..12] - set by the 4-bit *Board Address* dial;
- A[11..0] - H”0B8”.

3.1.4 *Board Address dial*

The LTU *Board Address* dial (rotary switch) is shown in **Figure 3.1**. The selected hexadecimal number (4 bits) becomes part of both the LTU VME slave address (section **3.1.1**) and the VME master address (section **3.1.3**).

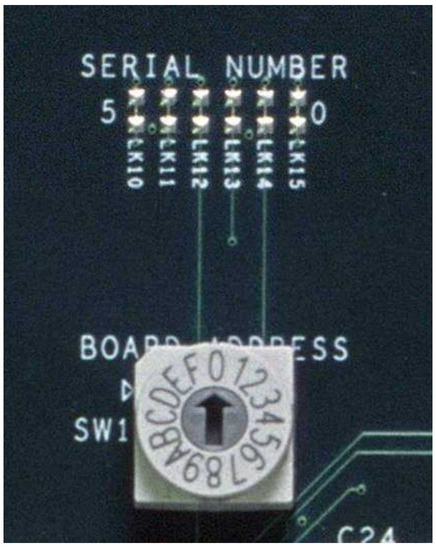

Figure 3.1 The *Board Address* dial and the *Serial Number* links

Although all the settings are possible, *only 0 to 7 are legal*. The range is sufficient since no more than 4 LTU boards shall ever be used in the same crate. The restriction is necessary since the dial also sets the 3-bit address of the I^2C ADC that controls the LTU supply voltages (section **3.20.1**).

Note: It is quite likely that in future versions of the LTU firmware only the 3 least significant bits of the dial will be used for setting the address. As a result, settings 5 and D, for example, would select the same address. The most significant bit of the dial could then be used for some other selectable option.

3.1.5 *Setting of the TTCvi base address*

The setting of the TTCvi base address is done with 4 hexadecimal rotary switches, SW1 - SW4. When the TTCvi board [4] is used as part of the ALICE trigger TTC partition, *the following setting is mandatory*:

SW1 - VME address bits A[23..20]:	**8**;
SW2 - VME address bits A[19..16]:	**0**;
SW3 - VME address bits A[15..12]:	*identical* to the setting of the LTU *Board Address* dial;
SW4 - VME address bits A[11..08]:	**0.**

Only with the mandatory setting, the LTU VME master shall address correctly the *FIFO2* of the corresponding TTCvi board.

3.1.6 *The LTU local addresses*

The LTU *local address* is decoded from the 10-bit address space that maps into the VME address bits A[11..2]. The local addresses are conveniently independent of the setting of the *Board Address* dial. They are generated by the FPGA firmware, used in the LTU documentation and referenced in the application software. Unless stated otherwise, here and in the rest of the document, the word ”address” stands for the *local address*.

The following example illustrates the “translation” of the local address H”37” of an LTU board with the *Board Address* dial set to 5 into a full 24-bit VME slave address:

A[23..16]:	H”**81**”	- mandatory, see section **3.1.1**;
A[15..12]:	**5**	- setting of the LTU *Board Address* dial;
A[11..2]:	H”**37**”	- *local address*;
A[1..0]:	**0**	- always 0, only D32 data transfer supported.

The local address H”37” (10 bits) corresponds to the full VME address H”8150DC” (24 bits).

Note: Similar conversions shall be performed automatically by the user-friendly application software; symbolic names shall be used instead of the numerical values.

3.2 The LTU board identification

The identification of all the ALICE trigger boards is explained in [2], section **3.24**.

3.2.1 *Board type code*

The LTU’s *Board Type* code is H”56”- data bits D[7..0] of the *Board Type Code* read word (*CODE_ADD*, address 1); the remaining data bits D[31..8] are undefined.

3.2.2 *Board serial number*

Each LTU board has a unique *Serial Number* - data bits D[5..0] of the *Serial Number* read word (*SERIAL_NUMBER*, address 2); the remaining data bits D[31..6] are undefined.

The serial number is set by the links shown in **Figure 3.1**: a printed short circuit, or a soldered 0Ω resistor reads 0; an open circuit reads 1. The board (decimal) serial number is also written with permanent ink on the component side of both the J1 and the J2 connectors.

3.2.3 *FPGA firmware version*

Firmware Version of the *VME Controller* FPGA: arbitrary 8-bit code, data bits D[7..0] of the *Firmware Version* read word (*VERSION_ADD,* address 3)*;* the remaining data bits D[31..8] are undefined.

Firmware Version of the *LTU Logic* FPGA: arbitrary 8-bit code, data bits D[7..0] of the *Firmware Version* read word (*VERSION_ADD,* address H"20")*;* the remaining data bits D[31..8] are undefined.

3.3 LED indicators

3.3.1 *Front panel LEDs*

The LTU front panel is shown in **Figure 3.3.1**.

All the front panel LEDs:

- are permanently *on*, if the corresponding signal is permanently asserted;
- are permanently *off*, if the signal is permanently cleared;
- *flash off* for ~0.3s, if the signal is asserted, but makes a short transition to the clear state;
- *flash on* for ~0.3s, if the signal is cleared , but makes a short transition to the asserted state;
- *toggle* on (~0.3s) and off (~0.3s), if the signal is continuously changing state.

A programmable bit causes all the front panel LEDs to continuously toggle. This is a quick way of testing the operation of the LEDs and their control electronics. The bit is asserted and cleared by the data bit D[0] of the *LED Test* word (*TEST_ADD,* local address H"30").

VME indicators

RD - VME read

Any VME *read* in the selected board address range (A[23..0], AM[5..0]).

WR - VME write

Any VME *write* in the selected board address range (A[23..0], AM[5..0]).

Clock indicators

BC - BC clock status

Complement of the **BC Error** signal (see section **3.20**). In normal conditions, the LED is permanently on.

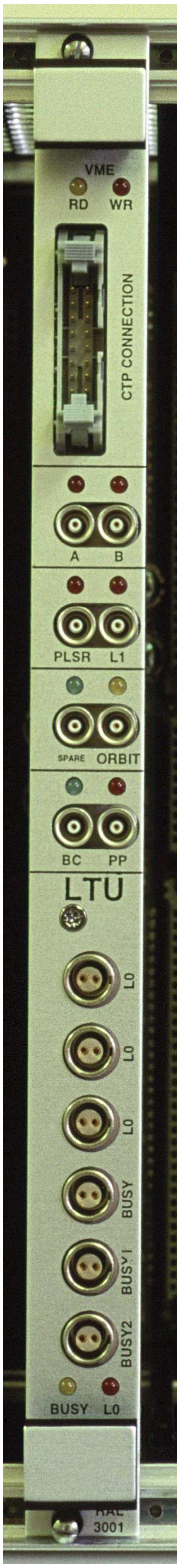

Figure 3.3.1 LTU front panel

ORBIT - ORBIT signal status

Complement of the **ORBIT Error** signal. In normal conditions, the LED is permanently on.

The **ORBIT Error** signal indicates that the ORBIT signal is not present, or its frequency is too low.

Signal indicators

PP - Pre-pulse output

L0 - L0 output

L1 - L1 output

BUSY - LTU *internal* **BUSY** signal (see [x], **Figure x.x)**

Note: The LED displays the status of the *internal* **BUSY** rather than the **BUSY** output because in the *Stand-alone* mode the **BUSY** output is permanently asserted; in the *Global* mode, on the other hand, the internal **BUSY** is identical to the LTU **BUSY** output

PLSR - Pulser input

Spare LED

The **Spare** connector and the LED are provided for future use. Currently, the connector provides an ECL output of the **BC** clock; the LED is controlled by the software - it flashes whenever the *Software LED* write word is generated (*SOFTWARE_LED*, local address H”57”, dummy data) and could be used for software development/debugging.

A, B - *ScopeProbe* LEDs

The LEDs display the status of the selected *ScopeProbe* output.

3.3.2 *PCB LEDs*

Two green SMD LEDs are mounted on the component side of the LTU PCB:

INIT - the *LTU Logic* FPGA correctly configured and initialised;

Note: In normal operation, the LED should be permanently on. If the LED is off, check/reload the content of the flash memory and configure the *LTU Logic* FPGA.

LOCK - the PLL of the *LTU Logic* FPGA locked to the delayed **BC** input.

Note: In normal operation, the LED should be permanently on. If the LED is off, make sure that the **BC** input is connected and correct **-** the front panel BC LED permanently on.

3.4 General software comments

3.4.1 *System* and *User*

The following definitions are an attempt to simplify the necessary task of setting the restrictions upon the software access to the LTU hardware.

- The ***System*** is the software suite developed by and the sole responsibility of the Birmingham CTP group.
- The ***User*** is either a member of the sub-detector group interacting with the LTU, or a software suite developed by the sub-detector group that controls the LTU application.

Only the ***System*** has a *direct and unrestricted access* to the LTU hardware. The ***User*** interacts with the LTU hardware either "manually", *via* the *dialog boxes* provided by the ***System***, or indirectly, *via* the application software, by addressing the drivers that are part of the ***System***. The ***System*** checks, helps and comments the ***User***'s access; the access could be refused if considered inappropriate or damaging.

3.4.2 *Parameters* and *Variables*

- ***Parameters*** are the LTU settings *accessible only by the **System***. Examples: the number of bunch-crossings in the LHC orbit; the location of the large LHC gap; the bunch-crossing number of the calibration trigger; *etc.* . There is no benefit in ***Users*** having access to those settings; also, arbitrary or erroneous settings are likely to cause confusing and difficult-to-debug system behavior.
- ***Variables*** are the LTU settings that enable the ***User*** to "construct" or execute the appropriate LTU application. They are *set indirectly*, by interacting with the ***System*** (see above). The means of interaction include *Dialog Boxes* (for settings and commands), the *Sequence Wizard* (an interactive sequence editor), the application drivers provided by the ***System***.

3.5 Flash memory interface

The flash memory on the LTU board stores the configuration data for the LTU logic FPGA. The data are transferred to the FPGA during the configuration (upon power-up, or initiated by the control processor).

Note: The flash memory chip is **Am29LV081B** (1M x 8-bit capacity; see data sheets for the operational details).The memory capacity exceeds the requirement for the selected FPGA and could store simultaneously three complete configurations (283kB each). Possible application: simultaneous alternative working configurations, the test set-up; *etc.* . The option is not currently used, but is available for future developments.

The flash memory is fully controlled by the control processor; the memory interface on the LTU board provides the necessary access.

Only the following command sequences are required (see the data sheet for details):

- *Read,*
- *Program (write),*
- *Chip Erase,*
- *Reset.*

Unlock Bypass, *Sector Erase, etc.*, could also be implemented, if found necessary.

Access to the flash memory

During the FPGA configuration (upon power-up; or started by the control processor), the *access to the memory is prevented*. The control program should verify the status of the **Configuration BUSY** signal (section **3.6.1**).

Memory address register

The flash memory address (f_add[19..0]) is generated by a 20-bit counter. The control program can clear the counter; the counter content can be automatically incremented *after* a read, or a write access (word *FLASHACCESS_INCR*).

For debugging purposes, in order to check the synchronism between the software and the hardware generated addresses, the control process can read the state of the 7 least significant bits of the address counter (f_add[6..0]).

Flash memory interface words

- *Clear Flash Memory Address* command (*FLASHADD_CLEAR* write word, local address H”12”, dummy data).
- *Flash Memory Data Read/Write with the Address Increment* after the access (*FLASHACCESS_INCR*, address H”10”, data bits D[7..0]).
- *Flash Memory Data Read/Write without the Address Increment* (*FLASHACCESS_NOINCR*, address H”11”, data bits D[7..0]).
- *Flash Memory Status* read word (*FLASH_STATUS*, D[7] - RY/BY status - see data sheets; D[6..0] - f_add[6..0]).

Typical erase and programming time

- Memory erase time (all data bits set to 1): 11s.
- Byte programming (write) time: 9μs.
- Serial configuration of the *VME Logic* FPGA (~0.5 MB): ~0.25s.

3.6 Functions of the *VME Controller* FPGA

There are two FPGAs on the LTU board:

- *VME Controller* FPGA (ALTERA MAX family EPM3512AFC256-7),
- *LTU Logic* FPGA (ALTERA Cyclone family EP1C12F324C7).

The main logic blocks implemented on the *VME Controller* FPGA are the following:

- VME bus interface (VME requester, VME master, VME slave - section **3.1**);
- Selection of the VME master write destination (section **3.6.3**);
- *Flash Memory* interface (section **3.5**);
- *LTU Logic* FPGA configuration controller (section **3.6.1**);
- board temperature measurement (section **3.6.2**);
- *Board Type* code (H"56") and the *Serial Number* registers (section **3.2**);
- *VME Controller* FPGA *Firmware Version* register (8 bits, section **3.2**).

3.6.1 *Configuration controller of the LTU Logic FPGA*

General approach

The configuration is performed automatically upon the power-up and, at any time, when requested by the control process.

The time-critical part of the process is performed entirely by the controller and does not require any processor intervention. The controller *always performs a complete configuration sequence* (proper timing, with "generous" safety margins, full content of the configuration file, *etc*.). The status of the FPGA at the end of configuration indicates the success or failure of the procedure. It is a task of the control process to find out the outcome (*Configuration Status* word) and, in case of an error, to decide and execute an appropriate next step (repeated attempt; check/read of the flash memory content; re-loading of the configuration file; signaling of a hardware failure; *etc*.). In case of an error, the controller does not attempt any automatic recovery procedure.

Signals controlled/monitored by the software

Start Configuration - write word (*CONFIG_START*, address H"15", dummy data) starts the configuration sequence.

Configuration Status word (*CONFIG_STATUS*, address H"14") reads the status of the following signals:

D[5] - **Enable DCLK** - enables **DCLK** (FPGA signal, see data sheets),
D[4] - **nCONFIG** (FPGA signal, see data sheets),
D[3] - **INIT_DONE** (FPGA signal, see data sheets),

D[2] - **CONF_DONE** (FPGA signal, see data sheets),
D[1] - **nSTATUS** (FPGA signal, see data sheets),
D[0] - **Configuration BUSY** - configuration in progress.

The software should monitor the **Configuration BUSY**, and, when the sequence is complete, read the status bits in order to find out whether the process was successful. For debugging proposes, a program should also be developed that reads and records the status word in frequent intervals during the entire configuration sequence, and presents/displays the data in an appropriate format (waveform plot).

Controller description

The timing circuit is an 11-bit counter (T[10..0]) clocked by the VMEbus **SYSCLK** signal (16MHz, 62.5ns period). First, it holds the **nCONFIG** signal low for ~63µs (1k **SYSCLK** ticks; the minimum time is 40µs). Then it waits for another ~63µs (1k **SYSCLK** ticks; the minimum time is again 40µs) before it enables the configuration clock **DCLK** (VMEbus **SYSCLK** - 16MHz; the maximum frequency is 100MHz). 512k 8-bit words are sequentially read from the flash memory, serialized (LSB first) and transmitted to the FPGA serial data input **DATA0**. At the end, the **DCLK** is disabled (set to 0), and the **Configuration BUSY** flag cleared. The number of transmitted words/bits somewhat exceeds the size of the configuration file (around 2.324 Mbits for EP1C12), but it adds a safety margin, simplifies the “firmware”, all at a cost of a negligible time overhead.

The whole sequence lasts 263ms.

3.6.2 *Board temperature measurement*

The PCB temperature measurement is performed by the TC74 temperature sensor.

The measurement is triggered by the *Start Temperature Read* command (*TEMP_START* write word, address H”16”, dummy data). The conversion takes 336µs, during which time the status flag **BUSY** is asserted. The state of the **BUSY** signal can be monitored with the *Temperature Read Status* read word (*TEMP_STATUS*, address H”17”, D[0]). At the end of the conversion (**BUSY** cleared), the measured value is read with the *Temperature Read Data* word (*TEMP_DATA*, address H”18”, D[7..0]); the data content is binary 2’s complement of the temperature in °C.

The *TEMP_START* is ignored while the conversion is in progress (**BUSY** asserted); similarly, the *TEMP_READ* is likely to return meaningless data.

3.6.3 *Selection of the VME master write destination*

In *normal* operation, the LTU, as a VME master, writes to the *FIFO2* of the corresponding TTCvi board (section **3.1.3**). In a special *test mode*, the LTU can be configured to write to the *SnapShot* memory of another LTU board. The option is only used for testing purposes to verify the correct operation of the LTU logic: the write data, instead of being sent to the TTCvi FIFO that cannot be read by the control processor, are sequentially stored into the *SnapShot* memory of another LTU; the test-software can check the content and the order of transmission of the stored LTU messages/words.

The *SnapShot* memory can store only the 18 least significant bits D[17..0] of the 32-bit wide word, normally transmitted to the TTCvi. This is only a minor inconvenience since the most significant bits D[31..16] are always set to H"8001" (see [1], section **3.6.1**).

Prior to being used, the *SnapShot* memory of the "spy" LTU must be put into the *VME Access/Write* mode (see section **3.17**).

The *Master Mode* word (*MASTER_MODE*, address H'19", write and read) selects the destination to which the LTU writes as the VME master:

Data bits	Content
D[4]	0: *normal operation* - TTCvi; 1: *test* - another LTU
D[3..0]	Setting of the *Address Dial* of the "spy" LTU (section **3.1.4**); *don't care* in normal operation.

The content of the *Master Mode* word is, *strictly*, the ***Parameter*** - to be handled only by the ***System*** (see section **3.4.2**).

3.7 ADC readout

3.7.1 *ADC interface*

The ADC "measures" the overlap of signals received from the CTP and the signals generated at the LTU as a function of the **BC** clock delay. The input is a DC voltage, "smoothed" by a low-pass/integrating filter with a time constant of about 100µs. For details - see the [1], section **3.14.12**.

The conversion is triggered by the *ADC Start* command (*ADC_START* write word, address H"33", dummy data). The conversion takes about 170µs, during which time the status flag **BUSY** is asserted. The state of the **BUSY** signal can be monitored with the *ADC Data* read word (*ADC_DATA*, address H"34", data bit D[8]). At the end of the conversion (**BUSY** cleared), the measured value is read with the same *ADC_DATA* read word (data bits D[7..0]). The 8-bit data content is a binary number proportional to the input voltage level (full scale - FFh, corresponds to ~3.3V). The exact value is irrelevant since the measurements are relative - no calibration is required.

Few words of warning about the programming of the **BC** delay scan sequence:

- An abrupt change of the delay setting could “unlock” the PLL in the FPGA (in principle - not yet experimentally confirmed). Following the delay change, first check that the loop is “locked” (**PLL Locked** bit D[1], *BC Status* word, section **3.20**).

- Following the delay change (and a successful PLL lock check), allow a generous time for the filtered input voltage to reach the final value (~0.5ms). Alternatively, repeat the readout until a stable result is achieved - readout differences settle within the expected noise level.

3.7.2 *BC delay*

The **BC** clock is first delayed by a programmable delay line and then used as a reference input to the PLL on the *LTU Logic* FPGA.

The data bits D[4..0] of the *BC Delay* word (*BC_DELAY_ADD*, address H”32”, write and read) define the programmable delay; the delay range is 0 to 31ns - 1ns *per* LSB.

3.8 Default values of some LTU software constants

The following list gives the default values (upon power on) for a number of system constants. The constants could be set to different values (VME write), or read back (VME read), with the corresponding VME words.

The constants affect the LTU operation only in the *Stand alone* mode (the CTP emulation).

The constants are, *strictly*, system ***Parameters*** (see section **3.4.2**).

Word mnemonic	Number of bits	Value
LAST_BC	12	3563
PREPULSE_BC	12	3436
CALIBRATION_BC	12	3556
GAP_BC	12	3446
ORBIT_BC	12	1
L1_ DELAY	9	224
L2_DELAY	12	3520

The emulated LHC orbit consists of [*LAST_BC*..0] bunch-crossing intervals.

The first bunch crossing of the emulated *Large LHC Gap* is *GAP_BC* (the last bunch crossing of the *Large LHC Gap* is bunch crossing 0).

The emulated **Pre-pulse** signal is generated in the *PREPULSE_BC* bunch crossing.

The emulated *Calibration Sequence* is “triggered” in the *CALIBRATION_BC* bunch crossing.

The emulated **Orbit** signal is generated in the *ORBIT_BC* bunch crossing.

The selection of values for the *LAST_BC, GAP_BC, PREPULSE_BC, CALIBRATION_BC* and *ORBIT_BC* should be *carefully harmonized*. Incompatible values will result in "strange" LTU behaviour (example: if only the value of the *LAST_BC* is reduced to 3500, the emulation might "hang" forever since the *CALIBRATION_BC* interval shall never be reached).

3.9 L1 output polarity and L1 and ORBIT delay

In order to maintain compatibility with both the TTCex board (normal configuration of the TTC partition) and the TTCvi board (currently unforeseen, but possible application), the *polarity* of the **L1** output needs to be programmable (see [1], section **3.14.7**).

The need to set correctly the timing of the **ORBIT** output, connected to the TTCvi board (see [1], section **3.14.8**), and the **L1** output, connected to the TTCex board (see [1], section **3.14.7**), calls for the signals' *delay* to be programmable. It suffice just to inserted, under the program control, a delay of 12.5ns - one half of the **BC** period; it is achieved by clocking the signal with the inverted **BC**.

The *TTC Interface* word (*TTC_INTERFACE*, address H"4C") sets (VME write) and reads back the status of the following flags that control the **ORBIT** and the **L1** outputs:

D[0]	***Delay ORBIT*** flag: 0 - no delay; 1 - delay;
D[1]	***Delay L1*** flag: 0 - no delay; 1 - delay;
D[2]	***L1 Polarity*** flag: 0 - positive (TTCex board; default); 1 - negative (TTCvi board).

The content of the *TTC_INTERFACE* word is, strictly, the ***Parameter*** - to be handled only by the ***System*** (see section **3.4.2**).

3.10 LTU BUSY signal

The operation of the **BUSY** logic on the LTU board is fully described in [1], section **3.9**.

Enable BUSY inputs

The *BUSY Enable* write word (*BUSY_ENABLE,* address H"4E") asserts and clears the *Enable BUSY1* (data bit D[0]) and *Enable BUSY2* (data bit D[1]) control flags. The flags are included in the *BUSY Status* read word.

Note: By design, an unconnected or not powered front panel **BUSY** input automatically assumes *BUSY* status. The configuration software must find out which inputs are used and enable/disable them accordingly.

Software generated BUSY

The *Software Generated BUSY* write word (*SOFT_BUSY*, address H"50") asserts and clears the *Software BUSY* (data bit D[0]) signal. The signal is included in the *BUSY Status* read word.

BUSY status

The *BUSY Status* read word (*BUSY_STATUS*, address H"4F") contains the following:

Data bit	Signal
D[7]	**BUSY** output,
D[6]	**BUSY2** input,
D[5]	**BUSY1** input,
D[4]	*Software BUSY* status,
D[3]	*L2 FIFO Nearly Full* status,
D[2]	*L1 FIFO Nearly Full* status,
D[1]	*Enable BUSY2* flag,
D[0]	*Enable BUSY1* flag.

3.11 Monitoring of the L1 FIFO and the L2 FIFO occupancy

The operation of both the *L1 FIFO* and the *L2 FIFO* is described in [1], sections **3.8.2** and **3.8.3**. The following options are included in the LTU logic in order to monitor the flow of the *L1/L2a/L2r Messages* and the occupancy of the FIFOs.

***Maximum FIFO Occupancy* word**

The *L1/L2 FIFO Maximum Occupancy* read word (*FIFO_MAX*, address H"56") contains the following:

Data bits	Content
D[15..8]	Maximum occupancy of the *L2 FIFO*
D[7..0]	Maximum occupancy of the *L1 FIFO*

The "maximum occupancy" is the maximum number of data words stored in the corresponding FIFO *at any time since the maximum occupancy register was last cleared* (see the next section).

***Clear Maximum FIFO Occupancy* register**

The *Clear L1/L2 FIFO Maximum Occupancy* write word (L1MAX_CLEAR, address H"52"/L2MAX_CLEAR, address H"53"; dummy data) clears the content of the corresponding *FIFO Maximum Occupancy* register; the content of the register is immediately updated to the current occupancy of the FIFO and increased whenever that value is exceeded.

More on the FIFO monitoring

The *FIFO Nearly Full* threshold for both FIFOs is set at 128 words (firmware constant, not programmable). The *FIFO Nearly Full* status is included in the *BUSY Status* word (section **3.10**). The status is accessible via the *ScopeProbe* option (section **3.18**) and it can be recorded with the *SnapShot* memory (section **3.17**). The instances of the *FIFO Nearly Full* status are counted (section **3.16**) and their contribution to the *LTU BUSY* status ([1], **Figure 3.9**) is timed (section **3.16**).

The *Leaky Bucket Controller* ([1], section **3.8.2**) halts (**HALT** signal) the VME master write to the TTCvi FIFO if the accumulated number of the VME master write operations, with the “leak” of one word every 60 **BC** intervals (1.5μs) taken into account, exceeds 200; both the leak rate and the **HALT** threshold are firmware constants - they are not programmable. The **HALT** status is accessible via the *ScopeProbe* option (section **3.18**) and it can be recorded with the *SnapShot* memory (section **3.17**).

3.12 LTU backplane connections/Logic Analyser connections

Backplane Output Enable word

The *Backplane Output Enable* word (*BACKPLANE_EN*, address H”55”, write and read) sets and clears bits that independently enable the LTU backplane signals; when cleared, a single bit tri-states all *Logic Analyser* connections.

Bit allocation:

Data bit	Enabled signal
D[0]	**BC**
D[1]	**ORBIT**
D[2]	**L2 DATA**
D[3]	**L2 STROBE**
D[4]	**L1 DATA**
D[5]	**L1/L1 STROBE**
D[6]	**L0**
D[7]	**PREPULSE**
D[8]	All *Logic Analyser* outputs

The setting of the enable bits is, *strictly*, the system ***Parameter*** (section **3.4.2**)

Logic Analyzer connections

The PL2 connector (38 pin, AMP MICTOR type, part number 2-767004-2) on the LTU board is compatible with the Tektronix logic analyzer probe P6434.

The option is intended for the LTU debugging and development; the connector will be soldered only on a small number of boards (2-3).

MICTOR pin	*Channel*	*Signal*
5	**CLK:0**	**BC**
37	**A2:0**	**ORBIT**
35	**A2:1**	**PREPULSE**
33	**A2:2**	**L0**
31	**A2:3**	**L1/L1 STROBE**
29	**A2:4**	**L1 DATA**
27	**A2:5**	**L2 STROBE**
25	**A2:6**	**L2 DATA**
23	**A2:7**	Sub-detector **BUSY** (synchronized)
21	**A3:0**	LTU **BUSY** (synchronized)
19	**A3:1**	**L1 FIFO Nearly Full**
17	**A3:2**	**L2 FIFO Nearly Full**
15	**A3:3**	**Leaky Bucket Halt**
13	**A3:4**	**START ALL** (selected **START** option)
11	**A3:5**	**Any Error**
9	**A3:6**	**VME write strobe**
7	**A3:7**	**VME read strobe**
14	**C1:4**	**Device Wants Bus**
12	**C1:5**	**MSTROBE**
10	**C1:6**	**TTC LS** (serialized TTCvi output, a header and 8 LS bits)
8	**C1:7**	**TTC MS** (serialized TTCvi output, a header and 8 MS bits)

3.13 LTU emulator

Sequence List memory (SLM)

The *Sequence List* memory utilises a single FPGA M4K block configured as a simple dual-port memory with a capacity of 256 16-bit words. The memory operates in a synchronous mode; both read and write addresses and the write data are registered.

The appropriate content of the memory is described in [1], **section 3.10.2**.

Note 1: The format of the emulation description file ([1], **Table 3.10.2)** has been slightly changed. For the updated version - see [2], **Table 3.6.1**.

The data that define the emulation sequence are stored into the memory from the VME bus; for debugging purposes, the memory content can also be read by the VME processor. During the emulation, the data are being continuously accessed by the emulator state logic.

Clear SLM Address word

The *Clear SLM Address* command (SLM_ADD_CLEAR, address H”45”, dummy data) clears both the SLM write (8 bits) and the SLM read (8 bits) address counters.

Note 2: The command *must not be generated during the emulation*, when the memory is controlled by the emulation logic and is being continuously accessed.

SLM Data word

The *SLM Data* read word (*SLM_DATA*, address H”46”, data bits D[15..0]) is used only for testing/debugging purposes.

Neither the VME write nor the VME read require the presence of the **BC** clock. The *Note 2* also applies.

Following the instruction, the corresponding (write or read) address counter is automatically incremented.

Clear Pipeline word

The “*Pipeline*” consists of two dual port memories, both addressed in a circular mode; they provide a programmable delay for the **L1** and **L2** signals, generated by the emulator logic.

Any arbitrary content of the memories needs to be “flashed out” *before* the emulated outputs are enabled. The *Clear Pipeline* command (*PIPELINE_CLEAR*, address H”66”, dummy data) triggers the “flashing out” process that takes around 200μs; during the “flashing”, the *Pipeline Busy* flag is asserted (see *Emulation Status* word).

The *Pipeline* is always active during the emulation, triggered, simultaneously with the emulation logic, by the *Start Emulation* word. At the end of an emulation run, in order to output signals that are still contained in the delay memories and complete the last emulated sequence, the *Pipeline* remains active for another ~200μs; during that time, the active status of the *Pipeline* is indicated by the asserted *Pipeline Busy* flag.

Start Emulation word

The *Start Emulation* command (*EMULATION_START*, address H”3B”, dummy data) starts the emulation process that is entirely controlled by the LTU logic. The emulation can only be performed in the *Stand-alone* mode.

Note 3: The LTU emulator is a synchronous system that can function correctly only if the **BC** clock is present. The software that controls the emulation should check the status of the **BC** clock before starting th**e** application (see section **3.20**).

Set Quit/Break commands

The *Set Quit* and the *Set Break* write words (*QUIT_SET*, address H”3C”, *BREAK_SET*, address H”3D”; dummy data) assert the *Quit*/the *Break* command flags (for functional details - see [1], section **3.10.3**). Both flags are automatically cleared by the hardware.

Note 4: Usually, the commands are executed instantly and the flags are cleared after a very short time. For that reason, the status of the flags in not made available to the control process/processor. The effect of the commands could be realised from other emulation status bits (example: *Emulation Busy* bit of the *Emulation Status* read word; also: the content of various counters - *Start*, *Pre-pulse*, *L0*, *L1*, *L2 Strobe*, *etc*.).

Emulation Status read word

The *Emulation Status* word (*EMU_STATUS*, address H”47”) reads the the status of the following signals:

Data bit	Signal
D[0]	*Emulation Busy*,
D[1]	*Pipeline Busy*,
D[2]	*Timing Test* flag status.

The *Emulation Busy* status bit is set by the *Start Emulation* word and cleared by the LTU hardware upon the completion of the emulation process (a single-pass sequence pattern; a result of setting the *Quit* or the *Break* flag; for details, see [1], section **3.10.3**).

The meaning of the *Pipeline Busy* status bit is explained in section ***Start Emulation* word**.

The *Timing Test* flag is described in section **3.19**.

Clear ORBIT Counter command

The *Clear ORBIT Counter* command (*ORBIT_CLEAR*, address H”3A”, write only, dummy data) clears the *ORBIT Counter*.

The free-running 24-bit counter is part of the CTP emulator; it provides the event identification data of the *L2a Message*.

Writing the sequence data into the SLM

- Make sure that the LTU is in the *Stand-alone* mode and that the emulation is not running (*Emulation Busy* bit of the *Emulation Status* read word).
- Clear the SLM address counters (*Clear SLM Address* write word).
- Write sequentially the sequence description data into the SLM (*SLM Data* write word).

Note 5: The capacity of the SLM provides for 32 sequences. If the full capacity is not used, the control process needs only to write the data for the defined sequences. Nevertheless, it might be convenient to always write the full memory content (256 words), filling the unused slots with a fixed “blank sequence” pattern (generated by the sequence editor). The option could be used/useful for software checks of the SLM content.

- Generate an additional *SLM Data* write instruction (dummy data, all 1s, for example). This “dummy” write is required by the hardware to properly complete the last “useful” write operation; if omitted, the last data word *will not be stored* in the SLM.

Reading the content of the SLM

- Make sure that the LTU is in the *Stand-alone* mode and that the emulation is not running (*Emulation Busy/Pipeline Busy* bits of the *Emulation Status* read word).
- Clear the SLM address counters (*Clear SLM Address* write word).
- Read sequentially the data from the SLM (*SLM Data* read word).

The reading of the SLM content is primarily a testing/debugging option, but it could/should also be used to enhance the robustness of the LTU’s control software. The SLM content should be checked at a convenient stage of the emulation setting to make sure that it is not corrupted and is consistent with the adopted format. The consistency checks could include the following:

- At least one *Last* flag asserted.
- No illegal *Sequence Codes* (0) before the last sequence.
- The same status of the ClT flags and the corresponding L1SwC/L2swC flags in the words 0 and 4.
- The “blank sequence” data (see *Note 5*) in the unused slots.
- *Etc. ...*

Running the emulation

- Make sure that the LTU is in the *Stand-alone* mode and that the **BC** clock is present (see *Note 3*).
- Check that the SLM memory contains properly formatted sequence data (*optional*; for details, see the previous section).
- Select the appropriate source of the *Start* signal (see [1], section **3.10.7**).
- If appropriate, select the configuration of the error-emulation circuit (see [1], section **3.10.9**).
- Clear the *Pipeline* (see section ***Start Emulation* word**).
- Wait until the *Pipeline Busy* flag is cleared (~200μs).
- Start the emulation - ***Start Emulation* word**.

- Monitor the operation of the emulator - *Emulation Busy*/*Pipeline Busy* bits of the *Emulation Status* read word; content of various counters - *Start*, *Pre-pulse*, *L0*, *L1*, *L2 Strobe*, *etc*. .
- If required, terminate the emulation using the *Quit* or the *Break* command (*Set Quit/Set Break* word).
- Check that the emulation has been terminated (*Emulation Busy/Pipeline Busy* bits of the *Emulation Status* read word).

3.14 Emulation START signal

The **START** signal generator circuit is based on **Figure 3.10.7(b)** [1]; details are given in [1], section **3.10.7**.

START signal selector

The *START Set* word (*START_SET*, address H"48", write and read) selects among the available options for the generation of the **START** signal:

Data bit	Signal
D[3]	*LHC Gap Veto***,**
D[2]	*Pulser Mode* flag,
D[1..0]	*START Selector* code.

LHC Gap Veto

If 1, all **START** options are vetoed during the *LHC Gap* interval (unless the pending sequence is a calibration trigger).

Pulser Mode flag

If 1: *Edge Sensitive* mode - a rising edge of the **PULSER** input generates a single 25ns-pulse synchronous with the **BC** clock;

If 0: *Level Sensitive* mode - the **PULSER** input synchronized with the **BC** clock;

START Selector

Code	Selection
0	no input selected,
1	**PULSER** input,
2	*Random START*,
3	*Scaled-down BC*.

Software START

The *Software START* command (*SOFTWARE_TRIGGER*, address H"49", dummy data) generates a single 25ns-pulse synchronous with the **BC** clock.

The option is always enabled; the signal is *OR*ed with any other options selected by the *START Selector* and subject to the *LHC Gap Veto*.

PULSER input

The necessary synchronization of the **PULSER** input introduces a jitter of up to 25ns and a delay of 25ns (1 **BC** interval).

Random START

A circuit, based on a 31-bit linear feedback shift register, generates a random pattern of 25ns pulses synchronous with the **BC** clock; the pulse distribution is pseudo-random - the pattern repeats itself every ~53s ([2^{31} - 1] BC intervals).

The *Random Number* word (*RANDOM_NUMBER*, address H”4A”, data bits [30..0], write and read) sets the number of pulses distributed within each pseudo-random pattern-repetition period (range: 1 to [2^{31} -1]).

The *Random START* generator is free running, enabled if the option is selected (see above) in the *Stand-alone* mode and only during the emulation (*Emulation BUSY* asserted).

Scaled-down BC START

A 30-bit counter generates a 25ns pulse synchronous with the **BC** clock, with the programmable *interval between the pulses* in the range from 0 (continuously asserted output) to ~25s.

The *Counter Period* word (*COUNT_PERIOD*, address H”4B”, data bits [29..0], write and read) sets the duration in BC intervals (25ns) of the *interval between the pulses*.

The *Scaled-down BC START* generator is free running, enabled if the option is selected (see above) in the *Stand-alone* mode and only during the emulation (*Emulation BUSY* asserted).

General comment

The **START** signal is *NOT* a sequence trigger; it is just an input that *might* start the trigger sequence, providing that the sub-detector is ready to receive it (not **BUSY**). Especially at high rates, most of the **START** pulses will coincide with the emulator/sub-detector **BUSY** state and will be ignored. For that reason, their rate/average rate is irrelevant. On the other hand, that rate affects *the pause between consecutive sequences*, which is one of major concerns in testing the sub-detector front-end electronics. The “randomness” of that interval is provided, practically equally, by *both Random START* and the *Scaled-down BC START*, since the **START** generation and the trigger sequence execution are *mutually independent*.

At lower rates, the **START** frequency approaches the sequence trigger-frequency. If the relatively poor “randomness” of the pulse distribution at those lower rates (~1Hz and beyond) is a concern, the “*Software START*” should be used instead.

3.15 LTU error emulation

General

In addition to the options described in [1], section **3.10.9** (*Set Error* in the text that follows), the LTU provides another mechanism of error emulation - *Error on Demand*. Both options are equally available in *Stand-alone* and *Global* mode. The *Error on Demand* request can be issued “in parallel” - during the *Set Error* emulation.

The *Error on Demand* request targets a selected signal/message in the trigger sequence and generates a *single error* at the first occurrence of the targeted entity. In the *Set Error* mode, any set of signals/messages could be simultaneously selected and the corresponding errors occur whenever the targeted entity coincides with the time tag - a BC interval randomly distributed in time, with the programmable average rate. In the *Stand-alone* mode, the *Set Error* option applies only to sequences that have the *Error Prone* flag asserted (for details, see [1], section **3.10.9**), which enables, for debugging and development purposes, “mixing” of sequences that are *always correct* with the others that are *prone to emulated errors*.

Enable Error Emulation command

The error emulation options operate in both the *Global* and the *Stand-alone* modes. In order to reduce the risk of accidental (or malicious?) invocation, an additional flag - *Enable Error Emulation*, is introduced that, when cleared, disables *all* the mechanisms of error generation.

The *Enable Error Emulation* word (*ERROR_ENABLE*, address H”35”, data bit D[0]) asserts and clears the *Enable Error Emulation* flag.

The status of the flag is included in the *Error Emulation Status* word.

Error Selector word (*Set Error* mode only)

The *Error Selector* word (*ERROR_SELECTOR*, address H”37”) selects signals/messages that could have their transmission prevented in order to create the error condition. A signal/message is selected when its corresponding bit is asserted:

Data bit	Selected signal/message
D[0]	Enable *Pre-pulse Error,*
D[1]	Enable *L0 Error,*
D[2]	Enable *L1 Error,*
D[3]	Enable *L1 Message Error,*

D[4]	Enable *Simultaneous L1 and L1 Message Error*,
D[5]	Enable *L2a Message Error*,
D[6]	Enable *L2r Word Error*.

If both the *L1* and the *L1 Message Error* (data bits 2 and 3) are selected, the corresponding errors never occur simultaneously (in the same trigger sequence).

When the *Simultaneous L1 and L1 Message Error* is selected (data bit 4), the individual *L1 Error* or *L1 Message* E*rror* never occur, even if they are selected (data bits 2 and 3).

Current status of the error selector bits is included in the *Error Emulation Status* word.

Error timing (*Set Error* mode only)

An error - deliberately prevented transmission of a selected signal/message (see section ***Error Selector* word**), occurs when the signal, or the message head, coincides with the BC interval "tagged" by a pseudo-random signal generator ([1], section **3.10.9**).

The random signal generator is based on a free-running 31-bit linear feedback shift register; the duration of the pseudo-random sequence is ~ 53s. The programmable output signal rate is controlled by the *Error Rate* word (*ERROR_RATE*, address H"39", data bits D[30..0], write and read) that sets the probability (in percents) of the targeted signal/message coinciding in time with the random error time-tag and producing an error. The resolution of the probability setting, required by the application, decides the appropriate number of data bits in the word. Example: 10 bits provide for 1 permil resolution (NN.N % setting); 14 bits give 0.1 permil resolution (NN.NN % setting); *etc*. .

Error on Demand option

Error on Demand is an addition to the options described in [1]. The option is available in both *Global* and *Stand-alone* modes.

The error is "requested" by the control software. The *Error on Demand* command (*ERROR_DEMAND*, address H"38", write only) sets the *Error Code* and asserts the *Error Request* flag. The flag is cleared by the hardware upon the generation of the requested error condition; by the *Error on Demand* write word with the *Error Code* 0 (see below); by clearing the *Enable Error Emulation* flag (see section ***Enable Error Emulation* flag**).

Error Code (data bits D[2..0]):

0 - no action, clears pending *Error Request* (if any),
1 - *Pre-pulse Error*,
2 - *L0 Error*,
3 - *L1 Error* only,
4 - *L1 Message Error* only,

5 - *Simultaneous L1 and L1 Message Error*,
6 - *L2a Message Error*,
7 - *L2r Word Error*.

The *Error on Demand* word is ignored (has no effect) if it is invoked when the *Enable Error Emulation* flag is cleared.

Current status of the *Error Request* flag and the last selected *Error Code* are part of the *Error Emulation Status* word.

Error Emulation Status word

The *Error Emulation Status* read word (*ERROR_STATUS*, address H”36”) contains the following:

Data bits	Content
D[11]	*Enable Error* flag,
D[10]	*Error Request* flag,
D[9..7]	*Error Code* (3 bits),
D[6..0]	*Error Selector* (6 bits).

3.16 LTU counters and timers

General

All the counters/timers use the *Gray* code in order to enable the readout “on the fly”; during the VME read, the content of the selected counter/timer is latched into the common 32-bit register and automatically converted into the binary format.

Note: The mechanism of error generation might cause some unexpected disagreements in counting results. Examples: **L1** and **L1 Strobe** counts might not be the same; the sum of **L2a** and **L2r** might be different from the **L1/L1 Strobe** count; *etc*. .

Clear Counter command

A counter/timer is cleared with the VME write word *Clear Counter* (*COUNTER_CLEAR*, address H”58”, dummy data) if the corresponding *Clear Mask* bit is asserted; any sub-group or all the counters/timers can be simultaneously cleared.

Clear Mask word

The *Clear Mask* bits are set and cleared with the Counter Mask word (*COUNTER_MASK*, address H”65”, data bits D[11..0], write and read); accessing the word has no immediate effect on the counters. The allocation of *Clear Mask* bits is shown in the following tables (“clear bit”).

Counter allocation

1. The following counters are 32-bit *Gray-code* counters clocked by the rising edge of the input signal:

Counter	Input signal	Readout word	Clear bit
Pre-pulse Counter	**Pre Pulse**	*PP_COUNTER*	2
L0 Counter	**LO**	*LO_COUNTER*	3
L1 Counter	**L1**	*L1_COUNTER*	4
L1 Strobe Counter	**L1 Strobe**	*L1S_COUNTER*	5

Note:
- The **Pre-pulse** signal is 200ns wide.
- The **L1** signal has a programmable polarity.
- The *L1 Counter* and the *L1 Strobe Counter* should produce the same count, unless the errors are introduced.

2. The following counters are 32-bit *Gray-code* counters clocked by the **BC** when the corresponding input signal is asserted:

Counter	Input signal	Readout word	Clear bit
L2a Counter	**L2a**	*L2a_COUNTER*	6
L2r Counter	**L2r**	*L2r_COUNTER*	7
START Counter	**START**	*START_COUNTER*	8

Note: All the input signals are 25ns-wide pulses. In case of the **START** input [2], the pulses could occur in adjacent intervals and appear “joint” (*PULSER* input in *Level Sensitive* mode; *Scaled-down BC START* at maximum rate; *Random START*), but, nevertheless, they will be *all counted*.

3. The *BUSY Counter* is also a 32-bit *Gray-code* counter, triggered by the rising edge of the *sub-detector* **BUSY** signal ([1], **Figure 3.9**). It *counts the occurrences* of the signal; the time the signal is asserted is measured by the *Sub-detector BUSY Timer* (see below).

Counter	Input signal	Readout word	Clear bit
BUSY Counter	Sub-detector **BUSY**	*BUSY_COUNTER*	9

4. The following counters are 8-bit *Gray code* counters, clocked by the rising edge of the input signal. The counting stops when a counter reaches its maximum content - 255; the counter overflow is prevented.

Counter	Input signal	Readout word	Clear bit
L1 FIFO Nearly Full Counter	**L1 FIFO Nearly Full**	*L1FIFO_COUNTER*	0
L2 FIFO Nearly Full Counter	**L2 FIFO Nearly Full**	*L2FIFO_COUNTER*	1

Note: The counters count the occurrences of the **L1/L2 FIFO nearly full** status. In normal operation, this condition is unlikely to occur.

5. The timers are also 32-bit *Gray-code* counters, clocked with a ~ 0.4μs clock (**BC** signal divided by 16) when the input signal is asserted. The **LTU BUSY** (**BUSY** in **Figure 3.9** [1]) includes the contribution (if any) from both the *Software BUSY* and the *L1/L2 FIFO Nearly Full* status.

Timer	Input signal	Readout word	Clear bit
LTU BUSY Timer	**LTU BUSY**	*BUSY_TIMER*	10
Sub-detector BUSY Timer	Sub-detector **BUSY**	*SUBBUSY_TIMER*	11

3.17 LTU SnapShot memory

General

The capacity of the LTU Snapshot memory (SSM) is 1M words of 18 bits. When the data are recorded, the “snap-shot” covers 26.2ms of real time (294 full LHC orbits).

The SSM controller, part of the *LTU Logic* FPGA, supports two *operations*: *Recording* and *VME Access*. Each *operation* has two *modes*. The modes in the *Recording* are (sampling) *After* and (sampling) *Before*; the modes in the *VME Access* are *Read* and *Write*.

The control software must first set the *operation* and specify the *mode*. In order to protect the hardware, some actions are performed only if the setting of the *operation*/the *mode* is appropriate; otherwise the outcome is either no action or incorrect result.

The *operation* and the *mode* are set by the *SSM Command* word.

Note 1: The SSM is a synchronous memory and, in the *Recording* operation, can only function if the **BC** clock is present. The software that controls the memory should check the status of the **BC** clock before starting th**e** SSM application (the *BC Status* flag is available **-** see section **3.20**).

Note 2: The address and both the input and the output data of the SSM are registered. The read operation requires additional (dummy) reads in order to push the external address to the address register, and to pull the data from the output register. For correct operation, the control software should follow a recommended sequence (explained later).

Operation *Recording*

(Sampling) *After* mode:

Following the command *Start* (recording), the input data are sampled in each **BC** clock cycle and stored into consecutive memory locations. The command also asserts the *SSM BUSY* status flag. When the memory becomes full, the operation stops and the *BUSY* flag is cleared.

(Sampling) *Before* mode:

Following the command *Start* (recording), the input data are sampled in each **BC** clock cycle and stored into consecutive memory locations. The command also asserts the *SSM BUSY* status flag. When the memory becomes full, it overflows and overwrites previously recorded data. The command *Stop* (recording) terminates the sampling of input data and clears the *BUSY* flag.

If the commands *Start* or *Stop* are attempted when the set operation is not *Recording*, they are ignored by the hardware. Both commands are generated by writing to a VME address (dummy data). The LTU logic synchronizes the commands with the **BC** clock and, if the clock is not present, the commands have no effect. The *SSM BUSY* flag is included in the *SSM Status* word.

Operation *VME access*

(VME) *Read* mode:

The VME *SSM Data* read instruction:

- increments the content of the *SSM Address* register (front edge of the read strobe);
- reads the *current content* of the output data register;
- enters the *SSM Address* into the memory address register and updates the output data register (trailing edge of the read strobe).
- If the instruction is attempted when the set operation is not *VME Access*, or the set mode is not *Read*, the returned data will be wrong and the *SSM Address* register won't be affected.

(VME) *Write* mode:

The VME *SSM Data* write instruction:

- increments the content of the *SSM Address* register (front edge of the write strobe);
- enters the *SSM Address* into the memory address register, enters the write data into the memory input data register and performs the memory write cycle (trailing edge of the write strobe).

If the instruction is attempted when the set operation is not *VME* A*ccess*, or the set mode is not *Write*, the data will be ignored and the *SSM Address* register won't be affected.

***SSM Command* word**

The *SSM Command* write word (*SSM_COMMAND*, address H"67"):

D[1]	*Operation* bit,
D[0]	*Mode* bit.

Bit coding:

Operation	*Mode*	*Function*
0	0	*VME Access, Read*
0	1	*VME Access, Write*
1	0	*Recording, After*
1	1	*Recording, Before*

Note 3: The *SSM Command* automatically clears the *SSM BUSY* flag - the option is made available as a way of clearing the *SSM BUSY* in the absence of the **BC** clock (a “softer” recovery alternative to the board reset). The control software should not plan to use the option - the condition should never happen in normal operation.

Note 4: The *SSM Command* also disables the SSM; the option is introduced in order to protect the hardware - to avoid the data bus contention. The SSM is automatically selected (chip enable asserted) with each legal memory read or write operation. This is just an explanation of the way the hardware works; no action is required from the control software.

SSM Status word

The *SSM Status* read word (*SSM_STATUS*, address H”6C”):

Data bit	Signal
D[2]	*SSM BUSY,*
D[1]	*Operation,*
D[0]	*Mode.*

The *Operation* and the *Mode* flags are set by the *SSM Command* word; the *SSM BUSY* flag is controlled by the hardware.

SSM Address word

The *SSM Address* word (*SSM_ADDRESS*, address H”6A”):

Data bits	Signal
D[20]	*Overflow* (read only),
D[19..0]	*SSM Address* [19..0] (write and read).

The word sets or reads the content of the *SSM Address* register and reads the status of the *Overflow* flag. If the set operation is *Recording* and the *SSM BUSY* flag is asserted, the write data are ignored and the read data are arbitrary (illegal operation).

In the *Recording* operation, the *Overflow* flag indicates the overflow of the *SSM Address* register; in the *Recording/Before* mode, the status of the flag affects the way the data should be read (see later).

The *Overflow* flag is automatically cleared if the operation is not *Recording*, or by any attempt to set/change the content of the *SSM Address* register (*SSM Address* write word).

***SSM Data* word**

The *SSM Data* write word (*SSM_DATA*, address H”6B”) is used only for the testing of the SSM hardware.

Bit allocation of the *SSM Data* read word when the SSM is *Recording*:

Data bit	Signal
0	**Orbit**
1	**Pre-pulse**
2	**L0**
3	**L1/ L1 Strobe**
4	**L1 Data**
5	**L2 Strobe**
6	**L2 Data**
7	Sub-detector **BUSY** (synchronized)
8	LTU **BUSY** (synchronized)
9	**L1 FIFO Nearly Full**
10	**L2 FIFO Nearly Full**
11	*Leaky Bucket* **HALT**
12	**MSTROBE**
13	*TTC LS* (serialized TTCvi output, a header set to 1 and *8 LS bits*)
14	*TTC MS* (serialized TTCvi output, a header set to 1 and *8 MS bits*)
15	**STROBE**
16	**START ALL** signal (emulator, selected **START** option)
17	**Any Error**

***SSM Start* word**

The *SSM Start* word (*SSM_START*, address H”68”, dummy data) generates the *SSM Start* command. The word is ignored if the selected operation is not *Recording.* If the **BC** clock is not present, the instruction has no effect - it will not set the *SSM BUSY* flag nor start the recording.

***SSM Stop* word**

The *SSM Stop* word (*SSM_STOP*, address H”69”, dummy data) generates the *SSM STOP* command. The word is ignored if the selected operation is not *Recording.* If the **BC** clock is not present, the instruction has no effect - it will not clear the *SSM BUSY* flag nor terminate the recording.

Recording and data reading sequence

- Check the status of the **BC** clock - data recording is only possible if the clock is present (see *Note 1*).
- Check *SSM BUSY* flag (*SSM Status* word). If asserted, generate the *SSM Stop* command and then check again that the *SSM BUSY* is cleared (warning: the asserted *SSM BUSY* indicates that the recording is in progress; the control software should make sure that it does not interfere with the running application).
- Set the operation - *Recording,* and the mode - *Before* or *After* (*SSM Command* word).
- Clear the *SSM Address* register (*SSM Address* write word, data *all 0*); this automatically clears the *Overflow* flag.
- At the appropriate time generate the *SSM Start* command.
- Check (in less than 26ms) that the *SSM BUSY* flag is asserted (if not - a hardware error has occurred).
- If the selected mode is *After*:
    - Wait until *SSM BUSY* is cleared (around 26.2ms after the *SSM Start* - a longer delay indicates a hardware error).
    - Check that the content of the *SSM Address* register is 0 and the *Overflow* flag is cleared (if not - a hardware error has occurred).
    - Set operation/mode to *VME Access/Read*.
    - Clear the *SSM Address* register (not necessary - it should be already cleared, but it is a prudent thing to do).
    - Read 1M *SSM Data* words.
    - Check, finally, that the content of the *SSM Address* register is 0 (if not - a hardware error has occurred).
- If the selected mode is *Before*:
    - At the appropriate time generate the *SSM Stop* command.
    - Check that the *SSM BUSY* flag is cleared (if not - a hardware error has occurred).
    - Read that the content of the *SSM Address* register and the status of the *Overflow* flag (*SSM Address* word). The SSMADD is the memory address into which the last data word has been stored.
    - Set *VME Access/Read*.
    - If the *Overflow* flag is 0:
        - Clear the *SSM Address* register.
        - Read SSMADD (see above) *SSM Data* words (the sequence includes two dummy reads - see the section ***VME read sequence***).

    - Check, finally, that the content of the *SSM Address* register is (SSMADD + 2); if not - a hardware error has occurred.
  - If the *Overflow* flag is 1:
    - Read 1M *SSM Data* words, starting from the address (SSMADD + 1). The initial *SSM Address* setting is SSMADD - see the section ***VME read sequence.***
    - Check, finally, that the content of the *SSM Address* register is (SSMADD + 2); if not - a hardware error has occurred.

VME write sequence

The sequence is *only used for testing* of the SSM hardware: data write, followed by read and compare.

In the *VME Access* operation, the VME strobe is used as the SSM clock. In the VME write, the data are stable only at the falling edge of the strobe and that edge is used to clock the SSM (to enter the address and the data into the address and data input registers of the SSM). The rising/front edge of the strobe increments the *SSM Address* register.

The following sequence writes ***n*** data words D1..Dn into the addresses A1..An. Ax and Dx are undefined address and data values; *During* and *After* refer to the time during and after the VME write strobe. The table illustrates the propagation of the address and the data through the pipelined structure of the SSM.

- Set the *SSM address* to A1-1.

	SSM Address Counter	SSM address register		SSM input data register	
	During and *After*	*During*	*After*	*During*	*After*
• Write D2	A2	A1	A2	D1	D2
• Write D3	A3	A2	A3	D2	D3
• **...**					
• Write D1	A1	Ax	A1	Dx	D1
• Write Dn	An	An-1	An	D(n-1)	Dn

- Read the *SSM Address* An - the memory address in which the last data word Dn is stored (only a monitoring check).

Note 5: During the recording (see **Recording and data reading sequence**), the initial *SSM Address* setting is 0. As a result, the first data word is stored into the memory address 1; if the recording mode is *After*, the last data word is stored in the address 0. This is just a “technical detail”, of no concern to the users/software.

VME read sequence

In the VME read, similar to the VME write, the falling edge of the read strobe is used to clock the SSM (to enter the address into the address register and the memory data into the output data register of the SSM). The rising/front edge of the strobe increments the *SSM Address* register.

The following sequence reads ***n*** data words D1..Dn from the addresses A1..An. Ax and Dx are undefined address and data values; *During* and *After* refer to the time during and after the VME read strobe. The table illustrates the propagation of the address and the data through the pipelined structure of the SSM.

- Set the *SSM Address* to A1-1.

	SSM Address Counter	SSM address register		SSM output data register	
	During and *After*	*During*	*After*	*During*	*After*
• Dummy read	A1	Ax	A1	Dx	Dx
• Dummy read	A2	A1	A2	Dx	D1
• Read D1	A3	A2	A3	D1	D2
• Read D2	A4	A3	A4	D2	D3
• Read D3	A5	A4	A5	D3	D4
• **...**					
• Read Dn	An+2	An+1	An+2	Dn	D(n+1)

- Read the *SSM Address* An+2, 2 plus the memory address An in which the last data word Dn is stored (only a monitoring check).

The sequence includes a nec*e*ssary overhead of two dummy reads (undefined data).

3.18 Selection of signals for monitoring with the oscilloscope

General

Although every effort has been made to minimize the distortion of mutual timing among the selected signals, a skew of around 2ns is expected (simulation results). For accessible inputs and outputs, the skew could be calibrated.

There are 24 signals in each group **A** and **B** - their lists are given below and include the associated selection code. The code is set with the *ScopeProbe Select* word (*SCOPE_SELECT*, address H”54”, write and read):

Data bits

[9..5]	Selection code for group **B**,
[4..0]	*Selection* code for group **A.**

The list of selected signals and their associated selection codes

Group A

Code	*Signal*
0	**BC** (LTU, delayed; also available in *Group B*)
1	**Orbit** (input from the CTP or the LTU emulator)
2	**Pre-pulse** (input from the CTP or the LTU emulator)
3	**L1, L1 Strobe**
4	**L2 Strobe**
5	**Pulser** (input)
6	**BUSY1** (input)
7	**BUSY2** (input)
8	**VME Read**
9	**VME Write**
10	**STROBE**
11	**DWB** (Device Wants Bus)
12	**MSTROBE**
13	**ADC** (input)
14	**Error Request** (set by *Error on Demand* option)
15	**TTC MS** (serialized TTCvi output, a header set to 1 and *8 most significant bits*)
16	*Leaky Bucket* **HALT**
17	**START ALL** signal (emulator, selected **START** option)
18	**L0** (output; also available in *Group B*)

Group B

Code	*Signal*
0	**BC** (LTU, delayed; also available in *Group A*)
1	**BC** (input)
2	**Orbit** (output)
3	**Pre-pulse** (output)
4	**L0** (output; also available in *Group A*)
5	**L1** (output)
6	**L1** Data

7	**L2** Data
8	**START** signal (selected **START** during emulation and if not **BUSY**)
9	LTU **BUSY** (output)
10	**L1 FIFO Empty**
11	**L2 FIFO Empty**
12	**L1 FIFO Nearly Full**
13	**L2 FIFO Nearly Full**
14	**Any Error**
15	**L1 Strobe** (from emulator, before delay)
16	L2 Data (from emulator, before delay)
17	**TTC LS** (serialized TTCvi output, a header set to 1 and *8 least significant bits*)

3.19 *Timing Test* mode

The *Timing Test* is a rather obscure option, only to be used when an LTU board in the *Stand-alone* mode “emulates” the CTP and drives (*via* its backplane connections) a number of other LTUs that operate in the *Global* mode (beam-test configuration, for example).

The ***Timing Test* write word** (*TIMING_TEST*, address H”6E”, data bit 0) sets or clears the *Timing Test* flag. When the flag is asserted, the emulator’s **L1 Data** signal continuously toggles - the condition required for the automatic setting of the **BC** delay in the connected LTUs.

The status of the flag is part of the *Emulation Status* word (section **3.13**).

The setting of the flag affects the LTU operation only when the board is in the *Stand-alone* mode.

The setting of the flag is, *strictly*, the system *Parameter* (section **3.4.2**).

3.20 *BC Status* word

The *BC Status* word (*BC_STATUS*, address H”31”) reads the status of the **BC Error** (data bit D[0]) and the **PLL Locked** (data bit D[1]) signals.

The **BC Error** signal is asserted if:

- the **BC** *input* is not present;
- the frequency of the **BC** input is outside the 38MHz to 42MHz range;
- a missing **BC** pulse has been detected;
- a glitch has occurred.

Note 1: The **BC Error** logic monitors the status of the **BC** *input*; the **BC** clock used on the LTU board is generated by the *LTU Logic* FPGA’s PLL that uses the delayed **BC** *input* as a reference.

The **BC** LED on the board’s front panel displays the status of the **BC Error** signal (section **3.3.1**).

The **PLL Locked** signal is asserted when the PLL of the *LTU Logic* FPGA is properly locked to the delayed **BC** input. The **LOCK** LED on the PCB displays the status of the signal (section **3.3.2**).

Note 2: The LTU can operate correctly only if the **BC Error** signal is off and the **PLL Locked** is on.

3.21 Master reset of the LTU logic

The signal that resets the logic on the LTU board (**MRESET** - *Master Reset*) is generated:

- by the VME SYSRESET signal;
- by the *Software Reset* command (*SOFT_RESET*, address H”A”, write only, dummy data, ~1µs wide pulse);
- by the MAX809SEURT reset circuit, upon power-on (~240ms wide pulse).

The signal resets all the state machines and some flags and registers.

3.22 Setting of the LTU mode

The *Mode* word (*MODE*, address “4D”, write and read) sets and reads back the LTU mode (data bit D[0]):

- 1 - *Stand-alone* mode.
- 0 - *Global* mode.

3.23 Setting of the *L1 Message* format

The control software can completely suspend the *L1 Message*; only enable the transmission of the first word; or allow the full message to be transmitted (see [1], section **3.6.2**).

The format is set with the *L1 Message Format* word (*L1_FORMAT*, address H”51”, write and read):

Format Data bit D[1]	*Enable* Data bit D[0]	*L1 Message* format
Don’t care	**0**	No L1 Message transmitted
0	**1**	One word only
1	**1**	All five words

3.24 LTU control and monitoring

3.24.1 *LTU DC voltage monitoring*

The DC voltage monitoring is done *via* the backplane, using the I^2C serial data transmission. The device used on the board is Philips PCF8591P ADC (8-bit resolution, 4 multiplexed channels). The four inputs monitor/digitize the following supply voltages:

- +5V “after the fuse”;
- +3.3V generated by the DC/DC converter;
- +1.5V generated by the voltage regulator;
- -5.2V generated by the voltage regulator and “shifted” into the positive ADC range using the 3.3V ADC reference voltage.

The following formulae should be used to convert the ADC reading ***N*** (0 - 255) into the corresponding voltage ***V***:

Channel 0 (+5V supply “after the fuse”; range: 0 - 6.5V):

$$\mathbf{V} = 23.725 \cdot \mathbf{N} \text{ [mV]}.$$

Channel 1 (+3.3V supply; range: 0 - 6.5V):

$$\mathbf{V} = 23.725 \cdot \mathbf{N} \text{ [mV]}.$$

Channel 2 (+1.5V supply; range: 0 - 3.3V):

$$\mathbf{V} = 12.941 \cdot \mathbf{N} \text{ [mV]}.$$

Channel 3 (-5V supply; range: -7.9V - +0.2V):

$$\mathbf{V} = -(7920 - 44 \cdot \mathbf{N}) \text{ [mV]}.$$

Comment: The +5V “*before the fuse*”, common to all the boards in the LTC, is monitored *via* the crate control connection.

All the voltages are connected via serial resistors in order to avoid consequences of a short circuit in the control logic.

The supply voltage monitoring is done primarily to help system debugging and maintenance; accurate “measurement” of the voltage levels is of secondary importance. The way the voltages are “connected” to the ADC, and the selection of digitizing range are very much the result of that approach. Examples : the 3.3V is derived from the +5V supply and a likely error could be a short to +5V, so the range for the +3.3V should include +5V; in a similar way, the range for 1.5V should include +3.3V, from which it is derived; *etc.* .

The connection to -5.2V is clamped with a diode to prevent the ADC input becoming negative; also, with the connection to the Vref (3.3V), the -5.2V rail could be charged to a positive voltage - this is prevented by another diode.

The ADC address inputs A[2..0] are connected to the *Board Address* dial. In that way, the same rotary switch sets both the VME and the control addresses. But: the pull-up resistors for the address bits must be connected to the +5V *before* the fuse; otherwise, if the fuse is blown, the I^2C address would be set to 0 (the +5v is compatible with the Max3000 I/O levels).

All the four LTU boards in the LTC shall be connected *via* a common I^2C bus. There shall be 6 I^2C branches to accommodate all the LTU boards in the system - one bus per crate. Another branch shall provide for the six FO boards in the CTP crate; the remaining 5 boards shall share the eighth brunch.

The eight I^2C branches shall be multiplexed, using bus multiplexers (8 channels, PCA9548) and connected, *via* the I2CPORTv2 adapter card, to the standard DB-25 parallel port of a PC.

The ADC chip is buffered on each LTU board with the P82B96TD buffer in order to extend the maximum length of the bus branch cable. The same buffer shall be used at the other end of the link.

The multiplexer (PCA9548) and buffers for the 8 branches could be located in the CTP crate (no interface to the VMEbus, just support and the power supply). From there, a link should be provided to the PC adapter board; the length of the link should be long enough to rich the PC in the control room (50-60m ?).

3.24.2 *Measurement of the board temperature*

The only I^2C-compatible device with a selectable address (LM75A) has the same “group” address as the ADC - they cannot “coexist” on the same I^2C branch. The use of an analog temperature converter is not possible since all the available inputs of the ADC are required. Because of it, the temperature of the LTU board is monitored by the VME processor. The thermal sensor, TC74, is also I^2C compatible, but the address is fixed. The temperature is read as an 8-bit word, the LSB corresponds to 1°C. The TC74 is connected to and controlled by the *VME Controller* FPGA.