Technical Information Manual

Revision n. 6

21 May 2012

MOD. V767

128 CH.
GENERAL PURPOSE
MULTIHIT TDC

NPO:
0000197:V767x.MUTx/06

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. DESCRIPTION

1.1. FUNCTIONAL DESCRIPTION

The Model V767 is a 1-unit wide VME 6U module that houses 128 independent Time to Digital Conversion channels. The unit houses 4 TDC chips, developed by CERN/ECP-MIC Division [1], thus called from now on the CERN/ECP-MIC TDCs [2].

The CERN/ECP-MIC TDC is a General Purpose time-to-digital converter, with 32 channels per chip. The integrated circuit is developed as a full custom device in CMOS 0.7 µm technology, and allows COMMON START operations with a typical bin size of 0.8 ns LSB. All channels can be enabled to the detection of rising and/or falling edges and for each channel there is a digital adjust for the zero-ing of any offsets and pedestals.

Two different versions are available: the **Mod. V767** and the **Mod. V767B**. The two versions differ only for the JAUX connector for the CERN V430 VMEbus crate: the Mod. V767 uses the P1, P2 VME connectors and the JAUX connector, while the Mod. V767B has only the P1 and P2 VME connectors.

The data acquisition can be programmed in "EVENTS" (TRIGGER MATCHING with a programmable time window or START GATING modes) or in "CONTINUOUS STORAGE". The management of overlapping triggers is also performed.

The COMMON STOP operation, though not existing in the chip itself, can be easily implemented on the board by assigning one of the 128 channels to a STOP signal and by an adequate programming of the trigger window.

The board houses a local memory FIFO buffer 32 kwords deep that can be readout via VME (as single data, Block Transfer and Chained Block Transfer) in a completely independent way from the acquisition itself.

The module programming is performed via a microcontroller that implements a high-level interface towards the User in order to mask the board and the TDCs' hardware.

The unit accepts the following CONTROL signals (ECL differential, 110 Ω) in common to all channels:

- START: a common START input;
- TRIGGER: a common TRIGGER input;
- RST: the RESET signal allows to clear the buffers and reset the TDCs; upon programming, it can also reset other registers of the unit;
- CLOCK: allows to provide an external Clock to the board.

Two special signals (“BUSY”, "DATA READY") are also available on the CONTROL bus. They are ECL signals that allow to obtain wired-OR Global BUSY and Global DATA READY signals. All the above described CONTROL lines can be terminated on-board via internal DIP-switches (termination must be done only on last board in a chain).

Five front panel LEDs show the status of the unit:

- DTACK lights up each time the module generates the VME signal DTACK;
- DATA READY lights up when the Data Ready condition occurs (see § 5.7);
- BUSY lights up when no more data can be written;
- TERM ON lights up when all the lines of the CONTROL bus are terminated;
- TERM OFF lights up when no line of the CONTROL bus is terminated.

The module houses a VME RORA INTERRUPTER[4]: the interrupt is generated on the occurrence (User programmable) of one of the following conditions:

- at least one complete event is present in the buffer;
- the buffer is not empty (at least one word is present in the buffer);
- the buffer is almost full (programmable).

The V767 Model uses the P1 and P2 connectors of VME and the auxiliary connector for the CERN V430 VMEbus crate (Jaux Dataway) [5].

The V767B Model uses the P1 and P2 connectors of VME only (i.e. it does not have the auxiliary connector for the CERN V430 VMEbus crate) and consequently hosts a DC-DC converter for the -5V power supply.

The module works in A24/A32 mode; the recognized Address Modifier codes are:

AM=%2F: A24 GEO access
AM=%3F: A24 supervisory block transfer (BLT)
AM=%3D: A24 supervisory data access
AM=%3C: A24 supervisory 64-bit block transfer (MBLT)
AM=%3B: A24 user block transfer (BLT)
AM=%39: A24 user data access
AM=%38: A24 user 64-bit block transfer (MBLT)
AM=%0F: A32 supervisory block transfer (BLT)
AM=%0D: A32 supervisory data access
AM=%0C: A32 supervisory 64-bit block transfer (MBLT)
AM=%0B: A32 user block transfer (BLT)
AM=%09: A32 user data access
AM=%08: A32 user 64-bit block transfer (MBLT)

The module's Base Address is fixed by 4 internal rotary switches housed on two piggy-back boards plugged into the main printed circuit board. The Base Address can be selected in the range:

%00 0000 <--> %FF 0000 A24 mode;
%0000 0000 <--> %FFFF 0000 A32 mode.

The internal registers are available in D16 mode only, while the data buffer is available in D32, BLT32 or MBLT64. The module supports also the Chained Block Transfer mechanism (CBLT) and the Multicast commands (MCST). Geographical address is also available.

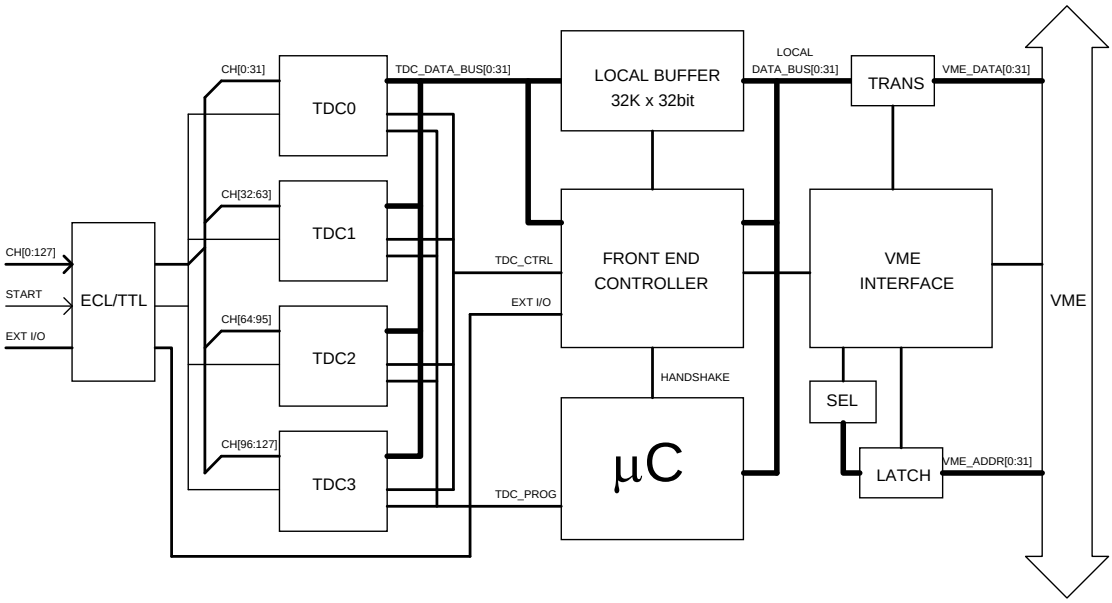


Fig. 1.1: Mod. V767 Block Diagram

1.2. PRINCIPLES OF OPERATION

The V767 operating principles are based on the functionality of the CERN/ECP-MIC TDC chip [2]. Five different module setups are selectable via software for different acquisition scenarios, namely:

- Stop Trigger Matching;
- Start Trigger Matching;
- Start Gating;
- Continuous Storage;
- Common Stop Emulation.

It is possible to switch from one operation setup to the other by simply resetting and reprogramming the module (with this operation the data in memory will be lost).

1.2.1. STOP TRIGGER MATCHING

An Event consists of the group of HIT signals that reach the enabled channels within a time window of programmable width and relative position with respect to the common TRIGGER signal. This operating setup does not foresee the use of the common START signal.

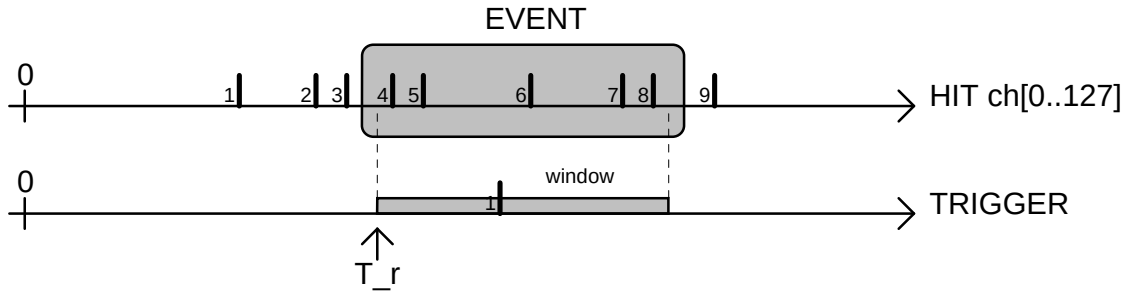


Fig. 1.2: Stop Trigger Matching Sequence

The stored data can be either an absolute time measurement (the zero time reference is represented by the RESET) or a relative time measurement referred to the beginning of the TRIGGER window (T_hit - T_r). The event is stored in the local buffer of the board with the following structure:

HEADER	event n. 0
DATUM n.1	HIT time(4)
DATUM n.2	HIT time(5)
DATUM n.3	HIT time(6)
DATUM n.4	HIT time(7)
DATUM n.5	HIT time(8)
EOB	num. of data read = 5

There is no limit to the number of words in an event. If the total rate of HIT signals is higher than the transfer rate of the data from the TDCs to the local buffer (or from the local buffer to the VME) an overflow condition will be flagged by the relevant bit in the Status Register (also by a front panel signal and LED). In this case there will be a data loss.

Please refer to § 5.9 to see a simple C-like language example to operate in Stop Trigger Matching mode.

1.2.2. START TRIGGER MATCHING

An Event consists of the group of START signals that arrive within a time window (of programmable width and relative position with respect to the common TRIGGER signal) and of the group of all HIT signals that reach the enabled channels, relevant to the START signals within the same time window.

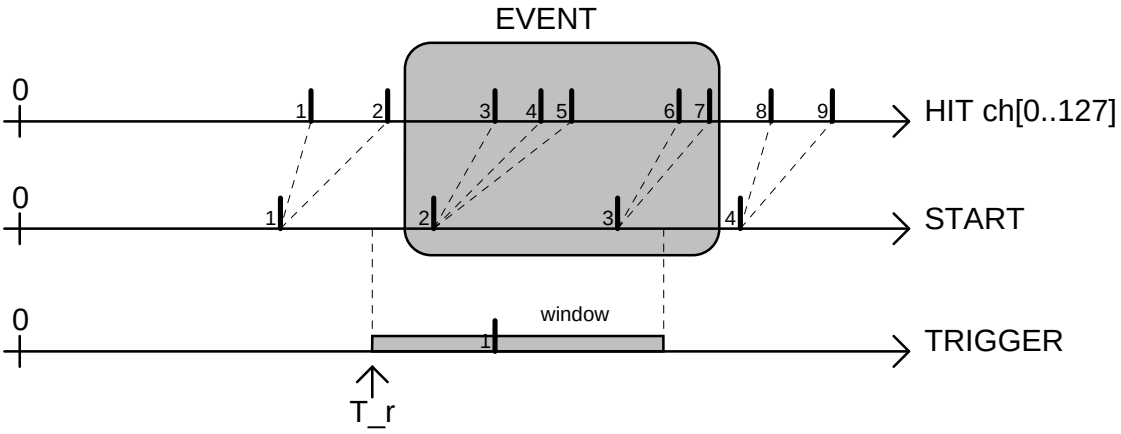


Fig. 1.3: Start Trigger Matching Sequence

The stored data can be either an absolute time measurement (the zero time reference is represented by the RESET) or a relative time measurement referred to the beginning of the TRIGGER Window for what concerns the START signals (T_start - T_r) and referred to the preceding START for what concerns the STOP signals (T_hit - T_start).

The TRIGGER signal is accepted at low resolution (bin size = Tclock) while the START and HIT signals are accepted at high resolution (bin size = Tclock/32). The event is stored in the local buffer of the board with the following structure:

HEADER	event n. 0
DATUM n.1	START time(2)
DATUM n.2	HIT time(3)
DATUM n.3	HIT time(4)
DATUM n.4	HIT time(5)
DATUM n.5	START time(3)
DATUM n.6	HIT time(6)
DATUM n.7	HIT time(7)
EOB	num. of data read = 7

It is also possible to disable the writing of the START times.

There is no limit to the number of words in an event. If the total rate of START and of HIT signals is higher than the transfer rate of the data from the TDCs to the local buffer (or from the local buffer to the VME) an overflow condition will be flagged by the relevant bit in the Status Register. In this case there will be a data loss.

Please refer to § 5.10 to see a simple C-like language example to operate in Start Trigger Matching mode.

1.2.3. START GATING

An Event consists of the group of HIT signals that reach the enabled channels within a time window that begins with the leading edge and ends with the trailing edge of the START signal. The common TRIGGER signal is not used.

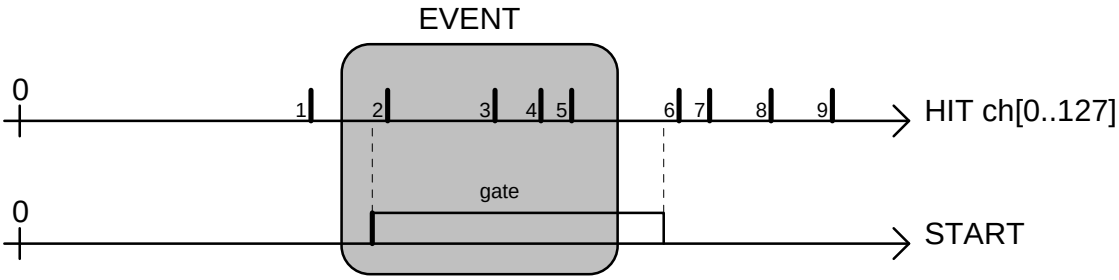


Fig. 1.4: Start Gating Sequence

The stored data can be either an absolute time measurement (the zero time reference is represented by the RESET) or a relative time measurement referred to the START (T_hit - T_start). The event is stored in the local buffer of the board with the following structure:

HEADER	event n. 0
DATUM n.1	START time
DATUM n.2	HIT time(2)
DATUM n.3	HIT time(3)
DATUM n.4	HIT time(4)
DATUM n.5	HIT time(5)
EOB	num. of data read = 5

It is also possible to disable the writing of the START times.

There is no limit to the number of words in an event. If the total rate of START and of HIT signals is higher than the transfer rate of the data from the TDCs to the local buffer (or from the local buffer to the VME) an overflow condition will be flagged by the relevant bit in the Status Register. In this case there will be a data loss.

In START GATING mode, before the acceptance of a new event it is necessary that all the data are transferred from the TDC chips to the Output Buffer. For this reason the module generates a BUSY signal. It is care of the User to inhibit the arrival of a new START signal when the BUSY signal is active.

Please refer to § 5.11 to see a simple C-like language example to operate in Start Gating mode.

1.2.4. CONTINUOUS STORAGE

In this acquisition mode the concept of "event" is meaningless, as no selection of a specific set of "valid data" (and thus to be stored in the local buffer) within the whole group of data is foreseen. All HIT signals that reach the enabled channels and all START signals lead to the storage of data in the local Buffer.

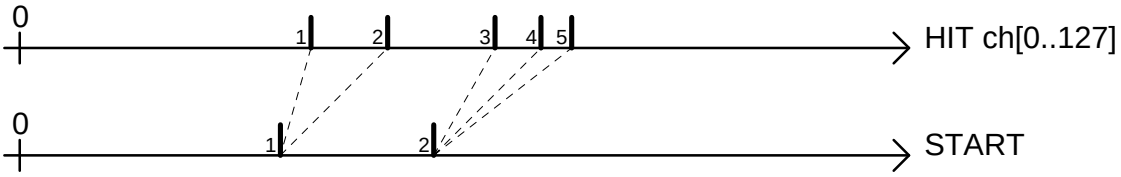


Fig. 1.5: Continuous Storage Sequence

The stored data can be either an absolute time measurement (the zero time reference is represented by the RESET) or a relative time measurement referred to the relevant START (T_hit - T_start).

The storage of data in the local Buffer does never foresee the writing of the control words (HEADER and EOB). The data are written in sequential order, reflecting the time evolution of the external signals:

DATUM n.1	START time(1)
DATUM n.2	HIT time(1)
DATUM n.3	HIT time(2)
DATUM n.4	START time(2)
DATUM n.5	HIT time(3)
DATUM n.6	HIT time(4)
DATUM n.7	HIT time(5)

It is also possible to disable the writing of the START times.

If the total rate of START and of HIT signals is higher than the transfer rate of the data from the TDCs to the local buffer (or from the local buffer to the VME) an overflow condition will be flagged by the relevant bit in the Status Register. In this case there will be a data loss.

Please refer to § 5.12 to see a simple C-like language example to operate in Continuous Storage mode.

1.2.5. COMMON STOP EMULATION

The COMMON STOP operation is not foreseen by the TDC chips. However it can be easily implemented with the following scheme:

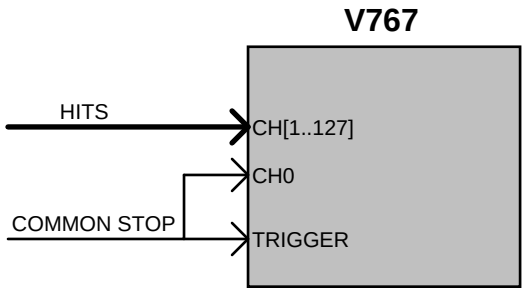


Fig. 1.6: Common Stop Emulation

by programming the board in STOP TRIGGER MATCHING mode, with the TRIGGER WINDOW ending with the occurrence of the TRIGGER/STOP signal (see Fig. 1.7).

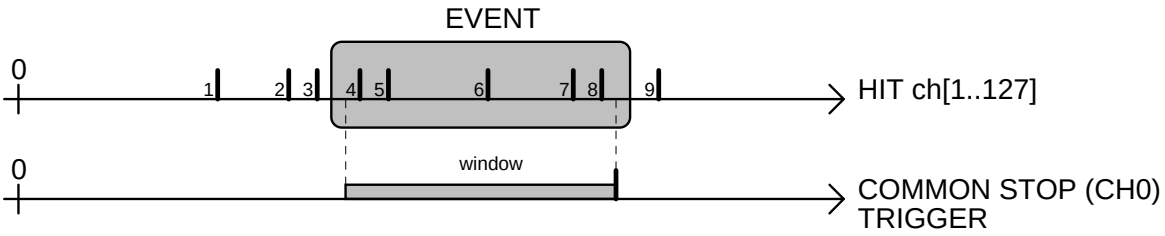


Fig. 1.7: Common Stop Sequence

The stored data consist in an absolute time measurement (the zero time reference is represented by the RESET) ; the timing information of the STOP (T_stop - T_hit) can be easily obtained by the software. The event is stored in the local buffer of the board with the following structure:

HEADER	event n. 1
DATUM n.1	HIT time(4)
DATUM n.2	HIT time(5)
DATUM n.3	HIT time(6)
DATUM n.4	HIT time(7)
DATUM n.5	HIT time(8)
DATUM n.6	STOP time
EOB	num. of data read = 6

2. SPECIFICATIONS

2.1. PACKAGING

1-unit wide VME unit. Height: 6U.

2.2. EXTERNAL COMPONENTS

(refer to fig. 2.1)

CONNECTORS
(refer to fig. 2.2 and fig. 2.3)

- N. 4, "INPUT A, B, C, D", Input connectors, Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins; for the 128 single channel inputs.
Connector A refers to Channels 0 to 31.
Connector B refers to Channels 32 to 63.
Connector C refers to Channels 64 to 95.
Connector D refers to Channels 96 to 127.

N.B.: The input connectors mate with cable connectors Robinson Nugent P50E-68-S-TG (3M 15-0209-5090-1) or P25E-68-S-TG, SOCKET-type; input cables must be flat cables, non-twisted type

- N. 1, “CONTROL”, input connector, Header 3M type, 8+8 pins, for the common control signals.

DISPLAYS

- N. 1, "DTACK", green LED, VME Selected. It lights up during a VME access.

- N. 1, "DATA READY", yellow LED. It lights up in occurrence of a Local/Global Data Ready.

- N. 1, "BUSY", red LED. It lights up in occurrence of a Local/Global BUSY.

- N. 1, "TERM ON", red LED. It lights up when the Control BUS has the termination ON.

- N. 1, "TERM OFF", red LED. It lights up when the Control BUS has the termination OFF.

2.3. INTERNAL COMPONENTS

(refer to fig. 2.4)

SWITCHES

- N. 4, rotary switches for the module's VME Base Address selection.

- N. 12, DIP switches for the Control Bus 110 Ohm terminations.

JUMPERS

- N. 1, "SRCCLK" for the External/Internal CLOCK selection.

- N. 5, JP2-JP6, for CAEN Internal Use only.

2.4. CHARACTERISTICS OF THE SIGNALS

- INPUT CHANNELS, START(1): Differential ECL level, 110 Ω impedance; min. width 10 ns.
- CLOCK(1): Differential ECL level, 110 Ω impedance; min. width 25 ns.
- TRIGGER(1): Rising-edge active, differential ECL level, 110 Ω impedance; min. width 25 ns.
- BUSY(1), DRDY(1): Active high, differential ECL level, 110 Ω impedance; min. width 25 ns.
- RESET(1): Active low, differential ECL level, 110 Ω impedance; min. width 25 ns.

(1) These signals are provided with DIP-switch insertable 110 Ω terminations, in order to connect more V767 modules in a daisy chain mode. The 110 Ω terminations must be inserted on the lines of the last module of the chain. All inputs are connected in a way that if the input connector is not inserted they are forced to a 0 logical level.

2.5. GENERAL

TDC CHIP SPECIFICATIONS

PARAMETER	***LO_RES***	***TYP***	**HI_RES**
NUMBER OF CHANNELS	128 + 1		
RESOLUTION (bin size)	Clock Period / 5bit		
	1.6 ns	0.8 ns	0.5 ns
DYNAMIC RANGE	20 bit		
	1.6 ms	0.8 ms	0.5 ms
DOUBLE HIT RESOLUTION	10 ns		
MEDIUM RATE (per channel, all channels simultaneous)	approx. 1 MHz		
DIFFERENTIAL NON LINEARITY	absolute timing: < 15 %		
	relative timing: < 1.5 %		
INTEGRAL NON LINERITY	0.3 ns		
STANDARD DEVIATION (estimated)	0.6		

VME BOARD SPECIFICATIONS

LOCAL BUFFER	FIFO 32K (TDC to FIFO trans. rate = 20 MHz)
TESTABILITY	data path and memory test (from VME)
CONTROL BUS TERMINATION	ON/OFF by dip-switch
CLOCK SOURCE	Internal (40 MHz) / External
TRIGGER WINDOW	software setting (from 1 clock cycle to full dynamic range)
ACQUISITION MODES	start or stop trigger matching, start gating, continuous storage
DATA READOUT	TRG MATCH or START GATING: sequential event RO CONTINOUS STORAGE: sequential data stream RO
VME READ OUT RATE	about 10 MHz
MAXIMUM EVENT SIZE	not defined

2.6. POWER REQUIREMENTS

Mod. V767		**Mod. V767B**	
+ 5 V	2.3 A	+ 5 V	3.5 A
– 5 V	1.5 A	-	-

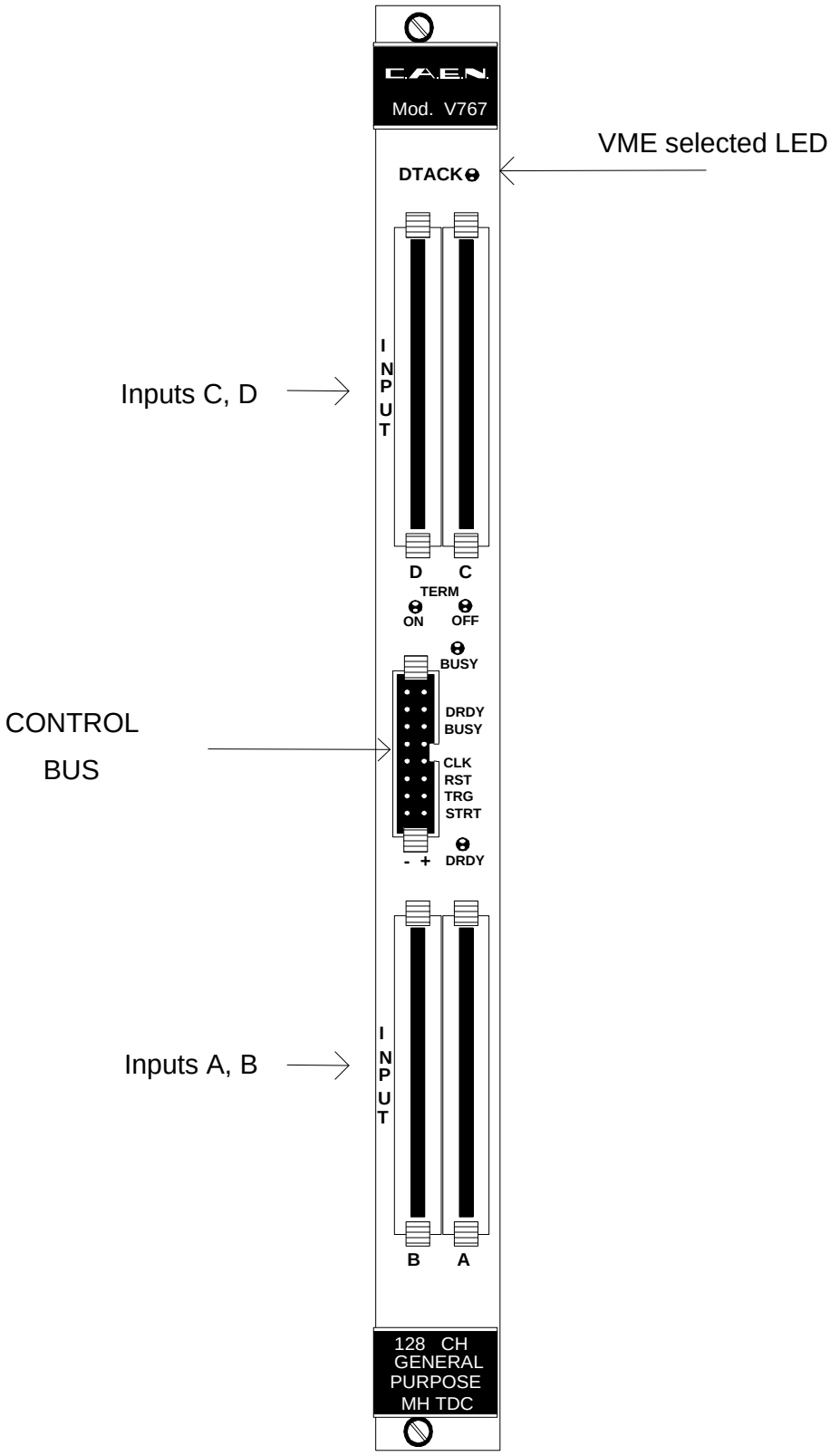


Fig. 2.1: Mod. V767 Front Panel

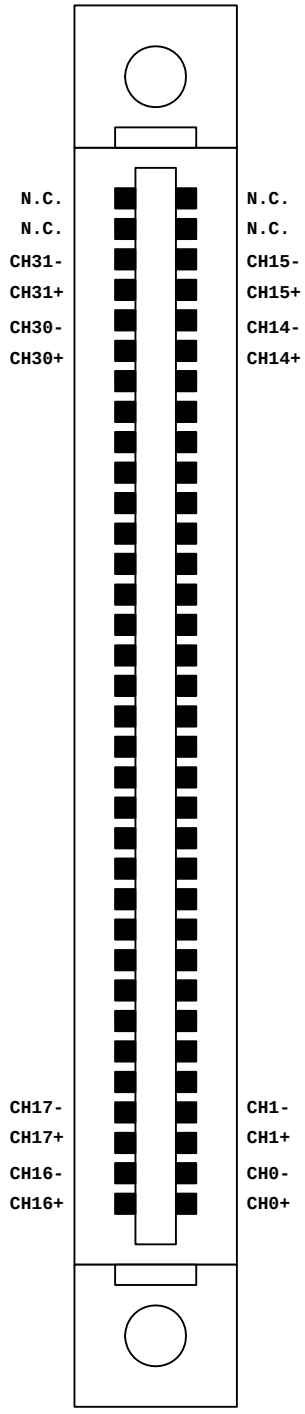


Fig. 2.2: INPUT connector pin assignment

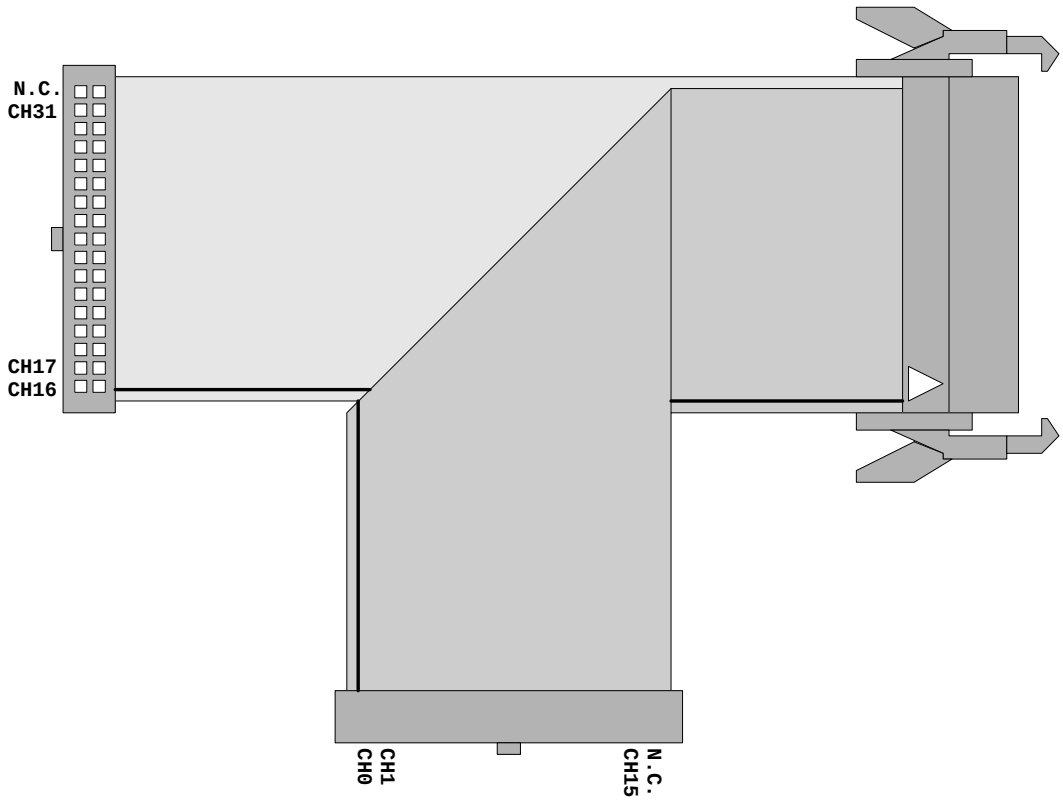
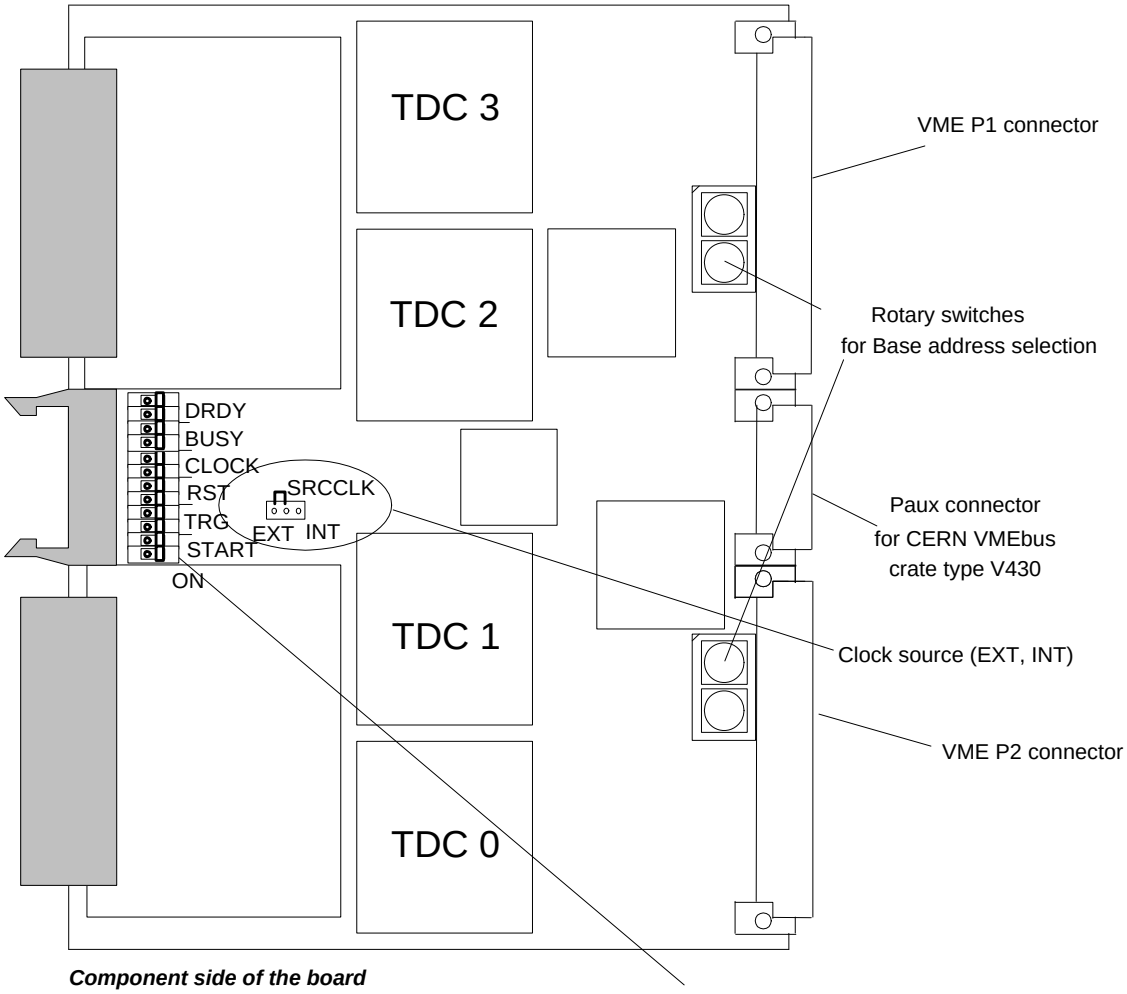


Fig. 2.3: INPUT connector cabling



Component side of the board

DIP Switches which allow to insert the 110 Ω terminations: for each signal there are two DIP switches which must be inserted both in oder to insert the 110 Ω termination on the relevant line.

Fig. 2.4: Mod. V767 Components Locations

3. VME INTERFACE

This module implements the CR/CSR space as required by the new VITA23 standard [6]. The CR (Configuration ROM) space contains read-only information on the hardware itself (e.g. manufacturer's ID, module characteristics, etc.). The CSR space contains the Control and Status Registers of the module (e.g. interrupt vector and level, CBLT/MCST base address, etc.).

3.1. ADDRESSING CAPABILITY

The board's addressing can be done in different ways, described here below.

3.1.1. ADDRESSING VIA BASE ADDRESS

The module works in A24/A32 mode. The Address Modifiers code recognized by the module are:

AM=%3F: A24 supervisory block transfer (BLT)
AM=%3D: A24 supervisory data access
AM=%3C: A24 supervisory 64-bit block transfer (MBLT)
AM=%3B: A24 user block transfer (BLT)
AM=%39: A24 user data access
AM=%38: A24 user 64-bit block transfer (MBLT)
AM=%0F: A32 supervisory block transfer (BLT)
AM=%0D: A32 supervisory data access
AM=%0C: A32 supervisory 64-bit block transfer (MBLT)
AM=%0B: A32 user block transfer (BLT)
AM=%09: A32 user data access
AM=%08: A32 user 64-bit block transfer (MBLT)

The module's Base Address is fixed by 4 internal rotary switches housed on two piggy-back boards plugged into the main printed circuit board (see Fig. 3.3). It is also possible to reassign via VME the Base address of the board (see Ref. [6]). The new address is lost at Power-Off and the rotary switch setting will be restored at Power-On or after a Reset (Bit Clear Register, see § 3.7).

The Base Address can be selected in the range:

% 00 0000	⇔	% FF 0000	A24 mode
% 0000 0000	⇔	% FFFF 0000	A32 mode

The Address Map of the page is shown in Table 3.1. The most relevant registers of the CR space are shown in Table 3.4.

3.1.2. ADDRESSING VIA GEOGRAPHICAL ADDRESS

The module works in A24 mode only. The Address Modifier code recognized by the module is:

AM=%2F: A24 GEO access

All the module's registers (exception made for the Output Buffer) can be accessed (with AM = %2F) via geographical addressing. The geographical address is automatically read out at each RESET from the SN5..SN1 lines of the Jaux connector: each slot of the VME crate is identified by the status of the SN5..SN1 lines (e.g. slot #5 will have these lines respectively at 00101) thus the module inserted in slot #5 will have a GEO address set to 00101.

The complete GEO address in A24 mode is:

A[31:24] don't care
A[23:19] GEO
A[18:0] offset

The Address Map of the page is shown in Table 3.1. The most relevant registers of the CR space are shown in Table 3.4.

In the case of the Mod. V767B where the SN5..SN1 lines are not available, the addressing via geographical address is not possible.
Although in the Mod. V767B it is possible to perform a write access to the GEO register for data identification during CBLT operation (see § 3.1.4, § 3.5 and § 5.16.1), it is incorrect to use the GEO register for addressing purposes when there is no JAUX.

3.1.3. BASE/GEO ADDRESSING EXAMPLE

The two following examples show a real situation with V767 boards inserted in a VME crate. The boards are addressed both via Base Addressing and via GEO Addressing.

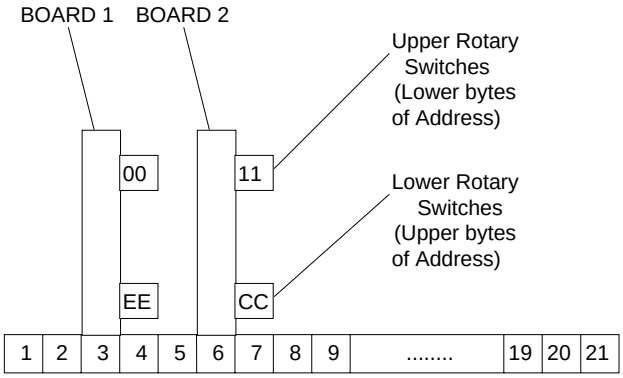


Fig. 3.1: Base/GEO Addressing Example 1

Board 1 access.
Base addressing A32: %EE000000 + Reg_Address
Base addressing A24: %000000 + Reg_Address
GEO addressing A24: %180000 + Reg_Address (Output Buffer excluded).

Board 2 access.
Base addressing A32: %CC110000 + Reg_Address
Base addressing A24: %110000 + Reg_Address
GEO addressing A24: %300000 + Reg_Address (Output Buffer excluded).

3.1.4. MCST/CBLT ADDRESSING

The module works in A32 mode only. The Address Modifiers code recognized by the module are:

AM=%0F: A32 supervisory block transfer (CBLT)
AM=%0D: A32 supervisory data access (MCST)
AM=%0B: A32 user block transfer (CBLT)
AM=%09: A32 user data access (MCST)

The boards can be accessed in Multicast Commands (MCST, see [6]) mode, that allows to write in the registers of several boards at the same time by accessing only once in A32 the MCST Base Address (see § 3.14). The boards can be accessed in Chained Block Transfer (CBLT, see [6]) mode, that allows to read out sequentially a certain number of contiguous boards in a VME crate. This access is allowed in BLT32 mode only on the MCST Base Address (see § 3.14).

N.B.: The Addresses used for MCST and CBLT operations are the same, i.e. throughout this User's Manual the "MCST Base Address" identifies the same Address, used both for MCST commands (in Write only) and the CBLT Readout (in Read only, for the Output Buffer only).

The MCST Base Address must be set in a different way respect to the ordinary Base Address: its most significant byte (i.e. bits 31 to 24) must be written in the MCST/CBLT Register and must be set in common to all boards (i.e. all boards must have the same setting of the MCST/CBLT Base Address on bits 31 to 24).

In CBLT and MCST operations, the IACKIN and IACKOUT VME lines are used for the control transfer from one board to the following. No empty slots must thus be left between the boards or, in alternative, empty slots can be left only in case VME crates with automatic IACKIN/IACKOUT short-circuiting are used.

Once the addresses have been set, the first and last board in a chain must have, respectively, only the FIRST_BOARD and only the LAST_BOARD bit set to 1 in the MCST_Control (see § 3.16). All intermediate boards which are active must have, on the contrary, both the FIRST_BOARD and the LAST_BOARD bits set to 1 in the MCST_Control (see § 3.16).

The Address Maps of the V767 module in MCST and CBLT modes are shown in Table 3.2 and Table 3.3.

The complete address in A32 mode is:

A[31:24] MCST Address
A[23:16] 00
A[15:0] offset

In CBLT operation the data coming from different boards are tagged with the HEADER and with the EOB words containing the GEO address in the 5 MSB (see also § 5.14). In the Mod. V767B (i.e. the version without the JAUX) it is up to the user to write the GEO address (this operation is possible only if the JAUX is not present) in the GEO register before executing the CBLT operation. If the GEO address is not written in the relevant register before performing the CBLT operation, it will not be possible to identify the module which the data are coming from.

3.1.5. MCST/CBLT ADDRESSING EXAMPLE

The following example shows a real situation with V767 boards inserted in a VME crate. The boards are addressed via MCST and CBLT Addressing.

The vme_write and vme_blt_read are two examples of user procedures that contain as parameters the complete Address (Base + offset), the data (either written or read), the addressing mode (A24 or A32) and the data mode (D16, D32 or D64).

vme_write(Address, Data, Addr_Mode, Data_mode);
vme_blt_read(Address, Buffer_pointer, Addr_Mode, Data_mode)

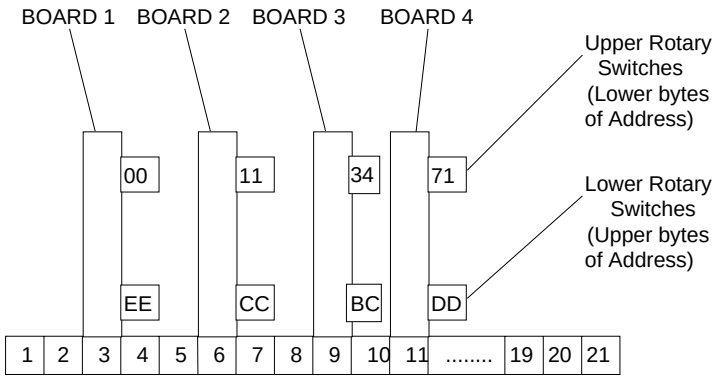


Fig. 3.2: MCST/CBLT Addressing Example

vme_write(0xEE000016, 0xAA, A32, D16) /* set (via Base Addressing) MCST Address = 0xAA */
(or **vme_write(**0x180016, 0xAA, A24, D16) /* set (via GEO Addr.) MCST Address = 0xAA */)

vme_write(0x11CC0016, 0xAA, A32, D16) /* set (via Base Addressing) MCST Address = 0xAA */
(or **vme_write(**0x300016, 0xAA, A24, D16) /* set (via GEO Addr.) MCST Address = 0xAA */)

vme_write(0x34BC0016, 0xAA, A32, D16) /* set (via Base Addressing) MCST Address = 0xAA */
(or **vme_write(**0x480016, 0xAA, A24, D16) /* set (via GEO Addr.) MCST Address = 0xAA */)

vme_write(0x71DD0016, 0xAA, A32, D16) /* set (via Base Addressing) MCST Address = 0xAA */
(or **vme_write(**0x510016, 0xAA, A24, D16) /* set (via GEO Addr.) MCST Address = 0xAA */)

vme_write(0xEE000020, 0x01, A32, D16) /* set (via Base Addressing) board 1 = First */
(or **vme_write(**0x180020, 0x01, A24, D16) /* set (via GEO Addr.) board 1 = First */)

vme_write(0x11CC0020, 0x03, A32, D16) /* set (via Base Addressing) board 2 = Active */
(or **vme_write(**300020, 0x03, A24, D16) /* set (via GEO Addr.) board 2 = Active */

vme_write(0x34BC0020, 0, A32, D16) /* set (via Base Addressing) board 3 = Inactive */
(or **vme_write(**0x480020, 0, A24, D16) /* set (via GEO Addr.) board 3 = Inactive */

vme_write(0x71DD0020, 0x02, A32, D16) /* set (via Base Addressing) board 4 = Last */
(or **vme_write(**510020, 0x02, A24, D16) /* set (via GEO Addr.) board 4 = Last */)

vme_write(0xAA000052, 0x1000, A32, D16) /* set boards 1, 2 & 4 in STOP_TRIG_MATCH
and Clear All Data in MCST addressing; Board 3
is Inactive and doesn't receive the setting opcode */

vme_write(0xAA00005A, 0, A32, D16) /* send a software trigger to boards 1, 2 & 4
in MCST addressing; Board 3 is Inactive and doesn't

receive the software trigger; boards 1, 2 & 4 will have an event in memory */

vme_blt_read(0xAA000000, buff_cont, A32, D32) /* read events in memory of boards 1, 2 & 4 in CBLT mode; Board 3 is Inactive and doesn't send data */

3.2. INTERRUPTER CAPABILITY

The Mod. V767 houses a RORA-type VME INTERRUPTER which is generated when the DATA READY is asserted TRUE.
The interrupt responds to 8 bit, 16 bit and 32 bit interrupt acknowledge cycles providing an 8-bit STATUS/ID on the VME data lines D00..D07 and removes its interrupt request when DATA READY is asserted FALSE (see § 3.2.4).

3.2.1. INTERRUPT STATUS/ID

The interrupt STATUS/ID is 8 bit wide, and it is contained in the Interrupt Vector Register<7..0>. The register is available at the VME address Base + % 000C.

3.2.2. INTERRUPT LEVEL

The interrupt level corresponds to the value stored in the Interrupt Level Register <2..0>. The register is available at the VME address Base + % 000A. A value of 0 in the Interrupt Level implies that no interrupt is generated.

3.2.3. INTERRUPT GENERATION

The Interrupt Generation occurs on DATA READY condition TRUE. The DATA READY condition can be programmed via VME (see § 4.9 and 5.7).

3.2.4. INTERRUPT REQUEST RELEASE

The V767 INTERRUPTER removes its Interrupt request when the DATA READY is asserted FALSE.
In particular:

1) if DATA READY = NOT EMPTY	→	the interrupt request is removed when the buffer is empty (see also § 4.9.2);
2) if DATA READY = ALM FULL	→	the interrupt request is removed when in the buffer there are less than *n* data (*n* is programmable; see also § 4.9.3);
3) if DATA READY = EVENT READY	→	the interrupt request is removed when in the buffer there is no more a complete event (see also § 4.9.1).

3.3. DATA TRANSFER CAPABILITY

The internal registers are accessible in D16 mode, unless otherwise specified. Access in D32, BLT32, MBLT64 and CBLT32 is available for the data buffer.

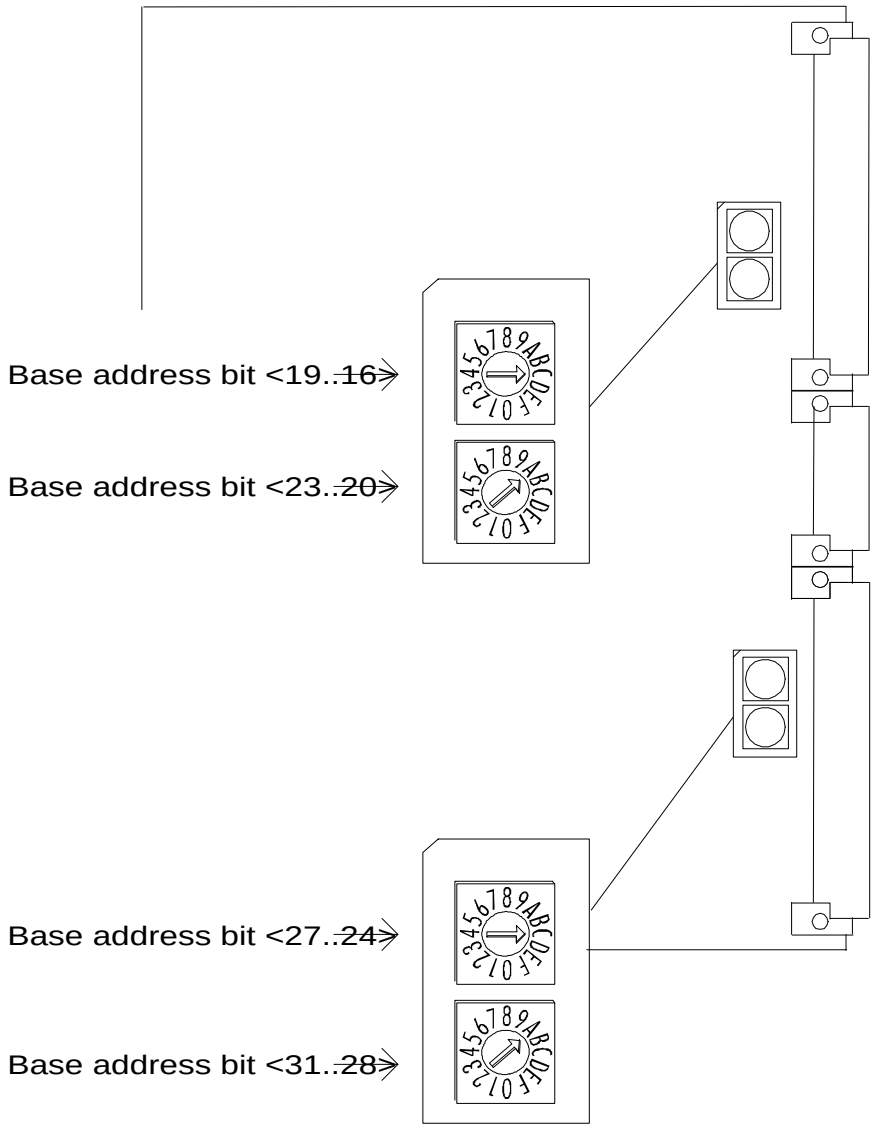


Fig. 3.3: Mod. V767 Base Address Setting

Table 3.1: Address Map for the Mod. V767

ADDRESS	REGISTER/CONTENT	TYPE
Base + %0000	Output Buffer (**)	read only
Base + %0004	Geographical Register	read only (*)
Base + %0006	Bit Set	read/write
Base + %0008	Bit Clear	read/write
Base + %000A	Interrupt Level	read/write
Base + %000C	Interrupt Vector	read/write
Base + %000E	Status_Register_1	read only
Base + %0010	Control_Register_1	read/write
Base + %0012	ADER_32	read/write
Base + %0014	ADER_24	read/write
Base + %0016	MCST Address	read/write
Base + %0018	Single Shot Reset	read/write
Base + %0020	MCST Control	read/write
Base + %0048	Status_Register_2	read only
Base + %004A	Control_Register_2	read/write
Base + %004C	Event Counter	read only
Base + %004E	Clear Event Counter	write only
Base + %0050	Opcode Handshake	read only
Base + %0052	Opcode Register	read/write
Base + %0054	Clear	write only
Base + %0056	Testword_H	write only
Base + %0058	Testword_L	write only
Base + %005A	Software Trigger	write only

(*): read/write in the Mod. V767B.

(**): from firmware rev 2.0 the Output Buffer is accessible also at Base + %2000 ÷ + %FFFC

Table 3.2: Address Map for the Mod. V767 in MCST operations

ADDRESS	REGISTER/CONTENT	TYPE
MCST Base + %0006	Bit Set	write only
MCST Base + %0008	Bit Clear	write only
MCST Base + %000A	Interrupt Level	write only
MCST Base + %000C	Interrupt Vector	write only
MCST Base + %0010	Control_Register_1	write only
MCST Base + %0018	Single Shot Reset	write only
MCST Base + %004A	Control_Register_2	write only
MCST Base + %004E	Clear Event Counter	write only
MCST Base + %0052	Opcode Register	write only
MCST Base + %0054	Clear	write only
MCST Base + %0056	Testword_H	write only
MCST Base + %0058	Testword_L	write only
MCST Base + %005A	Software Trigger	write only

Table 3.3: Address Map for the Mod. V767 in CBLT operations

ADDRESS	REGISTER/CONTENT	TYPE
MCST Base + %0000	Output Buffer	read only

Table 3.4: Address Map for the Mod. V767 CR space (most relevant registers)

ADDRESS	REGISTER	CONTENT
Base + %1026 to %102E	Manufacturer's ID	00-40-E6
Base + %1032 to %103E	Board ID	00-00-02-FF
Base + %104E	Revision ID	(*)
Base + %10E2	Slave Char. Parameters	5
Base + %10F6	Interrupter Capabilities	FE
Base + %1102	Function 0 Data Access Width	85
Base + %1106	Function 1 Data Access Width	85
Base + %1122 to %113E	Function 0 AM Code Mask	FF-00-00-00-00-00-FF-00
Base + %1142 to %115E	Function 1 AM Code Mask	FF-00-00-00-00-00-FF-00
Base + %1622 to %162E	Function 0 Address Decoder Mask	FF-FF-00-02
Base + %1632 to %163E	Function 1 Address Decoder Mask	00-FF-00-00
Base + %1F02, %1F06	Module Serial Number	Serial Number

* Depends on the featured firmware revision

3.4. OUTPUT BUFFER

(Base address + %0000 read only; from firmware rev 2.0 accessible also at Base + %2000 ÷ + %FFFC)

The output buffer contains the output data organised in 32-bit words. The data in the buffer are organised as follows:

A) In the case of *Start Trigger Matching*, *Stop Trigger Matching* (*Common Stop Emulation* included) and *Start Gating* mode the data in the buffer are organised in events.
Each event consists of (refer to Fig. 3.4):

- a **header** that contains the event number value;
- the **data words** containing the 20 bit converted time values, the channel number and an edge bit;
- an **End Of Block** (EOB) word containing the number of data words (i.e. the HEADER and EOB are not included) and a Status that is 1 if the TDC Chips have had an error, 0 otherwise.

B) In the case of *Continuous Storage* mode there are neither the header nor the EOB word, but only the data words.

In both cases, if a read access is performed to the buffer when it is empty, the readout will provide a NOT VALID DATUM arranged as shown in Fig. 3.4.

N.B.: in the Mod. V767B which does not have the JAUX, the GEO address contained in the HEADER and in the EOB must be written by the user via a write access to the relevant register (see § 3.5). If this operation is not performed, it will be not possible to identify which module the data are coming from when the CBLT access is used.

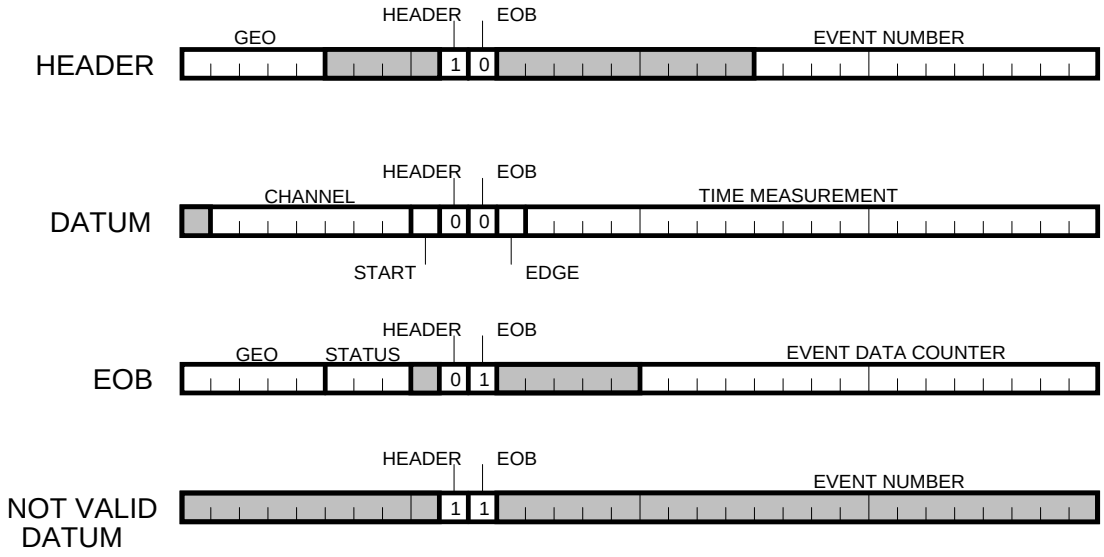


Fig. 3.4: Mod. V767 Output Buffer Data Structure

3.5. GEOGRAPHICAL REGISTER

(Base address + %0004 read only; read/write in the Mod. V767B)

This register contains the geographical address of the module read by the JAUX. The register content is the following:

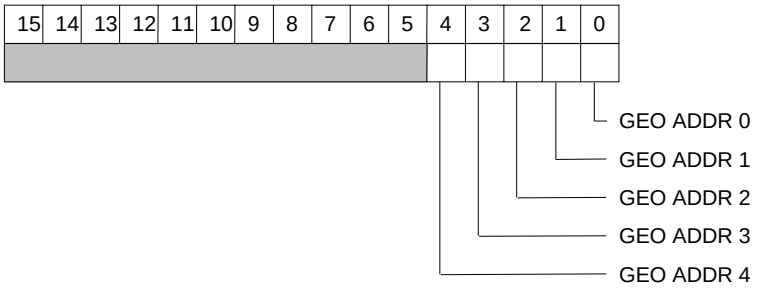


Fig. 3.5: GEO Address Register

GEO[4..0] corresponds to A23..A19 in the address space of the CR/CSR area: each slot has a relevant number whose binary encoding consists in the GEO ADDR 4 to 0.

In the Mod. V767B which does not have the JAUX, this register can be also written. The bits of this register are set to 1 by default. In CBLT operation it is up to the user to write the correct GEO address of the module in this register before operating so that the GEO address will be contained in the HEADER and the EOB words for data identification (see also § 5.14 and § 5.16.1). Conversely, the use of a write access to the GEO register for addressing purposes is completely incorrect.

N.B.: in the Mod. V767 a write access to the GEO register can be also performed, but is completely ineffective, i.e. the value will not be written in the register.

3.6. BIT SET REGISTER

(Base address + %0006 read/write)

This register allows to establish a RESET logic of the module and to enable the change of the base address via VME. A write access with the bits to 1 sets the relevant bits to 1 in the register. A write access with the bits set to 0 does NOT clear the register content: in order to clear the register content, the Bit Clear Register must be used (see § 3.7). A read access returns the status of this register. The register content is the following:

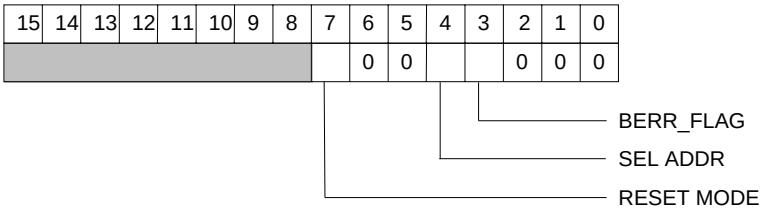


Fig. 3.6: Bit Set Register

BERR FLAG Bus Error Flag Bit. Meaningful in BLT/CBLT modes only.
= 0 Board has not generated a Bus Error;
= 1 Board has generated a Bus Error.

SELECT ADDRESS Select Address bit.
= 0 Base addresses are selected via Rotary Switch (power-on default);
= 1 Base addresses are selected via internal ADER registers.

RESET MODE: Sets the module to a permanent RESET status. The RESET is released only via write access with the relevant bit set to 1 in the Bit Clear Register, see § 3.7.

N.B.: After a Power-On or a VME Reset or a Sys-Res command, it is necessary to issue a command which makes the module wait for about 2 s to allow its initialisation (see, for example, the C-like language software example in § 5.9).

3.7. BIT CLEAR REGISTER

(Base address + %0008 read/write)

This register allows to clear the bits in the above described Bit Set Register. A write access with the bits to 1 sets the relevant bits to 1 in the register and clears the register content. A write access with the bits set to 0 does NOT clear the register content. The structure of the register is identical to the Bit Set Register. A read access returns the status of this register.

3.8. INTERRUPT LEVEL REGISTER

(Base address + %000A read/write)

The 3 LSB of this register contain the value of the interrupt level (Bits 3 to 15 are meaningless).

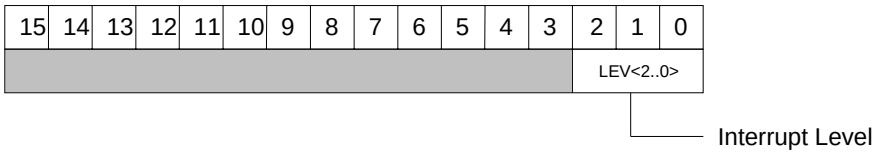


Fig. 3.7: Interrupt Level Register

3.9. INTERRUPT VECTOR REGISTER

(Base address + %000C read/write)

This register contains the value of the Interrupt STATUS/ID that the V767 INTERRUPTER places on the VME data bus during the Interrupt Acknowledge cycle (Bits 8 to 15 are meaningless).

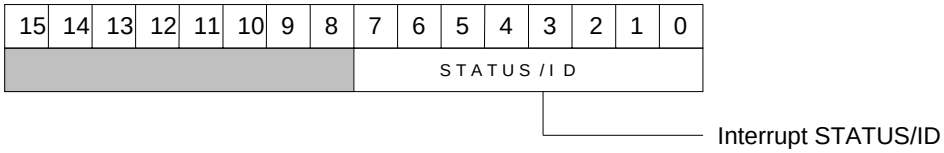


Fig. 3.8: Interrupt Vector Register

3.10. STATUS REGISTER 1

(Base address + %000E read only)

This register contains information on the status of the module.

NO TERM and TERM OK refer to the termination of the lines in the front panel CONTROL bus: the last module in a chain controlled via the front panel CONTROL connector must have this termination ON, while all the others must have it OFF. The insertion or removal of the termination is performed via internal DIP switches (see Fig. 2.4).

The BUSY and DATA READY signals are available both for the individually addressed module and as a global readout of a system of many units connected together via the CONTROL BUS.

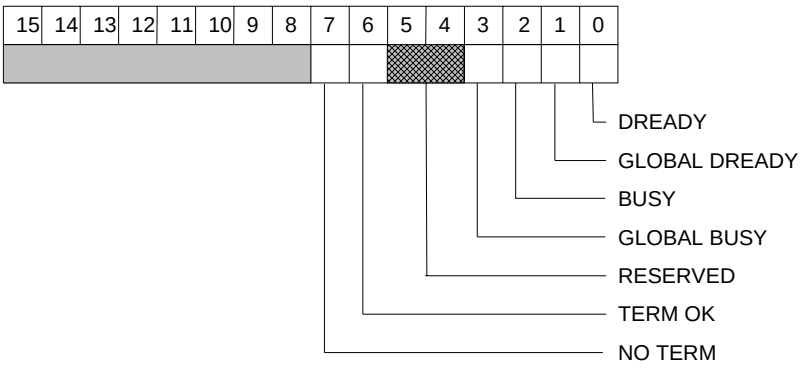


Fig. 3.9: Status Register 1

DREADY: Indicates that the Output Buffer has data.
= 0 No Data Ready.
= 1 Data Ready.

GLOBAL DREADY Indicates that at least a module in a chain has data.
= 0 No Module has Data Ready.
= 1 At least a Module has Data Ready.

BUSY Indicates either that a conversion is in progress, or the Output Buffer is full, or that the board is in TEST mode; goes low when the Output Buffer is not full and the module is in Normal Mode.
= 0 Module not Busy.
= 1 Module Busy.

GLOBAL BUSY Indicates that at least a module in a chain is BUSY.
= 0 No Module is Busy.
= 1 At least a Module is Busy.

TERM OK Termination ON bit.
= 0 Control Bus Termination is ON.

NO TERM Termination OFF bit.
= 0 Control Bus Termination is OFF.

3.11. CONTROL REGISTER 1

(Base address + %0010 read/write)

This register allows to perform some general settings of the module.

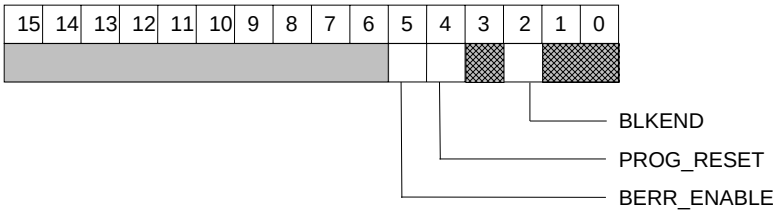


Fig. 3.10: Control Register 1

BLK_END End of Block bit. Used in Block Transfer mode only.
= 0 The module sends to the CPU all the requested data; when the Output Buffer is empty it will send not valid data.
If BERR_VME is enabled, a Bus Error is generated with the readout of the last word in the Output Buffer.
= 1 The module sends to the CPU all data until the first EOB word (End of first event); afterwards it will send not valid data.
If BERR_VME is enabled, a Bus Error is generated with the readout of the EOB word.

PROGRESET Programmable Reset Mode setting bit.
= 0 the front panel RESET acts only on data;
= 1 the front panel RESET acts on all the module, including the Front End.

BERR_EN Bus Error enable bit. Used in Block Transfer mode only.

= 1 BERR_VME line enabled.

(Bits 9 to 15 are meaningless).

3.12. ADDRESS DECODER REGISTER 32

(Base address + %0012 read/write)

This register contains the A31..A24 bits of the address of the module: it can be set via VME for a relocation of the Base Address of the module, as described in [6]. The register content is the following:

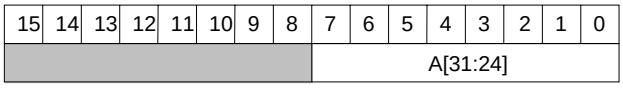


Fig. 3.11: ADER 32 Register

3.13. ADDRESS DECODER REGISTER 24

(Base address + %0014 read/write)

This register contains the A23..A16 bits of the address of the module: it can be set via VME for a relocation of the Base Address of the module, as described in [6]. The register content is the following:

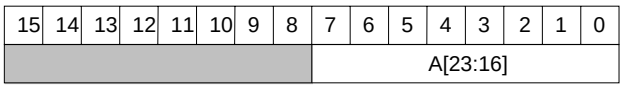


Fig. 3.12: ADER 24 Register

3.14. MCST ADDRESS REGISTER

(Base address + %0016 read/write)

This register contains the Multicast (MCST) address of the module, set via VME. The register content is the following:

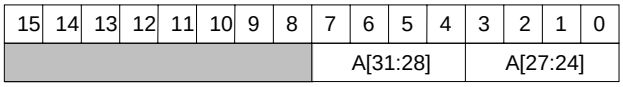


Fig. 3.13: MCST Address Register

3.15. SINGLE SHOT RESET REGISTER

(Base address + %0018 read/write)

A dummy access to this register allows to generate a single shot RESET of the module. Once issued, the module Front End is reset.

N.B.: After a Power-On or a VME Reset or a Sys-Res command, it is necessary to issue a command which makes the module wait for about 2 s to allow its initialisation (see, for example, the C-like language software example in § 5.9).

3.16. MCST CONTROL REGISTER

(Base address + %0020 read/write)

This register allows to perform some general settings of the module.

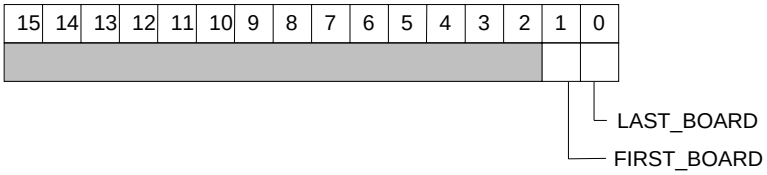


Fig. 3.14: Control Register 1

LAST_BOARD Last Board flag bit **(valid in CBLT and MCST modes only)**

FIRST_BOARD First Board flag bit **(valid in CBLT and MCST modes only)**

The status of the boards according to the bit value is the following:

Board Status	FIRST_BOARD bit	LAST_BOARD bit
Board disabled in CBLT or MCST chain	0	0
First board in CBLT or MCST chain	1	0
Last board in CBLT or MCST chain	0	1
Active intermediate board in CBLT or MCST chain (neither first nor last)	1	1

(Bits 2 to 15 are meaningless).

3.17. STATUS REGISTER 2

(Base address + %0048 read only)

This register contains further information on the status of the module buffer and TDC errors.

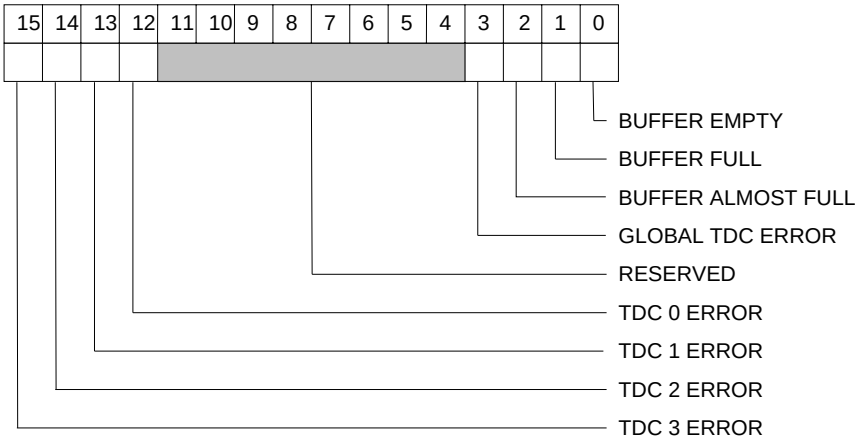


Fig. 3.15: Status Register 2

BUFFER EMPTY: Indicates if the Output Buffer is empty.
= 0 Buffer Not Empty.
= 1 Buffer Empty.

BUFFER FULL: Indicates if the Output Buffer is full.
= 0 Buffer Not Full.
= 1 Buffer Full.

BUFF. ALMOST FULL: Indicates if the Output Buffer is almost full. The selection of the Almost Full status is programmable via OPCODE (see § 4.9.5).
= 0 Buffer Not Almost Full.
= 1 Buffer Almost Full.

GLOBAL TDC ERROR: Indicates that at least a TDC chip has an error (See § 4.10.1).
= 0 No TDC chip has an error.
= 1 At least a TDC chip has an error.

TDC 0..3 ERROR: Indicates that a selected (0..3) TDC chip has an error (See § 4.10.1).

= 0 The selected TDC chip has no errors.
= 1 The selected TDC chip has an error.

3.18. CONTROL REGISTER 2

(Base address + %004A read only, bit 4 read/write)

This register allows to perform a reading of some settings of the module.

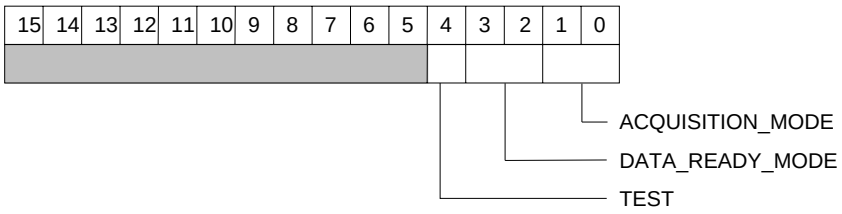


Fig. 3.16: Control Register 2

ACQUISITION MODE: Reads the status of the acquisition mode set via OPCODE without using the OPCODE reading (see § 4.3.5). The two ACQ_MODE bits of this register are related to the Acquisition Mode according to the following:

ACQ_MODE = (0, 0) ⇒ Stop Trigger Matching Mode;
ACQ_MODE = (0, 1) ⇒ Start Trigger Matching Mode;
ACQ_MODE = (1, 0) ⇒ Start Gating Mode;
ACQ_MODE = (1, 1) ⇒ Continuous Storage Mode.

DATA READY MODE: Reads the status of the data in the buffer. The two DR_MODE bits of this register are related to the Data Ready Mode according to the following:

DR_MODE = (0, 0) ⇒ Buffer contains a complete event;
DR_MODE = (0, 1) ⇒ Buffer almost full;
DR_MODE = (1, 0) ⇒ Buffer not empty;
DR_MODE = (1, 1) ⇒ Meaningless.

TEST: Test bit.
= 0 Normal mode;
= 1 Test mode.

3.19. EVENT COUNTER REGISTER

(Base address + %004C read only)

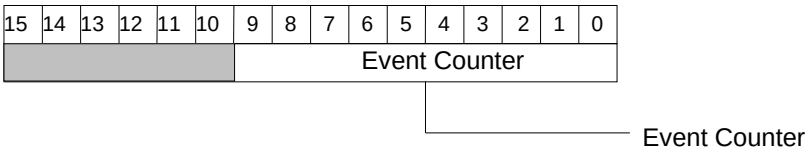


Fig. 3.17: Event Counter

EVENT CNT: 10 bit Event Counter.
It counts the number of events transferred from the TDCs to the output buffer.

3.20. CLEAR EVENT COUNTER REGISTER

(Base address + %004E write only)

A VME access (read or write) to this location clears the Event Counter.

3.21. OPCODE HANDSHAKE REGISTER

(Base address + %0050 read only)

The Opcode Handshake Register is used for the Handshake Protocol between the VME and the microcontroller. It uses only 2 bits: READ OK and WRITE OK.

All read and write operations with the Opcode Register can be performed, respectively, when the bit RO or WO is set. See § 4 for the usage of this register.

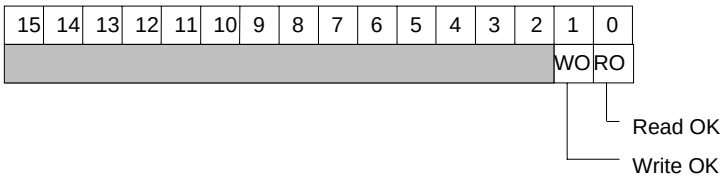


Fig. 3.18: Opcode Handshake Register

N.B.: because of V767 module's internal delays it is necessary to insert a 10-ms delay in the software after the check of the RO/WO bit, i.e. before performing the next R/W operation on the Opcode Register.

3.22. OPCODE REGISTER

(Base address + %0052 read/write)

The Opcode Register is used to send instructions to the microcontroller via 16-bit OPCODE setup words. The usage of this register is fully described in § 4.

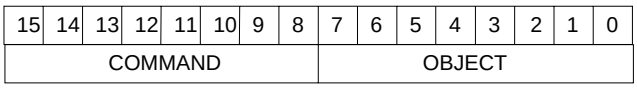


Fig. 3.19: Opcode Register

3.23. CLEAR REGISTER

(Base address + %0054 write only)

A VME access (read or write) to this location causes the following:

1. the TDCs are cleared;
2. the output buffer is cleared;
3. the readout controller is reset;
4. the Event counter is set to 0.

The same actions are performed at Power-ON and if the VME signal SYSRES is active.

3.24. TESTWORD_HIGH REGISTER

(Base address + %0056 write only)

This register allows to set a 32-bit word that is used for the memory test. The higher 16 bits are set via this register, while the lower 16 bits are set via the TESTWORD_LOW register.
The usage of this couple of registers is the following:

a) set the module in test mode;
b) write the 16 least significant bits in the TESTWORD_LOW register;
c) write the 16 most significant bits in the TESTWORD_HIGH register;

With the latter operation, the 32-bit pattern is transferred to the memory. If operations b) and c) are inverted the content of the 16 least significant bits could be meaningless.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TESTWORD[31:16]															

Fig. 3.20: TESTWORD_HIGH Register

3.25. TESTWORD_LOW REGISTER

(Base address + %0058 write only)

This register allows to set the lower 16 bits of the Test Word (see above).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TESTWORD[15:0]															

Fig. 3.21: TESTWORD_LOW Register

3.26. SOFTWARE TRIGGER REGISTER

(Base address + %005A write only)

A VME access (read or write) to this location generates a software trigger.

4. OPERATING CODES

4.1. PROGRAMMING CAPABILITY

The module programming is performed by means of an on-board microcontroller. The User sends and receives instructions and data to/from the microcontroller via 16-bit OPCODE setup words. The handshake is the following:

Write Operation:
- the VME (master) tests the WRITE_OK bit in the Opcode Handshake Register (see § 3.21); if the WO bit is set to 1, the VME can write a datum;
- the WO bit is automatically reset after the datum is written and is set back to 1 when the datum has been acquired by the server;
- when the WO bit is set back to 1, the VME can write another datum.

Read Operation;
- a valid datum can be read via VME only if the READ_OK (RO) bit in the Opcode Handshake Register (see § 3.21) is set to 1;
- the RO bit is automatically reset after the datum is read out and is set back to 1 when a new datum is ready to be read out;
- when the RO bit is set back to 1, the VME can read another datum.

N.B.: because of V767 module's internal delays it is necessary to insert a 10-ms delay in the software after the check of the RO/WO bit, i.e. before performing the next R/W operation on the Opcode Register.

In general, three different types of operation can be performed:

1. write an opcode,
2. write an operand,
3. read an operand.

By default, at power-on or after a VME reset, the WO bit of the Handshake Register is set to 1 and the RO to 0: actually, the first operation to be performed is always the write operation of an opcode (operation of type 1).
Conversely, the operation of type 2 must be always preceded by the write operation of an opcode which requires to write an operand: this is case, for example, of the opcode 'SET_WIN_WIDTH' which requires to input the operand '*width of the window*'. Similarly, the operation of type 3 must be always preceded by a write operation of an opcode which issues a read access: actually, this makes the RO bit of the Handshake Register to be set to 1 in order to allow the following read operation. For example, the write operation of the opcode 'READ_WIN_WIDTH' makes the RO bit of the Handshake Register be set to 1 and in this way allows the following read operation.

Before any read/write operation, it is necessary to run a while loop in order to wait until the module is ready. As a consequence, any R/W operation is in fact constituted by the following blocks:

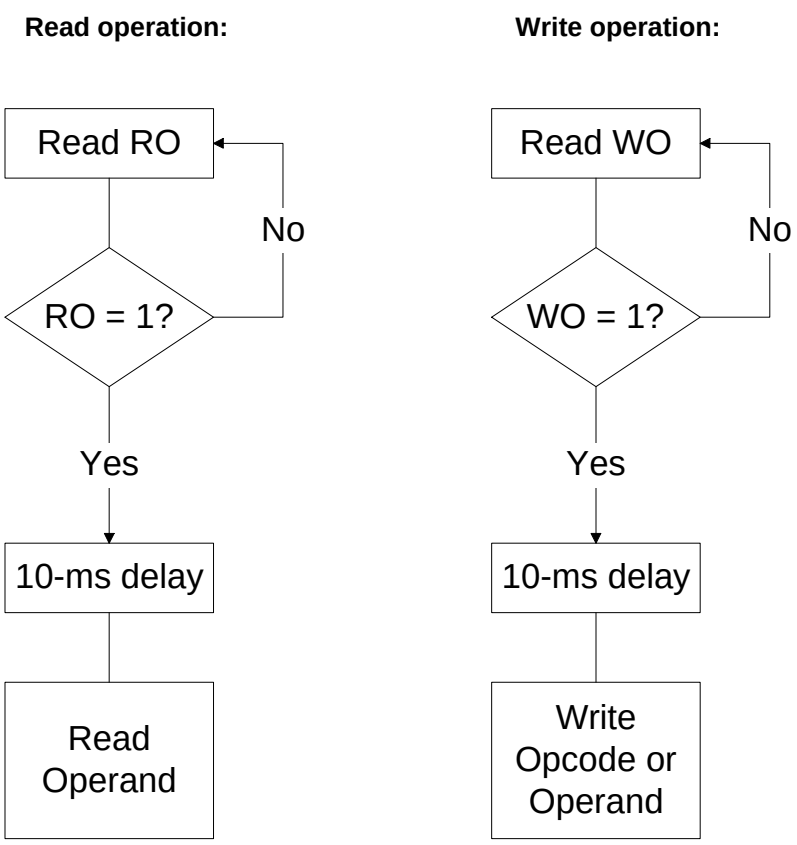


Fig. 4.1: R/W operations: software logic blocks

The following two examples in C-like language show, respectively, how to write an opcode or an operand and how to read an operand.

```
/*****************************************************************/
/* V767 PROGRAMMING ROUTINES                                     */
/*****************************************************************/

#define TIMEOUT  1000000

/*****************************************************************/
/* this function allows to write an opcode or an operand         */
/*****************************************************************/
int write_op (unsigned short data)

  {
  unsigned short *opc_hs, *opc_reg;
  unsigned short rdata;
  int time=0;

  /* sets V767 registers addresses */
  opc_hs = (unsigned short *)(V767_BASE_ADDRESS + 0x50);
  opc_reg = (unsigned short *)(V767_BASE_ADDRESS + 0x52);
```

```
  /* checks the Write OK bit */
  do
    {
    rdata = *opc_hs;
    time++;
    }
  while( (rdata != 0x02) && (time < TIMEOUT) );

  if(time == TIMEOUT)  /* a time out error is occurred */
    {
    printf("Timeout error\n");
    return(-1);
    }

  delay(10);  /* waits for 10 ms */

  *opc_reg = data;  /* vme write access to opcode register */

  return(0);
  }


/*****************************************************************/
/* this function allows to read an operand                       */
/*****************************************************************/
int read_op (unsigned short *data)

  {
  unsigned short *opc_hs, *opc_reg;
  unsigned short rdata;
  int time=0;

  /* sets V767 registers addresses */
  opc_hs = (unsigned short *)(V767_BASE_ADDRESS + 0x50);
  opc_reg = (unsigned short *)(V767_BASE_ADDRESS + 0x52);

  /* waits until a new read/write opcode operation is ready */
  do
    {
    rdata = *opc_hs;
    time++;
    }
  while( (rdata != 0x01) && (time < TIMEOUT) );

  if(time == TIMEOUT)  /* a time out error is occurred */
    {
    printf("Timeout error\n");
    return(-1);
    }

  delay(10);  /* waits for 10 ms */

  *data = *opc_reg;  /* vme read access to opcode register */

  return(0);
  }
```

The OPCODE setup words have the following format:

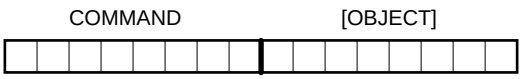


Fig. 4.2: Mod. V767 OPCODE Word

The COMMAND field specifies the operation to perform, while the OBJECT field (when required) specifies the object on which the operation must be performed (e.g. the channel number). If the object refers to the channel number (OBJ = nn in Table 4.1), it can vary from 0 to 7F; if it refers to the TDC number (OBJ = c in Table 4.1), it can vary from 0 to 3.
When the operation does not foresee an object, the OBJECT field is meaningless.

The communication with the microcontroller begins always by sending an OPCODE; if no operands are foreseen (nR = 0 and nW = 0) the cycle ends, otherwise the microcontroller remains in a wait status until the User hasn't read or written all the foreseen operands.

The following Table 3.1 contains, for each OPCODE, the symbolic name, the performed operation, the number of written and read operands and the number of significant bits.

Table 4.1: OPCODE Words for the Mod. V767

COM	OBJ	CODE	SYMBOLIC	OPERATION	nW	nR	nbit
TEST							
01	-	01xx	EN_MEM_TEST	enable memory test mode	-	-	-
02	-	02xx	DIS_MEM_TEST	disable memory test mode	-	-	-
03	-	03xx	READ_MEM_TEST	read memory test (on/off)	-	1	1
ACQUISITION MODE							
10	-	10xx	STOP_MATCH	set stop trigger matching	-	-	-
11	-	11xx	START_MATCH	set start trigger matching	-	-	-
12	-	12xx	START_GAT	set start gating	-	-	-
13	-	13xx	CONT_STO	set continuous storage	-	-	-
14	-	14xx	READ_ACQ_MOD	read acquisition mode	-	1	2
15	-	15xx	LOAD_DEF_CONFIG	load default configuration	-	-	-
16	-	16xx	SAVE_CONFIG	save User configuration	-	-	-
17	-	17xx	LOAD_CONFIG	load User configuration	-	-	-
18	-	18xx	ENABLE_AUTO_LOAD	enable auto load	-	-	-
19	-	19xx	DISABL_AUTO_LOAD	disable auto load	-	-	-
1A	-	1Axx	READ_AUTO_LOAD	read auto load	-	1	1
CHANNEL ENABLE							
20	nn	20nn	EN_CHANNEL	enable channel **nn**	-	-	-
21	nn	21nn	DIS_CHANNEL	disable channel **nn**	-	-	-
22	nn	22nn	READ_STAT_CH	read status channel **nn**	-	1	1
23	-	23xx	EN_ALL_CH	enable all channels	-	-	-
24	-	24xx	DIS_ALL_CH	disable all channels	-	-	-
25	-	25xx	WRITE_EN_PATTERN	write enable pattern for channels	8	-	16
26	-	26xx	READ_EN_PATTERN	read enable pattern for channels	-	8	16
TRIGGER							
30	-	30xx	SET_WIN_WIDTH	set window width	1	-	16
31	-	31xx	READ_WIN_WIDTH	read window width	-	1	16
32	-	32xx	SET_WIN_OFFS	set window offset	1	-	16
33	-	33xx	READ_WIN_OFFS	read window offset	-	1	16
34	-	34xx	SET_TRG_LAT	set trigger latency	1	-	16
35	-	35xx	READ_TRG_LAT	read trigger latency	-	1	16
36	-	36xx	EN_SUB_TRG	enable subtraction of trigger time	-	-	-
37	-	37xx	DIS_SUB_TRG	disable subtraction of trigger time	-	-	-
38	-	38xx	EN_OVL_TRG	enable overlapping triggers	-	-	-
39	-	39xx	DIS_OVL_TRG	disable overlapping triggers	-	-	-
3A	-	3Axx	READ_TRG_CONF	read trigger configuration	-	1	2
START							
40	-	40xx	EN_RO_START	enable read out of start time	-	-	-
41	-	41xx	EN_RO_ALL_START	enable read out of 4 start time	-	-	-
42	-	42xx	DIS_RO_START	disable readout of start time	-	-	-
43	-	43xx	EN_SUB_START	enable subtraction of start time	-	-	-
44	-	44xx	DIS_SUB_START	disable subtraction of start time	-	-	-
45	-	45xx	EN_EMPTY_START	enable empty start	-	-	-
46	-	46xx	DIS_EMPTY_START	disable empty start	-	-	-
47	-	47xx	READ_START_CONF	read start configuration	-	1	3

ADJUST							
50	nn	50nn	SET_ADJUST_CH	set channel **nn** adjust	1	-	8
51	nn	51nn	READ_ADJUST_CH	read channel **nn** adjust	-	1	8
52	-	52xx	SET_GLOB_OFFS	set global offset	1	-	16
53	-	53xx	READ_GLOB_OFFS	read global offset	-	1	16
54	-	54xx	EN_ALL_ADJUST	enable all adjusts	-	-	-
55	-	55xx	DIS_ALL_ADJUST	disable all adjusts	-	-	-
56	-	56xx	RESET_ALL_ADJUST	Reset all adjusts	-	-	-
57	-	57xx	READ_EN_ADJUST	read enable adjusts	-	1	1
EDGE DETECTION							
60	-	60xx	RISE_ALL	rising edge only on all channels	-	-	-
61	-	61xx	FALL_ALL	falling edge only on all channels	-	-	-
62	-	62xx	ODDR_EVENF	rising edge on odd ch. , falling edge on even	-	-	-
63	-	63xx	ODDF_EVENR	falling edge on odd ch. , rising edge on even	-	-	-
64	-	64xx	RISE_START	rising edge only start	-	-	-
65	-	65xx	FALL_START	falling edge only start	-	-	-
66	-	66xx	BOTH_ALL	both edges on all channels	-	-	-
67	-	67xx	READ_DETECTION	read edge detection configuration	-	3	2
DATA READY							
70	-	70xx	DR_EV_READY	set D.R. = event ready	-	-	-
71	-	71xx	DR_ALMOST_FULL	set D.R. = buffer almost full	-	-	-
72	-	72xx	DR_NOT_EMPTY	set D.R. = buffer not empty	-	-	-
73	-	73xx	READ_DR_MODE	read data ready mode	-	1	2
74	-	74xx	SET_ALM_FULL	set almost full level	1	-	15
75	-	75xx	READ_ALM_FULL	read almost full level	-	1	15
JTAG							
80	c	80xc	READ_TDC_ERR	read error code of TDC **c**	-	1	4
81	c	81xc	READ_TDC_ID	read ID code of TDC **c**	-	2	16
82	c	82xc	JTAG_W_SETUP	write JTAG setup register of TDC **c (INT. USE)**	8	-	16
REJECT OLD DATA							
90	-	90xx	SET_REJ_OFFS	set reject offset	1	-	16
91	-	91xx	READ_REJ_OFFS	read reject offset	-	1	16
92	-	92xx	EN_AUTO_REJ	enable automatic reject	-	-	-
93	-	93xx	DIS_AUTO_REJ	disable automatic reject	-	-	-
94	-	94xx	READ_EN_REJ	read enable reject	-	1	1
TRIGGER SEARCH WINDOW							
A0	-	A0xx	SET_AHEAD_WIN	set look ahead window	1	-	8
A1	-	A1xx	READ_AHEAD_WIN	read look ahead window	-	1	8
A2	-	A2xx	SET_BACK_WIN	set look back window	1	-	8
A3	-	A3xx	READ_BACK_WIN	read look back window	-	1	8
ADVANCED							
B0	-	B0xx	SET_DLL_CURR	set DLL current	1	-	5
B1	-	B1xx	READ_DLL_CURR	read DLL current	-	1	5
B2	-	B2xx	RESET_DLL	reset DLL	-	-	-
B3	-	B3xx	EN_DB_SYNC	enable double syncronizer	-	-	-
B4	-	B4xx	DIS_DB_SYNC	disable double syncronizer	-	-	-
B5	-	B5xx	EN_DB_PRIOR	enable double hit priority	-	-	-
B6	-	B6xx	DIS_DB_PRIOR	disable double hit priority	-	-	-
B/	-	B7xx	READ_ADV_CONF	read advanced configuration	-	1	3
B8	-	B8xx	SET_ERR_MASK	set error mask	1	-	4
B9	-	B9xx	READ_ERR_MASK	read error mask	-	1	4

4.2. TEST OPCODES

The following OPCODEs allow to perform tests on the FIFO and on the VME data readout.

4.2.1. ENABLE MEMORY TEST MODE (CODE 01xx)

Allows to write in the output FIFO Memory directly via VME, thus excluding the writing of data from the TDC chips into the FIFO. The data to write into the FIFO must be set via TESTWORD_H and TESTWORD_L registers (see § 3.24, 3.25). This OPCODE clears the FIFO content.

4.2.2. DISABLE MEMORY TEST MODE (CODE 02xx)

Restores the writing of data from the TDC chips into the FIFO Memory. This OPCODE clears the FIFO content.

4.2.3. READ MEMORY TEST ON/OFF (CODE 03xx)

Reads the status of the Memory test OPCODE settings. After this OPCODE a word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read out. The least significant bit of this word is related to the Enable/Disable Mode according to the following:
- LSB = 0 ⇒ TDC writes in the FIFO Memory;
- LSB = 1 ⇒ VME writes in the FIFO Memory.

4.3. ACQUISITION MODE OPCODES

The following OPCODEs allow to choose the acquisition modes as described in § 1.2 and to use either a startup configuration or a User defined configuration.

4.3.1. SET STOP TRIGGER MATCHING (CODE 10xx)

Allows to set the Stop Trigger Matching acquisition mode, as described in § 1.2.1. This OPCODE clears the FIFO content and resets the TDCs.

4.3.2. SET START TRIGGER MATCHING (CODE 11xx)

Allows to set the Start Trigger Matching acquisition mode, as described in § 1.2.2. This OPCODE clears the FIFO content and resets the TDCs.

4.3.3. SET START GATING (CODE 12xx)

Allows to set the Start Gating acquisition mode, as described in § 1.2.3. This OPCODE clears the FIFO content and resets the TDCs.

4.3.4. SET CONTINUOUS STORAGE (CODE 13xx)

Allows to set the Continuous Storage acquisition mode, as described in § 1.2.4. This OPCODE clears the FIFO content and resets the TDCs.

4.3.5. READ ACQUISITION MODE (CODE 14xx)

Reads the status of the Acquisition Mode OPCODE settings. After this OPCODE a word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read out. The two least significant bits of this word are related to the Acquisition Mode according to the following:
- LSB = 0, 0 ⇒ Stop Trigger Matching Mode;
- LSB = 0, 1 ⇒ Start Trigger Matching Mode;
- LSB = 1, 0 ⇒ Start Gating Mode;
- LSB = 1, 1 ⇒ Continuous Storage Mode.

4.3.6. LOAD DEFAULT CONFIGURATION (CODE 15xx)

Allows to load the Default Configuration: if AUTO_LOAD is disabled (code 18xx) at Power-ON the module will be programmed in the default operation mode:
- Stop Trigger Matching;
- Window Width = 100;
- Window Offset = 50;
- All channels enabled;
- Data Ready mode = Buffer not Empty.

4.3.7. SAVE USER CONFIGURATION (CODE 16xx)

Allows to save a User Configuration that can be recalled at any time by the User. This Configuration concerns:
- Acquisition Mode;
- Window Width;
- Window Offset;
- Enabled channels pattern;
- Data Ready mode.

If AUTO_LOAD is enabled (code 18xx) at Power-ON the module will be programmed in the user configuration operation mode:

4.3.8. LOAD USER CONFIGURATION (CODE 17xx)

Allows to load a User Configuration previously saved by the User (see above).

4.3.9. ENABLE AUTO LOAD (CODE 18xx)

Allows to load automatically the User Configuration either at the next Power-ON or at the next General RESET.

4.3.10. DISABLE AUTO LOAD (CODE 19xx)

Allows to disable the automatic load of the User Configuration at the next Power-ON or at the next General RESET. The Default Configuration will be loaded at Power-ON/RESET.

4.3.11. READ AUTO LOAD (CODE 1Axx)

Allows to read the status of the automatical load of the User Configuration. After this OPCODE a word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read out. The least significant bit of this word is related to the Auto Load status according to the following:
- LSB = 0 ⇒ Auto Load Mode disabled (Default Configuration loaded at Power-ON/RESET);

- LSB = 1 ⇒ Auto Load Mode enabled (User Configuration loaded at Power-ON/RESET).

4.4. CHANNEL ENABLE OPCODES

The following OPCODEs allow to enable or disable the individual TDC channels. These OPCODEs can be changed during an acquisition, so the User must be aware that a datum belonging to a previous enabling pattern may appear in the FIFO even if the channel has been disabled.

4.4.1. ENABLE CHANNEL nn (CODE 20nn)

Allows to enable channel nn (nn can be any value between %00 and %7F, corresponding respectively to channels 0 and 127).

4.4.2. DISABLE CHANNEL nn (CODE 21nn)

Allows to disable channel nn (nn can be any value between %00 and %7F, corresponding respectively to channels 0 and 127).

4.4.3. READ STATUS CHANNEL nn (CODE 22nn)

Reads the Enabled/Disabled status of channel nn. After this OPCODE a word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read out. The least significant bit of this word is related to the Enable/Disable status of the channel according to the following:
- LSB = 0 ⇒ channel disabled;
- LSB = 1 ⇒ channel enabled.

4.4.4. ENABLE ALL CHANNELS (CODE 23xx)

Allows to enable all channels.

4.4.5. DISABLE ALL CHANNELS (CODE 24xx)

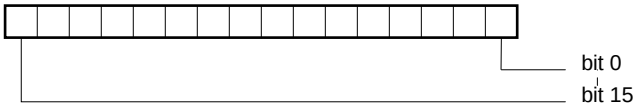
Allows to disable all channels.

4.4.6. WRITE ENABLE PATTERN (CODE 25xx)

Writes the pattern that allows to enable simultaneously the TDC channels (faster than in codes 20nn and 21nn). After this OPCODE 8 16-bit words must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until 8 16-bit words are written. The 16 bits of these words are related to the channel numbers according to the following:

word 0: bit 0 ⇒ channel 0
word 0: bit 1 ⇒ channel 1
.
word 0: bit 15 ⇒ channel 15

word 1: bit 0 ⇒ channel 16
word 1: bit 1 ⇒ channel 17
.
word 1: bit 15 ⇒ channel 31
:
word 7: bit 0 ⇒ channel 112
word 7: bit 1 ⇒ channel 113
.
word 7: bit 15 ⇒ channel 127



- bit = 0 ⇒ channel disabled;
- bit = 1 ⇒ channel enabled.

4.4.7. READ ENABLE/DISABLE WORDS (CODE 26xx)

Allows to read the pattern of the enabled channels. After this OPCODE 8 16-bit words must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until 8 16-bit words are read. The 16 bits of these words are related to the channel numbers according to § 4.4.6.

4.5. TRIGGER OPCODES

The following OPCODEs allow to set or read some trigger related parameters of the TDC chips. The following definition can be useful.

- Trigger Window: a time window of programmable width (**WIDTH**) and relative position (**OFFSET**) with respect to the common TRIGGER signal, defined as follows:

WIDTH = Tb - Ta ,

OFFSET = Ta - Tt .

Both the window width and the offset are expressed in clock cycles.

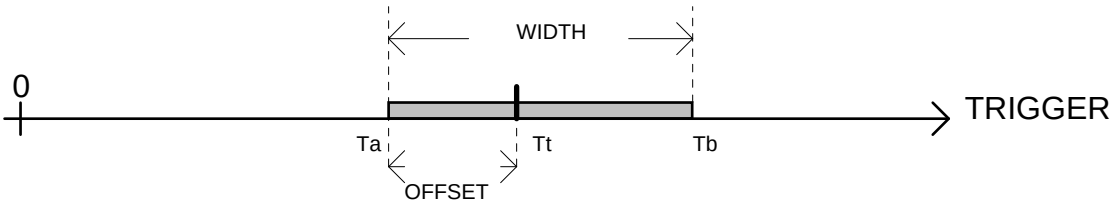


Fig. 4.3: Trigger Window: width and offset

4.5.1. SET WINDOW WIDTH (CODE 30xx)

Allows to set the width of the trigger window. After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The value of the word can be set in a range from 1 to 34000, or hex 84D0: the relevant window width is the word value times the clock period.

4.5.2. READ WINDOW WIDTH (CODE 31xx)

Allows to read the width of the trigger window, see § 4.5.1. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read.

4.5.3. SET WINDOW OFFSET (CODE 32xx)

Allows to set the offset of the trigger window with respect to the trigger itself, i.e. the time difference (expressed in clock cycles) between the trigger time and the start of the trigger window. After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The window offset value must be greater than -32000, or hex 8300. The offset and width value are linked by the:

OFFSET + WIDTH < 2000

This implies, together with the § 4.5.1 constraints, that the trigger window must lie between two values: Trigger_time - 32000 and Trigger_time + 2000. The window offset is synchronized with the clock cycle, thus there could be a jitter of one clock cycle in the actual offset position.

4.5.4. READ WINDOW OFFSET (CODE 33xx)

Allows to read the offset of the trigger window with respect to the trigger itself, see § 4.5.3. After this OPCODE 8 16-bit words must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read.

4.5.5. SET TRIGGER LATENCY (CODE 34xx)

Allows to set the latency of the trigger. Usually there is a delay from the trigger generation by a trigger supervisor and the actual trigger received by the V767 module: the trigger latency, expressed in clock cycles, allows to correct for this delay value, thus placing the offset of the trigger window in a different position.
After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The latency value can be any 16 bit value, though reasonable values are not greater than 50.

4.5.6. READ TRIGGER LATENCY (CODE 35xx)

Allows to read the latency of the trigger, see § 4.5.5.

4.5.7. ENABLE SUBTRACTION OF TRIGGER TIME (CODE 36xx)

Allows to enable the trigger time tag subtraction. Once enabled, the time measurement values read out from the TDCs (STOP in case of STOP Trigger Matching, START in case of START Trigger Matching) are referred to the trigger time tag, i.e. to the beginning of the trigger window. In STOP Trigger Matching mode this option is enabled by default, while in START Trigger Matching mode it is disabled by default.

4.5.8. DISABLE SUBTRACTION OF TRIGGER TIME (CODE 37xx)

Allows to disable the trigger time tag subtraction. Once disabled, the time measurement values read out from the TDCs (see here above) are referred to the RESET time (absolute time).

4.5.9. ENABLE OVERLAPPING TRIGGERS (CODE 38xx)

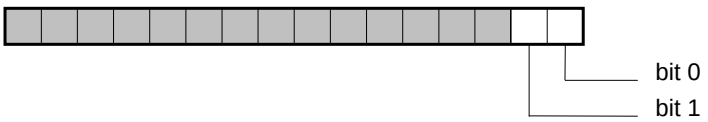
Allows to enable the overlapping of trigger windows.

4.5.10. DISABLE OVERLAPPING TRIGGERS (CODE 39xx)

Allows to disable the overlapping of trigger windows.

4.5.11. READ TRIGGER CONFIGURATION (CODE 3Axx)

Allows to read the trigger configuration. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits of this word only the 2 least significant bits are related to the start configurations according to the following:



Bit 0 is referred to the subtraction of trigger time:
Bit0=0 ⇒ trigger time subtraction disabled;
Bit0=1 ⇒ trigger time subtraction enabled;

Bit 1 is referred to the overlapping triggers:
Bit1=0 ⇒ overlapping triggers disabled;

Bit1=1 ⇒ overlapping triggers enabled.

4.6. START OPCODES

The following OPCODEs allow to set or read some START related parameters of the TDC chips.

4.6.1. ENABLE READOUT OF START TIME (CODE 40xx)

Allows to enable the readout of the start time. Once enabled, one start time only will be transferred to the local buffer (the TDC0 one).

4.6.2. ENABLE READOUT OF 4 START TIMES (CODE 41xx)

Allows to enable the readout of the 4 start times (one per chip). Once enabled, 4 start times will be transferred to the local buffer. As a check of the data consistency, the 4 START times should be all the same or differ at most of one clock cycle. If the EMPTY START is disabled (Code 46xx) only the NOT EMPTY STARTS will be read.

4.6.3. DISABLE READOUT OF START TIME (CODE 42xx)

Allows to disable the readout of the start time. Once disabled, no start time will be transferred to the local buffer.

4.6.4. ENABLE SUBTRACTION OF START TIME (CODE 43xx)

Allows to enable the subtraction of the start time from the hit time. Once enabled, the subtraction result (i.e. Hit_time-Start_time) will be transferred to the local buffer.

4.6.5. DISABLE SUBTRACTION OF START TIME (CODE 44xx)

Allows to disable the subtraction of the start time from the hit time. Once disabled, the absolute Hit time (i.e. the time referred to the RESET) will be transferred to the local buffer. This OPCODE can be done only with the 42xx OPCODE setting.

4.6.6. ENABLE EMPTY START (CODE 45xx)

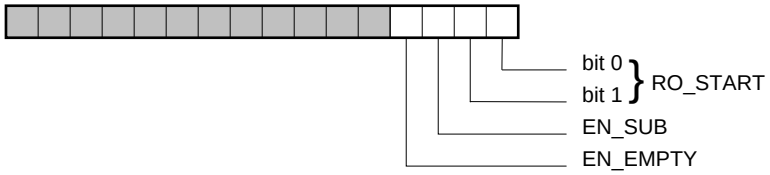
Allows to enable the readout of the start times (one per chip) also when there are no hits in the TDCs relevant to that particular START. Once enabled, 4 start times will be transferred to the local buffer if the 41xx OPCODE is enabled.

4.6.7. DISABLE EMPTY START (CODE 46xx)

Allows to disable the readout of the empty start times, i.e. start pulses that have no relevant hits. This OPCODE can be done only with the 41xx OPCODE enabled.

4.6.8. READ START CONFIGURATION (CODE 47xx)

Allows to read the start configuration. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits of this word only the 4 least significant bits are related to the start configurations according to the following:



EN_SUB and EN_EMPTY are 1 when respectively the subtraction of the start time is enabled (see § 4.6.4) and when the readout of the empty start times is enabled (see § 4.6.6).

Bit 0 and 1 are related to the Start Time readout mode according to the following:

	bit 1	bit 0
no start time readout	0	0
1 TDC start time readout	0	1
4 TDC start time readout	1	0

4.7. ADJUST OPCODES

The following OPCODEs allow to set or read some offset (fine counter) and global offset (coarse counter) adjustments of the TDC channels.

4.7.1. SET CHANNEL NN ADJUST (CODE 50nn)

Allows to introduce a positive offset for channel nn (nn can be any value between %00 and %7F, corresponding respectively to channels 0 and 127). After this OPCODE an 8-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until an 8-bit word is written. The offset value can be %0 to %FF: each bit has the weight of an LSB (TDC bin size).
N.B. With the occurrence of the first setting of a channel adjust after a Reset or Power-ON, the module automatically enables the channel adjust, sets all offsets to 0 and writes the offset for the selected channel to the set value. The writing operation at the first time takes approx. 10 seconds. The User must wait for this time after a new setting.

4.7.2. READ CHANNEL NN ADJUST (CODE 51nn)

Allows to read the positive offset (see above) for channel nn (nn can be any value between %00 and %7F, corresponding respectively to channels 0 and 127). After this OPCODE an 8-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until an 8-bit word is written.

4.7.3. SET GLOBAL OFFSET (CODE 52xx)

Allows to set a global offset in the coarse counter at the RESET time (see [2], § 3, p. 5). After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The offset value can be %0 to %FFFF.

4.7.4. READ GLOBAL OFFSET (CODE 53xx)

Allows to read the global offset in the coarse counter at the RESET time (see above). After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read.

4.7.5. ENABLE CHANNEL ADJUST (CODE 54xx)

Allows to enable the channel offset adjust that has been previously set with the 50nn OPCODE.

N.B. With the occurrence of the first issue of an enable channel adjust after a Reset or Power-ON (without any previous issue of a set channel adjust), the module automatically enables the channel adjust and sets all offsets to 0. The writing operation of zeroes in all offsets takes approx. 10 seconds. This operation is meaningless, nonetheless it can be issued. After a setting of the offsets on the desired channels, the channel adjust can be disabled and subsequently enabled. After these operations the channels will keep their offset setting without any delay.

4.7.6. DISABLE CHANNEL ADJUST (CODE 55xx)

Allows to disable the channel offset adjust that can be set with the 50nn OPCODE.

4.7.7. RESET ALL ADJUSTS (CODE 56xx)

Allows toset all the ajust values to 0.

4.7.8. READ ENABLE ADJUSTS (CODE 57xx)

Allows to read the status of the channel adjust enable configuration. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits of this word only the least significant bit is related to the enable configuration: bit 0 is set to 1 when the channel adjust is enabled (see § 4.7.5). The default configuration at power-ON is that the channel adjust is disabled.

4.8. EDGE DETECTION OPCODES

These OPCODEs allow to set the active edge detection on the TDC channels.

4.8.1. RISING EDGE ONLY ON ALL CHANNELS (CODE 60xx)

Allows to set the detection of the rising edge only of the input signals on all channels.

4.8.2. FALLING EDGE ONLY ON ALL CHANNELS (CODE 61xx)

Allows to set the detection of the falling edge only of the input signals on all channels.

4.8.3. RISING EDGE ON ODD CH., FALLING EDGE ON EVEN (CODE 62xx)

Allows to set the detection, on the input signals, of the rising edge on odd channels and falling edge on even channels.

4.8.4. FALLING EDGE ON ODD CH., RISING EDGE ON EVEN (CODE 63xx)

Allows to set the detection, on the input signals, of the falling edge on odd channels and rising edge on even channels.

4.8.5. RISING EDGE ONLY START (CODE 64xx)

Allows to set the detection of the rising edge only of the START pulse.

4.8.6. FALLING EDGE ONLY START (CODE 65xx)

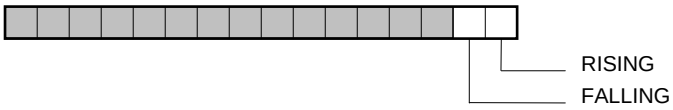
Allows to set the detection of the falling edge only of the START pulse.

4.8.7. BOTH EDGES ON ALL CHANNELS (CODE 66xx)

Allows to set the detection of both edges of the input signals on all channels and on the START channel. The two settings are strictly linked, e.g. it is not possible to set both edges on the START and rising edge only on the channels.

4.8.8. READ EDGE DETECTION CONFIGURATION (CODE 67xx)

Allows to read the edge detection configuration. After this OPCODE 3 16-bit words must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until 3 16-bit words are read. Of these 16 bits, only the 2 LSBs of these words are related to the edge detection configuration according to the following:



The first word is related to the EVEN channels; the second word is related to the ODD channels; the third word is related to the START channel. Bit 0 and 1 are related to the edge detection configuration according to the following:

	FALL	RISE
Rising edge	0	1
Falling edge	1	0
Both edges	1	1

4.9. DATA READY OPCODES

These OPCODEs allow to define the Data Ready status of the board.

4.9.1. SET D.R. = EVENT READY (CODE 70xx)

Data Ready condition is the occurrence of one complete event in memory.

4.9.2. SET D.R. = BUFFER ALMOST FULL (CODE 71xx)

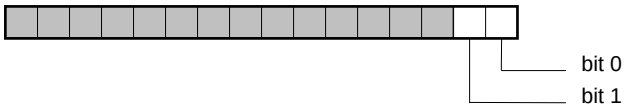
Data Ready is asserted if there is at least n data in memory. The value of n is set via code 74xx here below.

4.9.3. SET D.R. = BUFFER NOT EMPTY (CODE 72xx)

Data Ready is asserted if there is at least one datum in memory.

4.9.4. READ DATA READY MODE (CODE 73xx)

Allows to read the setting of the Data Ready mode. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of these 16 bits, only the 2 LSBs of these words are related to the data ready configuration according to the following:



Bit 0 and 1 are related to the Data Ready mode according to the following:

	bit 1	bit 0
Event Ready	0	0
Buffer almost full	0	1
Buffer not empty	1	0

4.9.5. SET ALMOST FULL LEVEL (CODE 74xx)

Allows to set the value n of data for the definition of "almost full" buffer. After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The number of data in memory (Almost full level) can be %2 to %3FFF.

4.9.6. READ ALMOST FULL LEVEL (CODE 75xx)

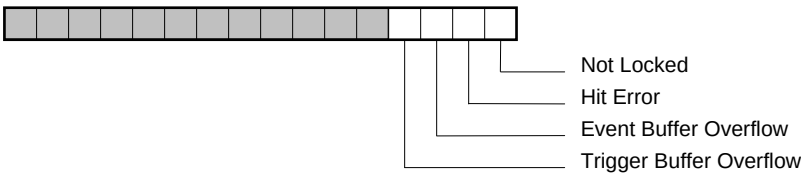
Allows to read the value n of data for the definition of "almost full" buffer. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read.

4.10. JTAG OPCODES

The following OPCODEs allow to read the error and ID codes of the TDCS via JTAG.

4.10.1. READ ERROR CODE OF TDC C (CODE 80xc)

Allows to read the error code of each TDC. The value of C can be 0 to 3. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of these 16 bits, only the 4 LSBs of these words are related to the TDC error codes according to the following (see [2], § 13, § 14.6):



N.B.: The error mask has one bit missing with respect to the [2], § 13, § 14.6: the SERIAL TRIGGER ERROR is not implemented.

4.10.2. READ ID CODE OF TDC C (CODE 81xc)

Allows to read the ID code of each TDC (see [2], § 14.3). After this OPCODE two 16-bit words must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until two 16-bit words are read.

4.10.3. WRITE JTAG SETUP REGISTER OF TDC C (INT. USE) (CODE 82xc)

This operation is for CAEN Internal Use Only.

4.11. REJECT OLD DATA OPCODES

The following OPCODEs allow to set the Reject Offset of the TDCs. It is recommended to change these settings for advanced operations only.

4.11.1. SET REJECT OFFSET (CODE 90xx)

Allows to set the Reject offset value, i.e. a time interval, preceding the beginning of the trigger window, used to reject old data from the TDC memory. After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. The default value (in Clock counts) is 16 (%10). A value of 0 is the beginning of the trigger window.

4.11.2. READ REJECT OFFSET (CODE 91xx)

Allows to read the Reject offset value. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read.

4.11.3. ENABLE AUTOMATIC REJECT (CODE 92xx)

Allows to enable the automatic reject (see [2], § 9, p. 11). As a default, at Power-On the automatic reject is enabled.

4.11.4. DISABLE AUTOMATIC REJECT (CODE 93xx)

Allows to disable the automatic reject (see [2], § 9, p. 11).

4.11.5. READ ENABLE REJECT (CODE 94xx)

Allows to read the setting of the Enable Reject mode. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of these 16 bits, only the LSB of this word is related to the Enable Reject mode according to the following:

- Bit0=0 ⇒ Relject Disabled;
- Bit0=1 ⇒ Relject Enabled;

4.12. TRIGGER SEARCH WINDOW OPCODES

The following OPCODEs allow to set some Trigger Window related parameters (Look Ahead and Look Back windows). See [2], § 9 for a full description of Look Ahead and Look Back windows. It is recommended to change these settings for advanced operations only.

4.12.1. SET LOOK AHEAD WINDOW (CODE A0xx)

Allows to set the Look Ahead Window value (see [2], § 9, p. 10). After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. Of the 16 bits, only the 8 Least Significant Bits are related to the look ahead window.

4.12.2. READ LOOK AHEAD WINDOW (CODE A1xx)

Allows to read the Look Ahead Window value (see [2], § 9, p. 10). After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits, only the 8 Least Significant Bits are related to the look ahead window.

4.12.3. SET LOOK BACK WINDOW (CODE A2xx)

Allows to set the Look Back Window value (see [2], § 9, p. 10). After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. Of the 16 bits, only the 8 Least Significant Bits are related to the Look Back window.

4.12.4. READ LOOK BACK WINDOW (CODE A3xx)

Allows to read the Look Back Window value (see [2], § 9, p. 10). After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits, only the 8 Least Significant Bits are related to the Look Back window.

4.13. ADVANCED OPCODES

The following OPCODEs allow to perform some advanced settings on the TDCs. See [2] for a complete description. It is recommended to change these settings for advanced operations only.

4.13.1. SET DLL CURRENT (CODE B0xx)

Allows to set the DLL current (see [2], § 2, p. 4). After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. Of the 16 bits, only the 5 Least Significant Bits are related to the DLL current.

4.13.2. READ DLL CURRENT (CODE B1xx)

Allows to read the DLL current (see [2], § 2, p. 4). After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits, only the 5 Least Significant Bits are related to the DLL current.

4.13.3. RESET DLL (CODE B2xx)

Allows to reset the DLL of the TDCs (see [2], § 14.4.2).

4.13.4. ENABLE DOUBLE SYNCHRONIZER (CODE B3xx)

Enables the Double Synchronizer (see [2], § 4).

4.13.5. DISABLE DOUBLE SYNCHRONIZER (CODE B4xx)

Disables the Double Synchronizer (see [2], § 4).

4.13.6. ENABLE DOUBLE HIT PRIORITY (CODE B5xx)

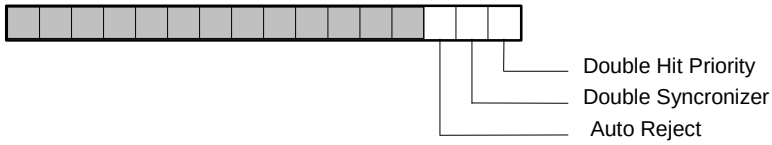
Enables the Double Hit Priority (see [2], § 7).

4.13.7. DISABLE DOUBLE HIT PRIORITY (CODE B6xx)

Disables the Double Hit Priority (see [2], § 7).

4.13.8. READ ADVANCED CONFIGURATION (CODE B7xx)

Allows to read the advanced configuration. After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. Only the 3 LSBs of this word are related to the advanced configuration according to the following:



4.13.9. SET ERROR MASK (CODE B8xx)

Allows to set the error mask, i.e. to enable the TDC error flag generation (see § 4.10.1). After this OPCODE a 16-bit word must be written at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is written. Of the 16 bits, only the 4 Least Significant Bits are related to the error mask. At Power-ON the error mask has all bits enabled.

4.13.10. READ ERROR MASK (CODE B9xx)

Allows to read the error mask (see above). After this OPCODE a 16-bit word must be read at the same location of the OPCODE itself. The microcontroller will remain in a wait status until a 16-bit word is read. Of the 16 bits, only the 4 Least Significant Bits are related to the error mask.

5. OPERATING MODES

5.1. INTRODUCTION

The data acquisition can be programmed in "EVENTS" (TRIGGER MATCHING with a programmable time window or START GATING modes) or in "CONTINUOUS STORAGE". The management of overlapping triggers is also performed.

The COMMON STOP operation, though not existing in the chip itself, can be easily implemented on the board by assigning one of the 128 channels to a STOP signal and by an adequate programming of the trigger window.

The module programming is performed via a microcontroller that implements a high-level interface towards the User in order to mask the board and the TDCs' hardware.

5.2. POWER-ON/RESET STATUS

At Power-ON or after a VME SYSRES some registers might be in an undetermined status. It is thus suggested to perform always a VME RESET (via Bit Set Register, see § 3.6) or a Single Shot Reset (access to address Base + %0018). After the RESET the module is in the following status:

1. the module is either in the Default Configuration (see 4.3.6) or in the User Configuration (see § 4.3.7) if the Auto Load (see § 4.3.9) is, respectively, disabled or Enabled;
2. the Output Buffer is cleared;
3. the Interrupt Level/Vector is set to 0;
4. the Event Counter is set to 0;
5. the bits 2, 4 and 5 of Control Register 1 are set to 0 (see § 3.11);
6. the MCST Address is set to 0xAA;
7. the OPCODE handshake (see § 3.21) has ROK=0 and WOK=1 (module ready to receive an OPCODE;

The following registers are not affected by a VME RESET or a Single Shot RESET. At Power_ON or after a VME SYSRES their status is the following:

1. The ADER24 and ADER32 are cleared;
2. the MCST Control is cleared (board inactive in MCST chain);
3. the Bit Set Register is cleared.

N.B.: After a Power-On or a VME Reset or a SYSRES, it is necessary to wait for about 2 s to allow for its initialisation (see, for example, the C-like language software example in § 5.9).

5.3. START AND TRIGGER INPUTS

These are 110 Ohm impedance ECL inputs and allow to control easily a system of many units via the front panel CONTROL BUS. Their action on the module depend from the selected operating mode (see below). Their minimum length must be greater than a clock period (25 ns for a 40 MHz CLOCK).
The trigger input is a rising-edge active signal.

5.4. RESET INPUT

This is a 110 Ohm impedance ECL input (active low) available on the CONTROL bus. Its minimum length must be greater than a clock period (25 ns for a 40 MHz CLOCK). The action of a pulse through this input can be programmed via Control Register 1 (see § 3.11): if the PROGRESET bit is set to 1 a RESET pulse is equivalent to the VME Reset or Single Shot Reset. If the PROGRESET bit is set to 0, a RESET pulse clears the Output Buffer and resets the TDC chips.

5.5. CLOCK INPUT

This is a 110 Ohm impedance ECL input available on the CONTROL bus. Maximum external CLOCK frequency is 45 MHz (60 MHz under implementation). The selection of External/Internal CLOCK is performed via internal SRCCLK Jumper (See Fig. 2.4).

5.6. BUSY OUTPUT

An active-high ECL signal ("BUSY") is available on the front panel CONTROL BUS. This is an ECL output into 110 Ohm impedance and allows to obtain a wired-OR Global Busy signal of a system of many units connected together via the CONTROL BUS.

Each module sets to 1 its Busy output in one of the following conditions:

1) the output buffer is full;
2) if the module is in TEST mode;
3) in Start Gating mode, with the occurrence of the START signal itself.

The module releases the Busy to 0 respectively (see above) when he following are true:

1) some data are read out from the output buffer (output buffer not full);
2) the module is in Normal mode;
3) in Start Gating mode, the START signal has become low and all TDC chips have been readout (timings ruled by the readout controller internal logic);

N.B.: When the Module is Busy it is care of the User not to generate another START or Trigger pulse.

If many units are connected via the CONTROL BUS START and BUSY signals, upon occurrence of the above described conditions in at least one module, the Global Busy signal is set to 1 and it is released to 0 only when all the V767 modules in the chain have completed the readout sequence and the entire system is ready to accept another START.

N.B.: The BUSY on each single boards is an ECL differential signal, but, once connected via the CONTROL BUS to other boards, only the positive end of the differential couple becomes the Wired-OR Global Busy signal, and is thus single ended.

5.7. DATA READY OUTPUT

An active-high ECL signal ("DRDY") is available on the front panel CONTROL BUS. This is an ECL output into 110 Ohm impedance and allows to obtain a wired-OR DATA READY signal of a system of many units connected together via the CONTROL BUS.

Each module sets to 1 its DATA READY output with the occurrence of one of the following situations, programmable via the Control Register 2 (see § 3.18):

1) Buffer contains a complete event;
2) Buffer almost full;
3) Buffer not empty.

Also the DATA READY signals can be Wire-ORed in order to have a Global DATA READY signal.

N.B.: The DATA READY on each single boards is an ECL differential signal, but, once connected via the CONTROL BUS to other boards, only the positive end of the differential couple becomes the Wired-OR Global DATA READY signal, and is thus single ended.

5.8. TRIGGER WINDOW DEFINITION

The following definition can be useful.

- Trigger Window: a time window of programmable width (**WIDTH**) and relative position (**OFFSET**) with respect to the common TRIGGER signal, defined as follows:

WIDTH = Tb - Ta ,

OFFSET = Ta - Tt .

Both the window width and the offset are expressed in clock cycles.

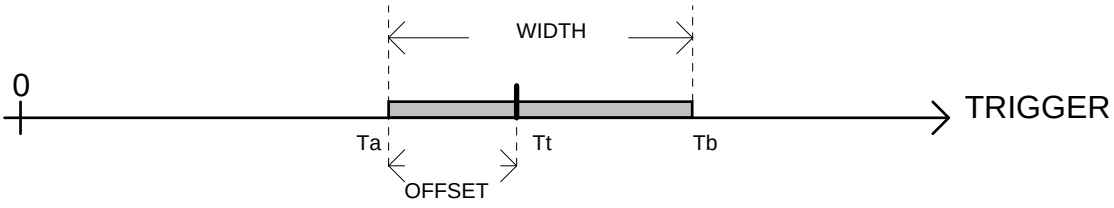


Fig. 5.1: Trigger Window

5.9. STOP TRIGGER MATCHING OPERATION

The STOP TRIGGER MATCHING operation is described in 1.2.1. The minimal set of OPCODE settings that the User must perform for STOP TRIGGER MATCHING operation are described here below.

- Reset;
- Set Stop Trigger Matching via OPCODE 1000 (see § 4.3.1);
- Set the Trigger Window Width (see § 4.5.1, default value after a RESET: 100);
- Set the Trigger Window Offset (see § 4.5.3, default value after a RESET: -50);
- Set Data Ready = Event Ready (suggested; see also § 4.9);

After these settings, the module is ready to receive the Trigger and individual Hit pulses.

Moreover, the user can also perform one or more of the following optional settings, if needed by the specific application:

- Enable the desired channels via the relevant OPCODEs (see § 4.4; default after a RESET: ALL enabled);
- Set the advanced Trigger Parameters (latency, subtraction & overlap, see § 4.5);
- Set the desired edge detection configuration (see § 4.8, default value after a RESET: Rising).

In the following a simple C-language example containing the settings required to operate in Stop Trigger Matching mode and a readout operation is listed. Refer also to § 4.1 for the procedure **`write-op`** called inside the program.

Example 1:

```
/*************************************************************/
/* simple example of Stop Trigger Matching setting and readout */
/*************************************************************/

void readout_data()
  {
  /* pointers to V767 registers */
  unsigned short *status1,rdata;
  unsigned long  *out_buff;

  unsigned long header,data,eob;

  /* sets V767 registers addresses */
  status1  = (unsigned short *)(V767_BASE_ADDRESS + 0x0E);
  out_buff = (unsigned long *)(V767_BASE_ADDRESS + 0x00);

  /* waits for DATA READY set to 1 */
  do
    rdata = *status1;  /* reads status register */
  while(!(rdata & 0x01));

  /* reads the event from output buffer and prints it on the screen */
  header = *out_buff;
  printf("Event number = %d\n",header & 0xFFF);
  data = *out_buff;
  printf("Channel = %d  -  Time = %d\n",(data>>24) & 0x3F, data & 0xFFFFF);
  eob = *out_buff;
  printf("Number of words = %d\n",eob & 0xFFFF);
  }

main()
  {
  unsigned short *reset, rdata;
  short win_offs, win_width;

  /* sets V767 registers addresses */
  reset  = (unsigned short *)(V767_BASE_ADDRESS + 0x18);

  /* resets the V767 and waits for 2 sec */
  *reset = 0;  /* vme write access to single shot reset register */
  delay(2000); /* waits until the initialization is completed (about 2 sec) */
```

```
win_offs = -100;
win_width = 200;

write_op(0x1000);       /* sets Stop Trigger Matching mode */
write_op(0x3000);       /* sets window width */
write_op(win_width);
write_op(0x3200);       /* sets window offset */
write_op(win_offs);
write_op(0x7000);       /* sets Data_Ready_Mode = Event_Ready */

printf("Ready to receive input signals\n");

readout_data();
}
```

In this software example the window parameters are set (in clock cycles) as follows:

- Win-offs = -100,
- Win_width = 200.

The resulting window is shown in Fig. 5.2:

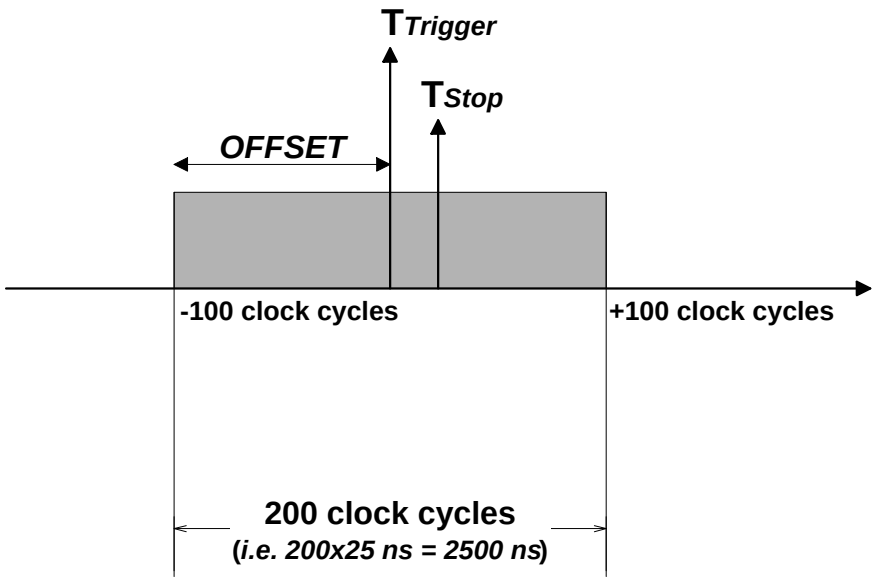


Fig. 5.2: Trigger Window to test Stop Trigger Matching operation mode

Send the TRIGGER and HIT CHANNEL 0 input signals with 100 ns delay between their rising edges (refer to Fig. 5.3):

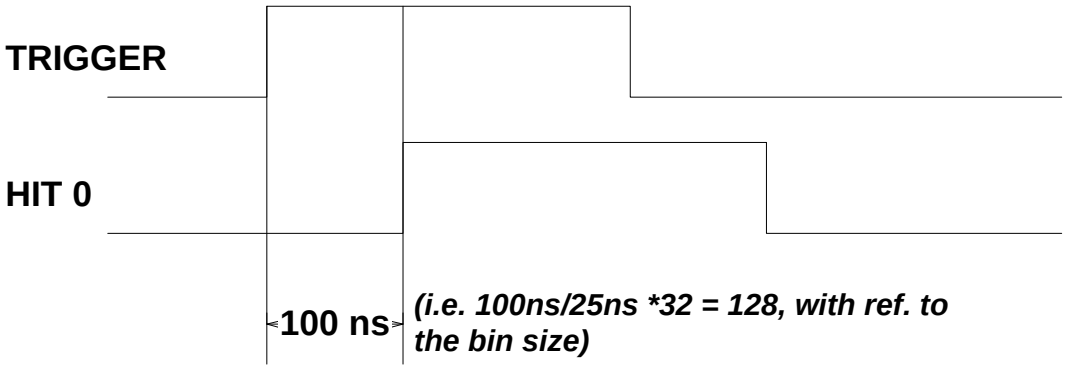


Fig. 5.3: TRIGGER and HIT 0 signals to test Stop Trigger Matching operation mode

As these signals are sent in, the DT_RDY LED will blink and, straight after, the following data will appear on the screen:

```
Event number = 0
Channnel = 0  -  Time = 3328 (*)
Number of words = 1

(*) : This time is referred to the beginning of the trigger window:

      Time = (Tstop - Ttrigger) - offset = 128 + 32*100 = 3328

      Ttrigger and Toffset are expressed in clock cycles (25ns = 32 bin size).
      Since the trigger time is defined with a resolution of 1 clock cycle,
      the result (3328) will have a 25-ns jitter. Use a start reference
      signal in order to have a 1-bin size resolution.
```

5.10. START TRIGGER MATCHING OPERATION

The START TRIGGER MATCHING operation is described in 1.2.2. The minimal set of OPCODE settings that the User must perform for START TRIGGER MATCHING operation are described here below.

- Reset;
- Set Start Trigger Matching via OPCODE 1100 (see § 4.3.2);
- (see § 4.4; default after a RESET: ALL enabled);
- Set the Trigger Window Width (see § 4.5.1, default value after a RESET: 100);
- Set the Trigger Window Offset (see § 4.5.3, default value after a RESET: -50);
- Set Data Ready = Event Ready (suggested; see also § 4.9);

After these settings, the module is ready to receive the Start, Trigger and individual Hit pulses.

Moreover, the user can also perform one or more of the following optional settings, if needed by the specific application:

- Enable the desired channels via the relevant OPCODEs (see § 4.4; default after a RESET: ALL enabled);
- Set the desired Start parameters configuration (see § 4.6);
- Set the advanced Trigger Parameters (latency, subtraction & overlap, see § 4.5);
- Set the desired edge detection configuration (see § 4.8, default value after a RESET: Rising).

In the following a simple C-language example containing the settings required to operate in Start Trigger Matching mode and a readout operation is listed. Refer also to § 4.1 for the procedure **`write-op`** called inside the program.

Example 1:

```
/*****************************************************************/
/* simple example of Start Trigger Matching setting and readout */
/*****************************************************************/

void readout_data()
  {
  /* pointers to V767 registers */
  unsigned short *status1, rdata;
  unsigned long  *out_buff;

  unsigned long header,data,eob;

  /* sets V767 registers addresses */
  status1  = (unsigned short *)(V767_BASE_ADDRESS + 0x0E);
  out_buff = (unsigned long *)(V767_BASE_ADDRESS + 0x00);

  /* waits for DATA READY set to 1 */
  do
    rdata = *status1;
  while(!(rdata & 0x01));

  /* reads the event from output buffer and prints it on the screen */
  header = *out_buff;
  printf("Event number = %d\n",header & 0xFFF);
  data = *out_buff;   /* start datum */
  printf("Start absolute Time = %d\n", data & 0xFFFFF);
  data = *out_buff;   /* stop datum */
  printf("Channel = %d  -  Time = %d\n",(data>>24) & 0x3F, data & 0xFFFFF);
  eob = *out_buff;
  printf("Number of words = %d\n",eob & 0xFFFF);
  }


main()
  {
  unsigned short *reset, rdata;
  short win_offs, win_width;
  long data;

  /* sets V767 registers addresses */
  reset  = (unsigned short *)(V767_BASE_ADDRESS + 0x18);
```

```
/* resets the V767 and waits for 2 sec */
*reset = 0;  /* vme write access to single shot reset register */
delay(2000); /* waits until the initialization is completed (about 2 sec) */

win_offs = -100;
win_width = 200;

write_op(0x1100);       /* sets Start Trigger Matching mode */
write_op(0x3000);       /* sets window width */
write_op(win_width);
write_op(0x3200);       /* sets window offset */
write_op(win_offs);
write_op(0x7000);       /* sets Data_Ready_Mode = Event_Ready */

printf("Ready to receive input signals\n");

readout_data();
}
```

In this software example the window parameters are set as follows (refer to Fig. 5.2):

- Win-offs = -100,
- Win_width = 200.

Send the TRIGGER, START and HIT CHANNEL 0 input signals as shown in Fig. 5.4:

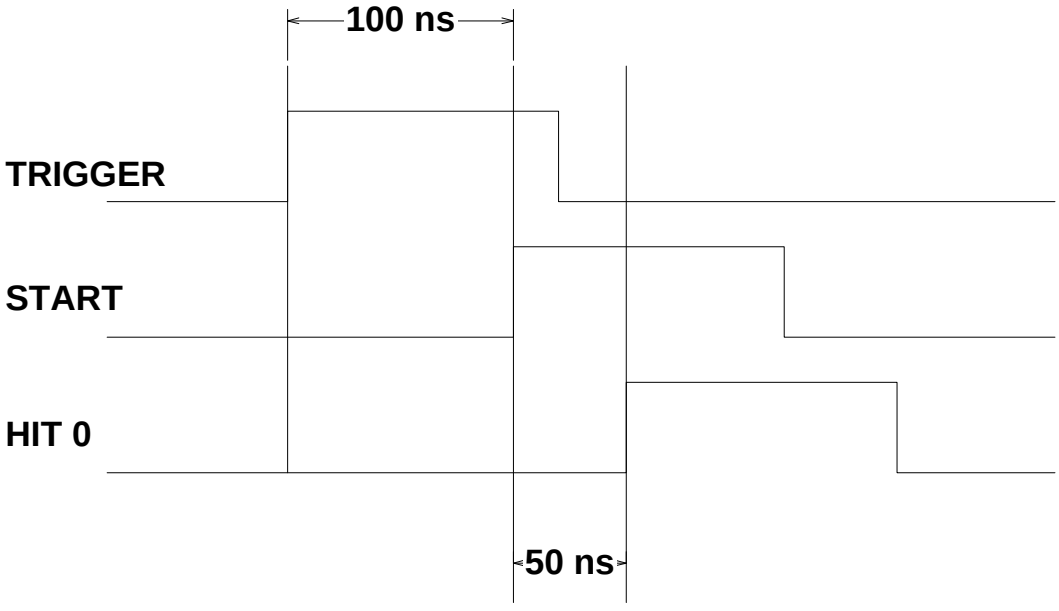


Fig. 5.4: TRIGGER, START and HIT 0 signals to test Start Trigger Matching mode

As these signals are sent in, the DT_RDY LED will blink and, straight after, the following data will appear on the screen:

```
Event number = 0
Start absolute time = xxxxx      (according to the absolute time counter)
Channnel = 0  -  Time = 64       (i.e. 50ns/25ns * 32 )
Number of words = 2
```

5.11. START GATING OPERATION

The START GATING operation is described in 1.2.3. The minimal set of OPCODE settings that the User must perform for START GATING operation are described here below.

- Reset;
- Set Start Gating via OPCODE 1200 (see § 4.3.3);
- Set Data Ready = Event Ready (suggested; see also § 4.9);

After these settings, the module is ready to receive the Start and individual Hit pulses.

Moreover, the user can also perform one or more of the following optional settings, if needed by the specific application:

- Enable the desired channels via the relevant OPCODEs (see § 4.4; default after a RESET: ALL enabled);
- Set the desired Start parameters configuration (see § 4.6);
- Set the desired edge detection configuration (see § 4.8, default value after a RESET: Rising).

In the following a simple C-language example containing the settings required to operate in Start Gating mode and a readout operation is listed. Refer also to § 4.1 for the procedure **`write-op`** called inside the program.

Example 1:

```
/*****************************************************/
/* simple example of Start Gating setting and readout */
/*****************************************************/

void readout_data()
  {
  /* pointers to V767 registers */
  unsigned short *status1, rdata;
  unsigned long  *out_buff;

  unsigned long header,data,eob;

  /* sets V767 registers addresses */
  status1  = (unsigned short *)(V767_BASE_ADDRESS + 0x0E);
  out_buff = (unsigned long *)(V767_BASE_ADDRESS + 0x00);

  /* waits for DATA READY set to 1 */
  do
    rdata = *status1;
```

```
  while(!(rdata & 0x01));

  /* reads the event from output buffer and prints it on the screen */
  header = *out_buff;
  printf("Event number = %d\n",header & 0xFFF);
  data = *out_buff;   /* start datum */
  printf("Start absolute Time = %d\n", data & 0xFFFFF);
  data = *out_buff;   /* stop datum */
  printf("Channel = %d  -  Time = %d\n",(data>>24) & 0x3F, data & 0xFFFFF);
  eob = *out_buff;
  printf("Number of words = %d\n",eob & 0xFFFF);
  }


main()
  {
  unsigned short *reset, rdata;

  /* sets V767 registers addresses */
  reset  = (unsigned short *)(V767_BASE_ADDRESS + 0x18);

  /* resets the V767 and waits for 2 sec */
  *reset = 0;  /* vme write access to single shot reset register */
  delay(2000); /* waits until the initialization is completed (about 2 sec) */

  write_op(0x1200);       /* sets Start Gating mode */
  write_op(0x7000);       /* sets Data_Ready_Mode = Event_Ready */

  printf("Ready to receive input signals\n");

  readout_data();
  }
```

Send the START and HIT CHANNEL 0 input signals as shown in Fig. 5.5:

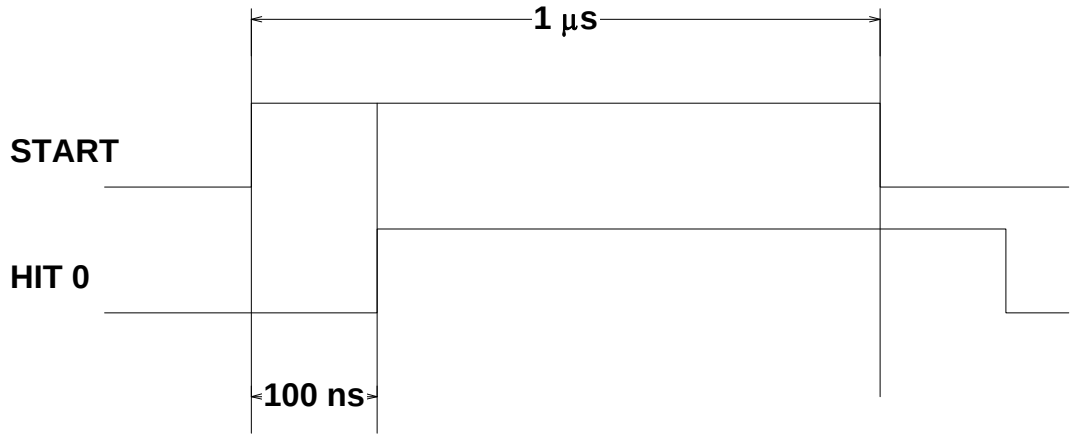


Fig. 5.5: START and HIT 0 signals to test Start Gating operation mode

As these signals are sent in, the DT_RDY LED will blink and, straight after, the following data will appear on the screen:

```
Event number = 0
Start absolute time = xxxxx      (according to the absolute time counter)
Channnel = 0  -  Time = 128      (i.e. 100ns/25ns * 32)
Number of words = 2
```

5.12. CONTINUOUS STORAGE OPERATION

The CONTINUOUS STORAGE operation is described in 1.2.4. The minimal set of OPCODE settings that the User must perform for CONTINUOUS STORAGE operation are described here below.

- Reset;
- Set CONTINUOUS STORAGE via OPCODE 1300 (see § 4.3.4);
- Set the desired Data Ready Mode (see § 4.9; Data ready = Almost Full or Not Empty mode. Event ready is NOT allowed);

After these settings, the module is ready to receive the Start and individual Hit pulses.

Moreover, the user can also perform one or more of the following optional settings, if needed by the specific application:

- Enable the desired channels via the relevant OPCODEs (see § 4.4; default after a RESET: ALL enabled);
- Set the desired Start parameters configuration (see § 4.6);
- Set the desired edge detection configuration (see § 4.8, default value after a RESET: Rising).

In the following a simple C-language example containing the settings required to operate in Continuous Storage mode and a readout operation is listed:

Example 1:

```
/*************************************************************/
/* simple example of Continuous Storage setting and readout */
/*************************************************************/

void readout_data()
  {
  /* pointers to V767 registers */
  unsigned short *status1,rdata;
  unsigned long  *out_buff;
  unsigned long data;
  int i;

  /* sets V767 registers addresses */
  status1  = (unsigned short *)(V767_BASE_ADDRESS + 0x0E);
  out_buff = (unsigned long *)(V767_BASE_ADDRESS + 0x00);

  /* reads three data */
  for(i=0; i<3; i++)
    {
    /* waits for DATA READY set to 1 */
    do
```

```
      rdata = *status1;
    while(!(rdata & 0x01));

    /* reads a datum from the output buffer and prints it on the screen */
    data = *out_buff;
    if(data & 0x800000)
      printf("Start absolute Time = %d\n", data & 0xFFFFF);  /* start datum */
    else
      printf("Channel = %d  -  Time = %d\n",(data>>24) & 0x3F,
             data & 0xFFFFF);  /* stop datum */
    }

  }


main()
  {
  unsigned short *reset, rdata;

  /* sets V767 registers addresses */
  reset  = (unsigned short *)(V767_BASE_ADDRESS + 0x18);

  /* resets the V767 and waits for 2 sec */
  *reset = 0;  /* vme write access to single shot reset register */
  delay(2000); /* waits until the initialization is completed (about 2 sec) */

  write_op(0x1300);       /* sets Start Gating mode */
  write_op(0x7200);       /* sets Data_Ready_Mode = Not Empty */

  printf("Ready to receive input signals\n");

  readout_data();
  }
```

Send the START, HIT CHANNEL 0 and HIT CHANNEL 1 input signals as shown in Fig. 5.6:

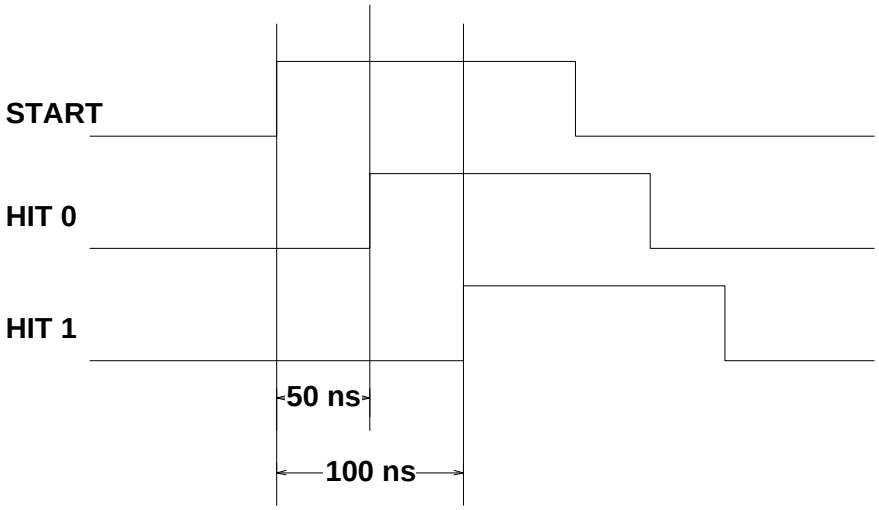


Fig. 5.6: START, HIT 0 and HIT 1 signals to test Continuous Storage operation mode

As these signals are sent in, the DT_RDY LED will blink and, straight after, the following data will appear on the screen:

```
Start absolute time = xxxxx      (according to the absolute time counter)
Channnel = 0   -  Time = 64      (i.e. 50ns/25ns * 32)
Channnel = 1   -  Time = 128     (i.e. 100ns/25ns * 32)
```

5.13. COMMON STOP EMULATION OPERATION

The COMMON STOP EMULATION operation is described in § 1.2.5. The minimal set of OPCODE settings that the User must perform for COMMON STOP EMULATION operation are described here below.

- Set Stop Trigger Matching via OPCODE 10xx (see § 4.3.1);
- Set the Trigger Window Width to a value n (see § 4.5.1);
- Set the Trigger Window Offset to -n (see § 4.5.3);
- Set Data Ready = Event Ready (suggested; see also § 4.9);

After these settings, the module is ready to receive the "emulated" Stop and individual Hit pulses.

Moreover, the user can also perform one or more of the following optional settings, if needed by the specific application:

- Enable the desired channels via the relevant OPCODEs (see § 4.4; default after a RESET: ALL enabled);
- Set the advanced Trigger Parameters (see § 4.5):
  subtraction DISABLED;
  desired overlap;
- Set the desired edge detection configuration (see § 4.8, default value after a RESET: Rising).

N.B.: The Trigger Window is synchronized with the Clock Cycle: there could be a jitter of one datum after the stop. A Software filtering is thus desired.

Via software the User must subtract the Hit time from the Stop time (Channel 0 Hit Time). In this way the complete Common Stop emulation is performed.

5.14. DATA BUFFER STRUCTURE

The data in the output buffer (Base address + %0000; from firmware rev 2.0 the Output Buffer is accessible also at Base + %2000 ÷ + %FFFC) are organised as follows:

A) In the case of *Start Trigger Matching*, *Stop Trigger Matching* (*Common Stop Emulation* included) and *Start Gating* mode the data in the buffer are organised in events.
Each event consists of (refer to Fig. 3.4):

- a **header** that contains the event number value;
- the **data words** containing the 20 bit converted time values, the channel number and an edge bit;
- an **End Of Block** (EOB) word containing the number of read words (HEADER and EOB not included) and a Status that is 1 if the TDC Chips have had an error, 0 otherwise.

B) In the case of *Continuous Storage* mode there are neither the header nor the EOB word, but only the data words.

Refer to § 3.4 for further details.

N.B.: in the Mod. V767B which does not have the JAUX, the GEO address contained in the HEADER and in the EOB must be written by the user via a write access to the relevant register (see § 3.5). If this operation is not performed, it will be not possible to identify which module the data are coming from when the CBLT access is used.

5.15. BLOCK TRANSFER MODE

The module supports the Standard BLT32 and MBLT64 modes. The block transfer readout is handled by the usage of two bits, namely the BERR_ENABLE and the BLKEND, available in the Control Register 1 (see § 3.11).

If BLK_END = 0 The module sends to the CPU all the requested data; when the Output Buffer is empty it will send not valid data. If BERR_VME is enabled, a Bus Error is generated with the readout of the last word in the Output Buffer.

If BLK_END = 1 The module sends to the CPU all data until the first EOB word (End of first event); afterwards it will send not valid data. If BERR_VME is enabled, a Bus Error is generated with the readout of the EOB word.

If BERR_EN =0 the BERR_VME line is disabled during block transfer. If the data in the Output Buffer are ended, the board will send meaningless data to the CPU (bit 21 and 22 of data word set to 1).

If BERR_EN =1 the BERR_VME line is enabled. After the last readout datum the board will generate a Bus Error.

5.16. ADVANCED SETTING AND READOUT MODES

The V767 module houses 128 channels. The set and readout time of these channels can be enhanced in various ways, namely Multicast Commands (MCST) and Chained Block Transfer (CBLT).

In order to perform CBLT and MCST operations, the higher base address bits of all the involved modules (i.e. bits 31 to 24, set via the two MSB rotary switches) must be set in common to all boards (i.e. all boards must have the same setting on bits 31 to 24).

The resulting MCST (CBLT) Base Address (See § 3.1.4) must be:

MCST (CBLT) Base Address = %NN000000,

where %NN is the common value set in the two MSB rotary switches.

In CBLT and MCST operations, the IACKIN and IACKOUT VME lines are used for the control transfer from one board to the following. No empty slots must thus be left between the boards or, in alternative, empty slots can be left only in case VME crates with automatic IACKIN/IACKOUT short-circuiting are used.

Once the addresses have been set, the first and last board in a chain must have, respectively, only the FIRST_BOARD and only the LAST_BOARD bit set to 1 in the MCST_Control_Register (see § 3.16). All intermediate boards which are active must have, on the contrary, both the FIRST_BOARD and the LAST_BOARD bits set to 1 in the MCST_Control_Register (see § 3.16).

5.16.1. CHAINED BLOCK TRANSFER MODE

Once set the address and position of the boards as described in § 3.14 and § 3.16, the boards can be accessed in Chained Block Transfer (CBLT, see [6]) mode, that allows to read out sequentially a certain number of contiguous boards in a VME crate. This access is allowed in BLT32 mode only.

N.B.: The CBLT operation can be performed only for the readout of the Output Buffer, whose address in CBLT mode is a dummy Address containing the common set value chosen by the User on the MCST Address Register (see § 3.16).

The CBLT uses always the VME Bus Error (BERR) generated by the last board in the chain as a data readout completion in the last board. This feature is very useful if the VME CPU handles the Bus Error in an efficient way. For a correct board operation, it is **MANDATORY** that a VME Bus Error is generated.

In CBLT operation the data coming from different boards are tagged with the HEADER and with the EOB words containing the GEO address in the 5 MSB (see also § 5.14). In the Mod. V767B (i.e. the version without the JAUX) it is up to the user to write the GEO address (this operation is possible only if the JAUX is not present) in the GEO register before executing the CBLT operation. If the GEO address is not written in the relevant register before performing the CBLT operation, it will not be possible to identify the module which the data are coming from.

5.16.2. MULTICAST COMMANDS

Once set the address of the boards as described in 5.16, the boards can be accessed in Multicast Commands (MCST, see [6]) mode, that allows to write in the registers of several boards at the same time by accessing only once a dummy Address. The latter is composed by the MCST Base Address plus the possible choices of commands shown in Table 3.2.

The MCST Control Register must NEVER be accessed in MCST mode because it can affect the CBLT and MCST operations themselves.

6. REFERENCES

[1] http://pcvlsi5.cern.ch:80/MicDig/

[2] J. Christiansen, **32 Channel general Purpose Time to Digital Converter**, Draft Version 0.3, CERN/ECP-MIC.
Available at URL: http://tdc.web.cern.ch/tdc/tdc32/tdc32.htm
see also:
http://pcvlsi5.cern.ch:80/MicDig/tdc32.htm
http://pcvlsi5.cern.ch:80/MicDig/jorgen/tdc32_ds.pdf

[3] H. Bleeker et al, **Boundary Scan Test, A Practical Approach**, Kluwer Academic Publishers, Netherlands, 1993.

[4] VMEbus Specification Manual Revision C.1, October 1985.

[5] G. Bianchetti et al., **Specification for VMEbus CRATE Type V430**, CERN-EP, January 1990.

[6] - VME64 Extensions Draft Standard, **VITA 1.1-199x**, Draft 1.8, June 13, 1997.
- VMEbus for Physics Apllications, Implementation Rules, Recommendations & Guidelines,
VITA23- 199x, Draft 1.0, 22 May 1997.
Both documents are available from URL: http://www.vita.com