Engineering specification

THE CTR USER GUIDE

Abstract

This document gives an introduction to the CTRI, CTRP and CTRV and its test program

Prepared by: Pablo Alvarez

Checked by:

Date: 17-Jul-07
## History of Changes

<table>
<thead>
<tr>
<th>Rev. No.</th>
<th>Date</th>
<th>Pages</th>
<th>Description of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>08-Jun-05</td>
<td></td>
<td>First draft.</td>
</tr>
<tr>
<td>2</td>
<td>17-Jul 2007</td>
<td>Added 5-7</td>
<td>CTRV Jumpers Added</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **MODULE OVERVIEW** .................................................................4
   1.1 OVERVIEW HARDWARE INTERFACE ..................................................5
   1.1.1 FRONT PANEL ..............................................................................5
   1.1.2 CTRV JUMPER SETTINGS ...............................................................5
      1.1.2.1 CTRV VERSION 2 (DRAWING BY Matthieu Cattin) ......................6
      1.1.2.2 CTRV VERSION 3 (DRAWING BY Matthieu Cattin) ......................7
   1.1.3 COUNTERS ..................................................................................8
   1.1.4 UTC SYNCHRONIZATION ............................................................8
   1.1.5 DOWNLOADABLE FPGA CONFIGURATION BIT-STREAM ....................8
   1.2 OVERVIEW SOFTWARE INTERFACE ..............................................9
      1.2.1 TELEGRAM RECEPTION ..........................................................9
      1.2.2 TRIGGER BLOCK .....................................................................10
   1.3 COUNTERS ..................................................................................10
      1.3.1 COUNTERS CONFIGURATION ....................................................13
         1.3.1.1 clocks ..............................................................................13
         1.3.1.2 starts ...............................................................................13
         1.3.1.3 output width ....................................................................13
         1.3.1.4 modes .............................................................................13
            1.3.1.4.1 single output, multiple output .......................................13
            1.3.1.4.2 Ext Start #2 as stop ..................................................13
         1.3.1.5 generation of bursts .......................................................13
         1.3.1.6 allowed configuration tables ............................................15
   2. **HANDS ON THE CTR (THE CTRTEST PROGRAM)** ......................16
      2.1 MODULE ENABLE .......................................................................16
      2.2 CTR MODULES INSTALLED .........................................................16
      2.3 VERSION ....................................................................................17
      2.4 STATUS REGISTER ....................................................................17
      2.5 RESET .......................................................................................17
      2.6 CABLE DELAY ...........................................................................18
      2.7 UTC ..........................................................................................18
      2.8 PLL ...........................................................................................18
      2.9 EVENT HISTORY .......................................................................19
      2.10 CENTRAL TIMING EVENTS (CTMS) .............................................19
      2.11 PERIPHERAL TIMING EVENTS (PTMS) .......................................20
      2.12 INTERRUPTS ............................................................................22
      2.13 COUNTERS ...............................................................................24
         2.13.1 CONFIGURATION EXAMPLE (GENERATING UTC SYNCRONOUS CLOCKS) ..........25
   3. **APPENDIX** ..............................................................................28
      3.1 MONITORING THE EXTERNAL CLOCK PERIOD ..........................28
      3.2 TIMING-FRAMES .......................................................................29
      3.3 TABLE OF CTRTEST COMMANDS .............................................30
   4. **REFERENCES** ..........................................................................31
1. MODULE OVERVIEW

The CTRP (PMC format), CTRI (PCI format) and CTRV (VME format) modules receive events through the General Machine Timing (GMT) link. This is a differential RS-485 network driven by the Timing Master (CTG) modules and distributed all around the accelerator complex.

Thanks to a set of AB-CO provided software libraries, the user can instruct the modules to perform a set of possible actions upon receiving a given event:

- Loading counters whose outputs will produce front panel pulses and/or bus interrupts.
- Producing an immediate front panel pulse.
- Producing an immediate bus interrupt.

These actions can also be triggered directly from software, i.e. without the need to wait for an event in the GMT cable.

The CTRP, CTRI and CTRV front panels consist of: a timing input, 50Ω adapted external clocks and external starts lemo inputs, and up to eight lemo outputs.

<table>
<thead>
<tr>
<th>Table 1 CTR platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format</strong></td>
</tr>
<tr>
<td>Counters</td>
</tr>
<tr>
<td>Interrupt Channels</td>
</tr>
<tr>
<td>Timing Input</td>
</tr>
<tr>
<td>50Ω Lemo-00 External Start Input</td>
</tr>
<tr>
<td>50Ω Lemo-00 External Clock Input</td>
</tr>
<tr>
<td>Lemo-00 outputs</td>
</tr>
<tr>
<td>Output LEDs</td>
</tr>
<tr>
<td>Ext Start LED</td>
</tr>
<tr>
<td>Ext Clock LED</td>
</tr>
<tr>
<td>Timing LED</td>
</tr>
<tr>
<td>Bus Access LED</td>
</tr>
<tr>
<td>IRQ LED</td>
</tr>
<tr>
<td>PLL Ok LED</td>
</tr>
<tr>
<td>CTR Ok LED</td>
</tr>
<tr>
<td>Enable LED</td>
</tr>
</tbody>
</table>
1.1 OVERVIEW HARDWARE INTERFACE

1.1.1 FRONT PANNEL

The CTR contains several connectors in its front panel:

- A connector for the GMT signal input. The electrical standard is RS-485.
- Lemo-00 connectors (SCEM 09.46.11.180.6) for the external start and the external clock inputs. The electrical standard is TTL, on-board terminated with 50 ohms.
- Three lemo-00 connectors for the outputs in the case of the CTRP and 4 in the case of the CTRI. These are driven by TTL 30 ohm drivers (Fast family).

The status of the card is described by the following LEDs:

- **Output LEDs** (CTRV, CTRI, CTRP): Output rising edge active.
- **Ext Start LED** (CTRV, CTRI, CTRP): Ext Start rising edge active.
- **Ext Clock LED** (CTRV, CTRI, CTRP): On if there has been a rising edge during last UTC second.
- **Timing LED** (CTRV, CTRI, CTRP): On if there have been 16 timing frames decoded during the last UTC second
- **Bus Access LED** (CTRV, CTRI, CTRP): On if there is READ/WRITE/INTERRUPT access.
- **IRQ LED** (CTRV): On if the CTR activates the interrupt request line.
- **PLL OK LED** (CTRV): On if the PLL has at least made an iteration and the 28 MSB of PLL Phase Error are equal to 0xFFFFFFFF or 0x000000.
- **Enable LED** (CTRV): The timing decoder is enabled.
- **CTR OK LED** (CTRV): FPGA is programmed.

1.1.2 CTRV JUMPER SETTINGS

In the CTRV you can find two different VME access:

- A16-D16 for the remote JTAG configuration. The base address for A16-A8 can be set using the switch SW2
- A24-D32 for the main functionality of the CTRV. The most significant nibble (A23-A20) is fixed to 0xC in the VHDL. The user can select the address lines A19-A15 using the switch SW1.

Jumpers X1->X2 for remote JTAG configuration should always be ON.

If an HPTDC is to be mounted on the CTRV, you should make sure the corresponding JTAG jumpers are mounted as shown in the following pictures.
1.1.2.1 CTRV VERSION 2

Drawing by Matthieu Cattin

**CTRV v2**

**First module:**
- A24 address = 0xC00000 (increment 0x10000)
- A16 address = 0x1000 (increment 0x100)

---

**2nd module:**
- A24 address = 0xC10000
- A16 address = 0x200

**3rd module:**
- A24 address = 0xC20000
- A16 address = 0x300

---

- Place this jumper if the HPTDC (IC21) is mounted.
- Place this jumper if the HPTDC (IC21) is NOT mounted.
1.1.2.2 CTRV VERSION 3

Drawing by Matthieu Cattin

**CTRV v3**

First module:

- A24 address = 0x0C000000 (increment 0x100000)
- A16 address = 0x100 (Increment 0x100)

2nd module:

- A24 address = 0x0C100000
- A16 address = 0x200

3rd module:

- A24 address = 0x0C200000
- A16 address = 0x300
1.1.3 COUNTERS

The module contains counters that can be loaded with predefined configurations upon arrival of a given GMT event or directly from the bus. Any combination of the outputs of these four counters can be routed to any of the output connectors through programming. To configure a counter, the user has several choices to make:

- Clock type: external (max. 50MHz), 40MHz internal, 10MHz internal, 1KHz internal. These last three clocks are derived from the GMT link and are therefore UTC-synchronous.
- Start type: next UTC millisecond, next UTC second, software, external or output of previous counter.
- Mode: single shot or repetitive (useful for dividing clocks and generating bursts).
- Delay 23 bits.
- Width 23 bits.

For a given output connector, the user can also select the polarity (positive or negative pulse).

1.1.4 UTC SYNCHRONIZATION

The CTR is synchronized at two levels:

- A mixed 40MHz Digital/Analog PLL locked to the 1MHz GMT carrier
- The second GMT frames indicate the Unix UTC to the CTR

This way the CTR is able to regenerate a 25ns resolution UTC locked time base.

Additionally, the optional mounting of an High Precision Time to Digital Converter chip (HPTDC) allows the outputs event tagging with a resolution of down to 1ns.

1.1.5 DOWNLOADABLE FPGA CONFIGURATION BIT-STREAM

In order to update the FPGA configuration, for revisions and bug corrections we can write directly through the bus to the EEPROM that programs the FPGA. This leaves open the possibility to easily change the logic.
1.2 OVERVIEW SOFTWARE INTERFACE

All communication with the CTRP is handled by a device driver and a C library provided by the AB-CO group. For users wanting to interface at a higher level, a FESA C++ class is provided. Platforms supported include PowerPC/LynxOS, Intel/LynxOS and Intel/Linux. The user gets from this library a set of services:

- Configuration of counters and outputs (see above).
- Subscription to interrupts.
- Current UTC time.
- Telegram information (a description of current and next machine cycles broadcast through the GMT link).
- Event history (a circular buffer containing the last events received through the GMT link).
- UTC time tags for the load time, start time and output time of every counter. These have a resolution of 25 ns and a precision of around 100 ns.
- Telegram Reception and buffers

1.2.1 TELEGRAM RECEPTION

The CTR is able to recognize telegram timing-frames from their header byte (see Table 4. Timing Frames). The first nibble is non-zero indicating the machine, and the second nibble is set to three to indicate it’s a Telegram for the given machine. For a telegram timing-frame, the CTR stores the Group-Value in that machine’s telegram-buffer at the word position indicated by the Group-Number. Thus, the CTR maintains telegram buffers for each machine. The telegram timing-frame to set the third CPS group to four would be 0x33030004. The CTR has two telegram buffers for each machine, the active buffer, and the incoming telegram buffer. When the Ready Telegram event

Figure 1. CTRP diagram
arrives (0x34FE00 for Ready CPS telegram), the CTR swaps the active and incoming telegram buffers for that machine; this usually happens on the last millisecond of a cycle (C_{i}) so that the telegram is already active by the start of the next cycle (C_{0}). The telegrams in the active buffers pilot the trigger blocks described above, and they can be read back from the host processor. Today the telegrams are limited to 32 Group Values of 16 bits each. The maximum machine number today is six; so six buffers of 32 16-bit words each can store the active telegrams, and another six telegram buffers store the incoming telegrams.

1.2.2 TRIGGER BLOCK

Table 2 Counter Trigger Blocks

<table>
<thead>
<tr>
<th>EVENT_CODE:</th>
<th>The event code that causes the counter to be loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>If this value is zero, the trigger is not used.</td>
<td></td>
</tr>
<tr>
<td>TGM_BLOCK:</td>
<td>An optional telegram condition</td>
</tr>
<tr>
<td>MACHINE:</td>
<td>The machine/accelerator ID (4-bits)</td>
</tr>
<tr>
<td>GROUP_NUMBER:</td>
<td>The number of the “group” in telegram (8-bits) (See –)</td>
</tr>
<tr>
<td>GROUP_VALUE:</td>
<td>The value of the “group” (16-bits)</td>
</tr>
<tr>
<td>COMPARE:</td>
<td>How the “group” is to be compared</td>
</tr>
<tr>
<td>COMP_EQUALITY:</td>
<td>{“group” == Telegram [group]}</td>
</tr>
<tr>
<td>COMP_LOGICAL_AND:</td>
<td>{“group” AND Telegram [group]! = 0}</td>
</tr>
<tr>
<td>COMP_NO_CHECK:</td>
<td>Ignore the telegram, just use the event</td>
</tr>
<tr>
<td>COUNTER_NUMBER:</td>
<td>The counter to which the configuration block belongs</td>
</tr>
<tr>
<td>This counter's configuration block is overwritten by the following block when the trigger condition is valid. If this value is zero no counters are used, and a bus interrupt is made directly from the timing-frame and telegram.</td>
<td></td>
</tr>
<tr>
<td>COUNTER_CONFIGURATION:</td>
<td>Write this block to the counter when triggered</td>
</tr>
<tr>
<td>The configuration is copied to the counter, if the counter is not in REMOTE.</td>
<td></td>
</tr>
</tbody>
</table>

The CTR is usually connected to a timing cable, and events arriving on this cable cause a counter to be loaded with a corresponding configuration block if the trigger trips. There is a pool of 2048 triggers. Each contains an EVENT_CODE, and an optional telegram condition. When the trigger is tripped, the configuration block is loaded into the specified counter.

1.3 COUNTERS

The CTRI and CTRP contain 4 23 bit counters (counters 1 to 4), which can be clocked by the internal UTC synchronous 40MHz and an external clock (0-50MHz). The CTRV contains two 4 23 bit counter blocks (8 counters in total), that can be clocked by the internal 40MHz or two external clocks (0-50MHz).
Additionally counter 0 can generate an immediate interrupt upon the arrival of a trigger and without waiting for the next millisecond.

Generally a counter is configured upon the arrival of an expected timing event. In this case, called **local control**, the configuration is stored in the CTR RAM. A counter can also be set in **remote control**, in this state the user can program directly the counter’s configuration registers and activate its output or interrupt line.

When a counter is loaded by an event or by a remote load, it waits for the next millisecond to validate its configuration, then it waits for its corresponding start. This start is driving the clock input of a flip-flop and must be resynchronized into the selected clock domain, therefore if the start is asynchronous to the selected clock domain there will be a jitter of 1 clock period in the output. If you use an asynchronous start to trigger two counters using the same clock, there will be a jitter of 1 clock tick between the outputs.
Figure 2. CTRP and CTRI Top Counter. The counters are chained forwards to provide to the next counter a chained clock or/and a chained start. There is also a backwards chaining to provide the stop to the previous counter. Any counter or clock can be routed to any output lemo. In the CTRV this chaining is only possible from counter 1 to 4 and 5 to 8 CTRV due to timing reasons.
1.3.1 COUNTERS CONFIGURATION

1.3.1.1 CLOCKS

The counters can be clocked using any of the following signals:
- 40MHz, 10MHz, 1KHz: Locked to the GMT millisecond.
- Ext Clk1: External signal. Up to 50MHz. Chained Clock (previous counter output)
- Ext Clk 2 (only CTRV)

1.3.1.2 STARTS

- 1KHz, 1PPS: Internally generated. Synchronous with UTC.
- Ext Start #1: User External Start.
- Ext Start #2: Only CTRV
- SelfStart: The counter is first started by the 1KHz signal, then it is restarted again when it arrives to zero.
- ChainedStart: Counter \(i-1\) enables counter \(i\). **Not valid for counter 4 to 5 in the CTRV**
- ChainedStart/SelfStart/ChainedStop*. This is a special mode used to generate bursts of a programmable period and phase. The start enable is given by channel \(i-1\), the stop signal is given by channel \(i+1\). **Not valid for counter 4 to 5 in the CTRV**
- Remote Start: The counter starts on the reception of a remote start. It does not have to be validated by a millisecond first.

1.3.1.3 OUTPUT WIDTH

The output width is controlled by a 24 bits counter clocked by the 40MHz VCXO. If the delay is made using an external clock the falling edge of the output should be 50ns before the next rising edge. The falling edge will also show a jitter of 25ns (positive polarity).

1.3.1.4 MODES

1.3.1.4.1 SINGLE OUTPUT, MULTIPLE OUTPUT

- **Mode single** stop the counter after it arrives to zero. A new load is necessary to generate new actions.
- **Mode multiple** restarts the counter when a start has been detected and also when the counter has arrived to zero.

1.3.1.4.2 EXT START #2 AS STOP

When the mode bit BURST is set to ONE, the counter is immediately stopped by Ext Start #2. **Only CTRV.**

1.3.1.5 GENERATION OF BURSTS

The generation of bursts in the CTR implies the use of three counters, **counter** \(i-1\) that provides the Burst Start, **counter** \(i\) that generates the burst itself, and **counter** \(i+1\) that stops **counter** \(i\).
Figure 3. Generating 1 burst of N pulses. Counter \( i \) has its start configured as ChainedStartStop and its mode as Single Output (Single Burst in this case). Counter \( i+1 \) has its clock configured as ChainedClock.

Figure 4. Multiple burst of N pulses. The same configuration as in Figure 3 but this time all the counters are configured as multiple output and Counter \( i+1 \) has its start set to SelfStart.

Any combination of clocks and starts is possible, but there are the following general constraints regarding the delays:

- Delay \( i-1 \) has to be >= 1 clock tick
- Delay \( i \) >= 2 clock ticks
- Delay \( i+1 \) >= 1
## 1.3.1.6 ALLOWED CONFIGURATION TABLES

### Single Mode Delay Range

<table>
<thead>
<tr>
<th>Clk vs Start</th>
<th>1KHz*</th>
<th>PPS*</th>
<th>Ext Start(#/2)</th>
<th>Self</th>
<th>ChStart/StartSelfStart</th>
<th>Remote</th>
<th>40MHz Chained***</th>
<th>Ext Clk #1 Chained***</th>
<th>Ext Clk #2 Chained***</th>
</tr>
</thead>
<tbody>
<tr>
<td>40MHz</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>10MHz*</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>1KHz*</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>40MHz Chained</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>Ext Clk #1</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>Ext Clk #1 Chained</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>Ext Clk #2</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
<tr>
<td>Ext Clk #2 Chained</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>0-0xFFFF</td>
<td>No</td>
<td>2-0xFFFF</td>
<td>0-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
<td>1-0xFFFF</td>
</tr>
</tbody>
</table>

*) 1KHz, 10MHz and PPS are signals derived from the 40MHz Clock
**) Actually a delay 0 on a chained start is allowed but this counter will not be able to propagate a new start to the next counter.
***) Delay 1 is allowed, but it will only work correctly if the start is synchronous with the clock domain. If they are not synchronous there will be a jitter of 1 clock tick between this counter and the previous counter.
2. HANDS ON THE CTR (THE CTRTEST PROGRAM)

The ctrtest program provides a manual interface to check the status of the ctr and configure it. Type `ctrtest` from the shell prompt to start it.

Once you are in the ctrtest program typing `h` will show you the ctrtest commands. The ctrtest syntax is simple and flexible. Commands are normally followed by a list of arguments. Parenthesis separate commands in the same line; adding a number before the opening parenthesis indicates the number of times this command should be executed. Some examples are:

- `mo` ➔ Displays module info
- `nm` ➔ Goes to next module
- `enb 1` ➔ Enables the timing reception of the active module
- `mo 1 3((enb 1) nm)` ➔ Goes to module 1, then do three times enable the timing and jump to the next module. If there are three modules in the chassis it will end at module 1.

2.1 MODULE ENABLE

The decoding of the timing signal can be disabled. When disabled the module will not start a new trigger search, the event history will not be written and the telegram info will not be modified.

**Beware! The outputs, the interrupts and the counters are not disabled!** If there is any counter downcounting it may generate a pulse or interrupt at its arrival to zero.

- `enb 1` ➔ Enables the timing reception of the active module
- `enb 0` ➔ Disables the timing reception of the active module

2.2 CTR MODULES INSTALLED

The command `mo` shows a list of cards installed in the chassis. It shows info about the bus installation and the timing cable identity. If you want to jump to a given module type `mo #module`. The command `nm` jumps to the next module and `pm` to the previous.

On a PCI platform we get:

```
pcgw12-Ctr[02]mo
Mod:1 Typ:CTR PCI Vendor:0x10dc Device:0x0300 PciSlot:0 Fpga:0xB00000 Plx:0x824C00 Cable:CPS_DEV Tgm:CPS<==
```

On a VME platform:

```
psdsc04-Ctr[02]mo
Mod:1 VME:0xdec00000 JTAG:0xdf000100 Vec:0xbb8 Lvl:2 Cable:SPS_DEV Tgm:SPS<==
```
2.3 VERSION

The command `ver` displays the Driver date, FPGA VHDL code date and the HPTDC version if present.

Without HPTDC we would get:

```
pcgw12:Cnt[02]ver
VHDL Compiled: [1112278091] Thu-31/Mar/2005 14:08:11
Drv Compiled: [1113895656] Tue-19/Apr/2005 07:27:36
ctrtest Compiled: Apr 18 2005 16:44:57
No HPTDC installed
```

With an HPTDC installed we get:

```
pcgw12:Cnt[08]ver
Drv Compiled: [1113895656] Tue-19/Apr/2005 07:27:36
ctrtest Compiled: Apr 18 2005 16:44:57
HPTDC Chip Version: 0x8470DACE Version: 1.3: OK
```

2.4 STATUS REGISTER

The module status register can be read by typing `rst`. For example

```
pcgw12:Cnt[08]rst
Mod:2 Status:Gmt-OK Pll-OK Xc1-ER Xc2-ER Slf-OK Enb-OK Tdc-ER Int-OK Bus-OK
```

Gmt-OK: The module is decoding frames from the cable
PLL-OK: The PLL has at least made an iteration and the 28 MSB of PLL Phase Error are equal to 0xFFFFFF or 0x000000
Xc1-ER Xc2-ER: The external clocks 1 and 2 are not present. No rising edge detected on these inputs during the last second
Slf-OK: The CTR FPGA has been programmed
Enb-OK: The timing reception has been enabled
Tdc-ER: No HPTDC mounted
Int-OK: Interrupts OK
Bus-OK: The bus is working OK

2.5 RESET

Typing `reset` in the test program prompt, the module FPGA receives a hard reset.

```
psdsc04:Cnt[03]reset
Reset Module: 1
```
2.6 CABLE DELAY

The CTR can compensate for cable delays of up to 125us. A CTR closer to the timing generator will be programmed with a greater input delay than a farther CTR, in such a way that any equal action or timetag will be simultaneous on both modules. The command `ind` displays the current delay. Adding a parameter will write a new value to the module, `ind #25nsTicks`. For example:

```
pcgw12:Ctr[04]ind
Module: 2 Input Delay: 4000 = 100.000000us
pcgw12:Ctr[05]ind 3800
Module: 2 Input Delay: 3800 = 95.000000us
```

2.7 UTC

The CTR contains a UTC time base and a register with the current C time. The C time is the payload of events received by the CTR with a 0x01 header, also called C events. These events are broadcast every ms, with their payload being incremented by one every time.

The command `utc` displays these values. In case the CTR is not connected to any timing cable, the command `utc 1` copies the system UTC to the module.

```
pcgw12:Ctr[06]utc
UTC: Fri-22/Apr/2005 10:17:36.7864138750 C:0065
pcgw12:Ctr[07]utc 1
UTC: Set time: Fri-22/Apr/2005 10:18:33
UTC: Fri-22/Apr/2005 10:17:44.4879194500 C:0566
```

2.8 PLL

The PLL parameters can be read by the command `pll`.

```
pcgw12:Ctr[05]pll
Error      : [0xfffffffe] -2
Integrator : [0xe69fefd0] -425726000
Dac        : [0xffffe6a0] -6496
LastItLen  : [0x000001c6] 454
Phase      : [0x00001194] 4500
NumAverage : [0x000000c8] 200
KP         : [0x00008000] 32768
KI         : [0x000003e8] 1000
AsPrdNs    : 22.353361 (ns)
Error*Period/Naverage  : -0.223534ns
Phase*Period/Naverage  : 502.950625ns
```

Phase in the PLL is measured by counting ticks of an independent quartz oscillator between the rising edge of the incoming GMT signal and a regenerated 1MHz clock. These measurements are averaged after a given number measurements (`NumAverage`). `Error*Period/Naverage : -0.223534ns` indicates that the last phase
error estimate was of -0.223534n. Phase*Period/Naverage : 502.950625ns indicates the set phase of the regenerated 1MHz respect to the GMT signal 1MHz carrier.

Delays smaller than 25ns can be compensated changing the PLL phase parameter. The PLL phase value can be calculated with the formula:

\[ \text{PhaseNavTicks} = \left( \frac{\text{NumAverage}}{\text{AsPrdNs}} \right) \times \text{PhaseNs} \]

\( \text{AsPrdNs} \): The period of the phase detector oscillator in ns
\( \text{PhaseNavTicks} \): PLL phase in AsPrdNs
\( \text{NumAverage} \): Number of phase samples averaged in the phase detector
\( \text{PhaseNs} \): Desired phase in ns (should be between 475ns/525ns)

Example: 4500ticks = (200samples/22.253361ns)*502.95ns

2.9 EVENT HISTORY

The CTR stores the incoming timing events, (excepting the millisecond frame) into a 1024 cells circular buffer. The command `reh [\#cells]` displays the last \#cells entries.

```
psdsc04:Cnt[04]reh 10
```

[C347] Ctm:0986/0x2F220101 C3319 Wed-18/May/2005 16:58:36.5901879500 ;Simple event: OEX.WWE-MTG
[C346] Ctm:0985/0x2F250101 C3219 Wed-18/May/2005 16:58:36.4901879500 ;Simple event: OEX.EBT-MTG
[C345] Ctm:0000/0x0ACB0200 C2929 Wed-18/May/2005 16:58:36.2006879500 ;Cable:SPS_DEV(203) TgvMch:2
[C343] Ctm:0000/0xB500581C C2727 Wed-18/May/2005 16:58:35.9986879500 ;UTC Low [22556]
[C342] Ctm:0404/0x64020000 C2662 Wed-18/May/2005 16:58:35.9331879500 ;Simple ADE event DX.SCY-MTG
[C341] Ctm:0400/0x64FE0000 C2661 Wed-18/May/2005 16:58:35.9321879500 ;Simple ADE event DX.RPLS-MTG
[C340] Ctm:0320/0x24040000 C2399 Wed-18/May/2005 16:58:35.6701879500 ;Simple SPS event SX.SBP-MTG
[C339] Ctm:0500/0x244E0000 C2398 Wed-18/May/2005 16:58:35.6691879500 ;Simple SPS event SX.RPLS-MTG
[C338] Ctm:0321/0x24140000 C2369 Wed-18/May/2005 16:58:35.6401879500 ;Simple SPS event SX.APPACQ-MTG

The columns contents are the following:
1. Arrival order, 1 being the last one
2. Address in the Event History Buffer,
3. Frame
4. C-Time
5. UTC
6. Description of the decoded frame

2.10 CENTRAL TIMING EVENTS (CTMS)

Central Timing Events (CTMs) are seen by the timing generator as objects described by a name, a number, an event code, a repetition time, etc. On the receiver side CTMs
are described by a number, an event code and a name. The command `ctmr` verifies the actual cable id and looks the corresponding CTMs up. The cable is a number indicating which GMT network the CTR is connected to. The CTR gets it by decoding the cable id event.

```
pcgw12:Ctr[05]ctmr
Created: CTIM: HX.SSC-MTG (0504) 0x14040000 OK
Created: CTIM: HX.SCY-MTG (0505) 0x14050000 OK
Created: CTIM: HX.APPACQ-MTG (0510) 0x140B0000 OK
Created: CTIM: HX.SBP-MTG (0512) 0x140A0000 OK
Created: CTIM: HX.RPLS-MTG (0600) 0x14FE0000 OK
Created: CTIM: MX.CTRIG (0911) 0x0100FFFF OK
Created: 6 CTIM equipments OK
```

The command `ctm` edits the existing CTMs. "?" shows you a list of possible commands and "." exits from the editor.

```
pcgw12:Ctr[03]ctm
[0]Ctm:504 Fr:0x14040000 ;Simple LHC event HX.SSC-MTG :
[1]Ctm:505 Fr:0x14050000 ;Simple LHC event HX.SCY-MTG :
                      +_____________+<CrLf>
                      Next CTIM equipment
                      <Index> Go to entry Index
                      ? Print this help text
                      . Exit from the editor
                      f<Frame> Change CTIM Frame
                      x Delete CTIM equipment
                      y<Id>,<Frame> Create CTIM equipment
```

The first column indicates the CTM entry index; the second column the CTM number; the third the event code and finally there is a short description ended by the CTM name.

2.11 PERIPHERAL TIMING EVENTS (PTMS)

Peripheral Timing Events (PTMs) are counter actions triggered by a CTM and an optional telegram condition. For example, generate a pulse of 1us width one millisecond after the arrival of HX.SCY-MTG.

PTMs can be loaded from a file (command `ptmr` [FileName]) or created manually (`ptm`). The PTMs can be stored into a file by `ptmw` [FileName]) The command `ea` is a quick way to edit existing PTMs actions, but it will not allow you to create new ones. The command `la` shows a list of the current PTMs. Remember that you have to associate CTMs to PTMs. This means that you have to load the current CTMs before
creating any PTM. If your PTMs are stored in the default PTM description file (/tmp/Ctr.info), the command `load` simplifies this task by merging in one command `ctmr`, `ptmrf` and the HPTDC configuration if mounted. For example:

```plaintext
pcgw12:Ctr[01].ptm
>>>Ptm:None defined : y 1002 1 1
>>>Ptm:1002 Mo:2 Ch:1 Sz:1 St:1 Created
[1]Ptm:1002 Mo:2 Ch:1 Sz:1 St:1 : y 3001 3 2
>>>Ptm:3001 Mo:2 Ch:3 Sz:2 St:2 Created
[1]Ptm:1002 Mo:2 Ch:1 Sz:1 St:1 : a
[1]Ptm:1002[Ctm:107(34:07:0000)][Chn:1 Str:Nor Mde:Once Clk:1KHz 1#400 Out] : ?
<CrLf> Next action
/<Action Number> Open action for edit
? Print this help text
. Exit from the editor
i<CTIM> Change CTIM trigger number CTIM
f<Frame> Change frame Frame
l<Trigger Condition> Change trigger condition ^%/&
m<Trigger Machine> Change trigger machine LHC/SPS/CPS/LEI/ADE
r<Trigger Group> Change trigger group number Group Number
g<Trigger group value> Change trigger group value Group Value
s<Start> Change Start 1KHz/Ext1/Ext2/Chnd/Self/Remt/Pps/Chnd+Stop
k<Clock> Change Clock 1KHz/10MHz/40MHz/Ext1/Ext2/Chnd
o<Mode> Change Mode Once/Mult/Brst/Mult+Brst
w<Pulse Width> Change Pulse Width Pulse Width
v<Delay> Change Delay Delay
e<enable> Enable or Disable output 1=Enable/0=Disable
```

`y 1002 1 1` creates a PTM of size 1 (one action) called 1002 that configures counter #1
`y 3001 3 2` creates a PTM of size 2 (two actions) called 3002 that configures counter #3

As we were on module #2 all these PTMs were assigned to it. If you want to edit a ptm on given module you will have to jump to that module using the ctrtest commands `mo`, `nm` or `pm`.

As usually, “?” shows the PTM editor list of commands and “.” exits. “a” opens the actions editor.

Once in the actions editor, “?” will show you again a list with all the possible actions.

- **i#CTM**: selects the CTM that triggers the counter
• **t#Condition**: selects the telegram condition, 0 no condition, 1 equality, 2 bit wise and-gate (checks that the selected bits are set to one)
• **m#Machine**: machine telegram (1 LHC, 2 SPS ...)
• **n#Group**: group number to be checked
• **g#Value**: Value or pattern to be found
• **s#Start**: Counter Start (0 Millisecond, 1 Ext Start1...)
• **k#Clock**: Counter Clock (0 Millisecond, 1 10MHz ...)

The `ctrtest` command `ea` opens directly the action editor dialog for every PTM. The counter configuration and its possibilities will be discussed on 2.13

### 2.12 INTERRUPTS

The possible interrupts in the CTR are:

- PPS, Millisecond
- Counter at Zero
- PLL iteration
- New GMT Frame decoded
- CTIM event (Counter 0 triggered by a given event)
- PTIM event (Counter triggered by a given event)
- Trigger Match found

If the driver is waiting for a particular PTIM or CTIM interrupt first it waits for a counter interrupt, once it has received this interrupt it checks that the counter has been configured by the expected event code.

For example, on the LHC cable we get:

```
pcgw12: Ctr[20] wi c 504
```

If the CTM event doesn't arrive before a timeout, the test program will display an error message. For example, if we set the timeout to 1 second, and wait 15 times for an event of period 5 seconds we get:

```
pcgw12: Ctr[23] tmo 1000
Timeout: [1000] Enabled
pcgw12: Ctr[24] 15(wi c 504)
Time out or Interrupted call
Time out or Interrupted call
Time out or Interrupted call
Time out or Interrupted call
```

```
Time out or Interrupted call
Time out or Interrupted call
Time out or Interrupted call
Time out or Interrupted call
```

```
Time out or Interrupted call
Time out or Interrupted call
Time out or Interrupted call
```

```
Time out or Interrupted call
```
**Time out or Interrupted call**

cpgw12:Ctr[01]

The interrupt timeout can be disabled by **tmo 0**. If there is a burst of interrupts, these can be queued by setting the queue flag to zero **qf 0**.

An interesting feature is that you can connect to different PTMs or CTMs simultaneously or not and wait for the first to happen. This way you can easily debug your timings if their frequency is “too high”. For example, let’s generate three interrupts every millisecond, with a delay between channels of 25ns

```
NoQueueFlag: ReSet, Queuing is On
QueueSize: 64
QueueOverflow: 53802
```

\[2\]Ptm:1002[Ctm:504(14:04:0000)][Chn:1 Str:Nor Mde:Mult Clk:40MHz 0#400 Out]
----------
\[3\]Ptm:2002[Ctm:504(14:04:0000)][Chn:2 Str:Nor Mde:Mult Clk:40MHz 1#400 Out]
----------
\[4\]Ptm:3002[Ctm:504(14:04:0000)][Chn:3 Str:Nor Mde:Mult Clk:40MHz 2#400 Out]

```
pcgw12:Ctr[12]la
pcgw12:Ctr[13]wi p 1002
pcgw12:Ctr[14]wi p 3002
Aborted wait loop after 64 iterations
```

You can verify that the delay between outputs was effectively 25ns and the repetition period 1ms. If we had had queuing off we would have missed PTM[3002] and PTIM[2002].
If the ISR is servicing to fast and slow interrupts you can get error messages like “Aborted wait loop after n iterations”. For example:

```
pcgw12:Ctr[0] tmo 0
Timeout: [0] Disabled
pcgw12:Ctr[10] wi 0x1000
Aborted wait loop after 64 iterations
pcgw12:Ctr[12] wi
pcgw12:Ctr[13]
```

Notice also that when we have waited for the 1KHz interrupt (wi 0x1000) we have obtained an interrupt time offset of approximately 7550ns. This is because the ISR time-tags the “non-counter interrupts” by reading the CTR UTC which is increasing every 25ns. PTMs and CTMs generate “counter interrupts” which are time-tagged on their rising-edge in the CTR.

### 2.13 COUNTERS

There are two different counter configuration methods depending if a counter is in remote or local state.

- **Local**: Default value. Creating PTM with the command `ptm`, or `ea` the PTM is already created. These commands have already been explained in 2.10 and 2.11
- **Remote**: The command `cnf` will look for counters in remote mode, and show you an edit dialog for each one of them.

As we have already mentioned, in the CTRP, CTRI there are 4 counters, and 8 in the CTRV. You can select the counter you want to talk to by the commands, `ch <Channel>`, `nch` to go to the next channel and `pch` to jump to the previous channel. You can change a counter into remote/local state with the command `rem [1/0]`.

For example:

```
pcgw12:Ctr[12] ch 1 rem 0 3(nch rem 1)
Active Counter: 1
Counter: 1 Local control from CTR card
Counter: 2 Remote controlled from host
Counter: 3 Remote controlled from host
Counter: 4 Remote controlled from host
pcgw12:Ctr[13] ch
Active Counter: 4
```

We have jumped to counter 1, we have set it to local control, and then we have done three times jump to the next channel and set it to remote control. By typing `ch` you can check the active counter.
2.13.1 CONFIGURATION EXAMPLE (GENERATING UTC SYNCRONOUS CLOCKS)

Let's generate now a 1KHz clock in channel 2, a 500Hz clock in channel 3 and a 100Hz clock in channel 3.

```
psdsc04:Cnt[22]cnf
Counter: 1 Not in Remote
[Ch:2] St:Nor Mo:Once Ck:1KHz 0(0) NoOut : s4o1v1b1
[Ch:2] St:Self Mo:Mult Ck:1KHz 1(0) Bus :
[Ch:3] St:Nor Mo:Once Ck:1KHz 0(0) NoOut : s4o1v2b1
[Ch:3] St:Self Mo:Mult Ck:1KHz 2(0) Bus :
[Ch:4] St:Nor Mo:Once Ck:1KHz 0(0) NoOut : s4o1v10b1
[Ch:4] St:Self Mo:Mult Ck:1KHz 10(0) Bus :
Counter: 5 Not in Remote
Counter: 6 Not in Remote
Counter: 7 Not in Remote
Counter: 8 Not in Remote
```

```
We can see that Counter #2 appears every 1ms, Counter #3 every 2ms and Counter #4 every 10ms. Everything is fine... **as long as you don't care about phase!**

If you care about phase, the easiest is to create a PTM that generates a clock on the arrival of a known phase event code (as for example the UTC event 0xB600FFFF).

```plaintext
psdsc04: Ctr[18] ptm
>>> Ptm: 1000 Mo: 1 Ch: 2 Sz: 1 St: 2 Created
[1] Ptm: 1000 Mo: 1 Ch: 1 Sz: 1 St: 1 : a
```

Now that we have a 100KHz clock of known phase we can use this counter output to start the next counter. Setting a delay in the second counter you will be able to adjust the phase. For example, let’s generate a 100Hz with a 25us phase respect to the PPS.

```plaintext
psdsc04: Ctr[19] ptm
[1] Ptm: 1000 Mo: 1 Ch: 1 Sz: 1 St: 1 : y 2000 2 1
>>> Ptm: 2000 Mo: 1 Ch: 2 Sz: 1 St: 2 Created
[1] Ptm: 1000 Mo: 1 Ch: 1 Sz: 1 St: 1 :
[2] Ptm: 2000 Mo: 1 Ch: 2 Sz: 1 St: 2 :
```

**WARNING!** Once the clocks have been started they should be set in remote mode! Otherwise the next match will reconfigure the counters and stop them for a while (some hundreds of microseconds!). The configuration can be automatically done at start up adding the next command to `/etc/rc.local`. 

```plaintext
```

---

**WARNING!** Once the clocks have been started they should be set in remote mode! Otherwise the next match will reconfigure the counters and stop them for a while (some hundreds of microseconds!). The configuration can be automatically done at start up adding the next command to `/etc/rc.local`. 

```plaintext
```
echo "(mo 1) load (enb 1)(wi p 1000) (ch 1) 2(rem 1 nch)(wi 0)q" | ctrtest >/dev/null

The counter configuration should be stored in /tmp/Ctr.info
3. APPENDIX

3.1 MONITORING THE EXTERNAL CLOCK PERIOD

To measure the external clock period you just have to divide your clock by a given number of counts, \#Counts, in order to generate a periodic interrupt. Connecting to the counter interrupt will provide you the On-Zero UTC. The period will be given by:

$$\text{Period} = \frac{\text{UTC\_INT(N)} - \text{UTC\_INT(N-1)}}{\#\text{Counts}}$$

For example:

```plaintext
psdsc04:Ctr[17]ch 1
Active Counter: 1
psdsc04:Ctr[18]rem 1
psdsc04:Ctr[19]cnf
[Ch:1] St:Nor Mo:Once Ck:1KHz 0(400) NoOut : s4o1k3b1v1000000
[Ch:1] St:Self Mo:Mult Ck:Ext1 1000000(400) Bus : .
psdsc04:Ctr[20]qf 1 10(wi 0x2)
NoQueueFlag: Set, Queuing is Off
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:42.8804294750] Cntr1 Hptdc: 0x8657C6C0
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:42.9064628750] Cntr1 Hptdc: 0x8A50B4C0
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:42.9324962500] Cntr1 Hptdc: 0x92428E40
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:42.9585294250] Cntr1 Hptdc: 0x963B7C40
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:42.9845628350] Cntr1 Hptdc: 0x019DE940
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:43.0366294500] Cntr1 Hptdc: 0x098FC300
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:43.0626627000] Cntr1 Hptdc: 0x0D88AF40
Mod[1] Int[0x002 1] Time[Mon-02/May/2005 10:03:43.0886959250] Cntr1 Hptdc: 0x11819D00
```

Depending on the integration period you chose you may get different period values,

Table 3 Ext Clock period over time (Philips PM5716 pulse generator).

<table>
<thead>
<tr>
<th>UTC_INT (s)</th>
<th>((N) - (N-1))/#\text{Ticks (ns)})</th>
<th>((N) - (N-2))/(2\cdot#\text{Ticks (ns)})</th>
<th>((N) - (N-3))/(3\cdot#\text{Ticks (ns)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>42.88042948</td>
<td>26.033275</td>
<td>26.0333875</td>
<td>26.03331667</td>
</tr>
<tr>
<td>42.90646288</td>
<td>26.033375</td>
<td>26.0333875</td>
<td>26.03331667</td>
</tr>
<tr>
<td>42.93249625</td>
<td>26.033175</td>
<td>26.0333275</td>
<td>26.03331667</td>
</tr>
<tr>
<td>42.95852943</td>
<td>26.0332875</td>
<td>26.03333125</td>
<td>26.03334167</td>
</tr>
<tr>
<td>43.01059613</td>
<td>26.033335</td>
<td>26.033329167</td>
<td>26.03334167</td>
</tr>
<tr>
<td>43.03662945</td>
<td>26.0332325</td>
<td>26.03329167</td>
<td>26.03334167</td>
</tr>
<tr>
<td>43.0626627</td>
<td>26.0332875</td>
<td>26.033329167</td>
<td>26.03334167</td>
</tr>
<tr>
<td>43.08869593</td>
<td>26.0332375</td>
<td>26.03332667</td>
<td>26.03334167</td>
</tr>
</tbody>
</table>
### 3.2 TIMING-FRAMES

At the time of writing this user guide, the timing-frame layout is as follows...

Table 4. Timing Frames

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Used in LHC</th>
<th>Nib1</th>
<th>Nib2</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>spsSscEvent</td>
<td>MCH</td>
<td>0</td>
<td></td>
<td>Super-Cycle Number</td>
<td>Binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsMillisecondEvent</td>
<td>0x0</td>
<td>1</td>
<td>Millisecond</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adeMillisecondEvent</td>
<td>0x0</td>
<td>1</td>
<td>Millisecond</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsSecondEvent(legacy)</td>
<td>0x0</td>
<td>2</td>
<td>Second</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsMinuteEvent (legacy)</td>
<td>0x0</td>
<td>3</td>
<td>Minute</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsHourEvent (legacy)</td>
<td>0x0</td>
<td>4</td>
<td>Hour</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsDayEvent (legacy)</td>
<td>0x0</td>
<td>5</td>
<td>Day</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsMonthEvent (legacy)</td>
<td>0x0</td>
<td>6</td>
<td>Month</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsYearEvent (legacy)</td>
<td>0x0</td>
<td>7</td>
<td>Year</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTCHigh (validates UTCLow on next ms)</td>
<td>0xb</td>
<td>6</td>
<td>00</td>
<td>UTC(32 downto 16)</td>
<td>Binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTCLow</td>
<td>0xb</td>
<td>5</td>
<td>00</td>
<td>UTC(15 downto 0)</td>
<td>Binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TelegramEvent</td>
<td>MCH</td>
<td>3</td>
<td>GroupNumber</td>
<td>GroupValue</td>
<td>Binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MtgIdentEvent</td>
<td>0x0</td>
<td>A</td>
<td>CableIdent</td>
<td>Binary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>psCDownEvent</td>
<td>0x8</td>
<td>1</td>
<td>Millisecond to endCycle</td>
<td>BCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineEvent</td>
<td>MCH</td>
<td>4</td>
<td>Code</td>
<td>Binary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tcuRepetitiveEvent</td>
<td>MCH</td>
<td>5</td>
<td>GroupNumber</td>
<td>GroupTyp</td>
<td>BCD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spsMachineEvent</td>
<td>MCH</td>
<td>1</td>
<td>Code</td>
<td>CycleTyp</td>
<td>CycleNum</td>
<td>BCD</td>
<td></td>
</tr>
<tr>
<td>spsSuperCycleEvent</td>
<td>MCH</td>
<td>2</td>
<td></td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MCH: Machine Number Network**

<table>
<thead>
<tr>
<th>MCH</th>
<th>Number</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHC</td>
<td>1</td>
<td>CERN</td>
</tr>
<tr>
<td>SFS</td>
<td>2</td>
<td>CERN</td>
</tr>
<tr>
<td>CPS</td>
<td>3</td>
<td>CERN</td>
</tr>
<tr>
<td>FSB</td>
<td>4</td>
<td>CERN</td>
</tr>
<tr>
<td>Not Used</td>
<td>5</td>
<td>CERN</td>
</tr>
<tr>
<td>ADE</td>
<td>6</td>
<td>CERN</td>
</tr>
<tr>
<td>SCT</td>
<td>3</td>
<td>CTF3</td>
</tr>
<tr>
<td>FCT (100Hz)</td>
<td>4</td>
<td>CTF3</td>
</tr>
</tbody>
</table>

**GroupTyp Number**

<table>
<thead>
<tr>
<th>GroupTyp</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCLUSIVE</td>
<td>1</td>
</tr>
<tr>
<td>BIT-PATTERN</td>
<td>2</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>4</td>
</tr>
</tbody>
</table>

**Ready Telegram**: is a MachineEvent with the code = 0xFE

**MachineEvent**

| MCH | 4   | 0xFE |

---

**MCH: Machine Number Network**

<table>
<thead>
<tr>
<th>MCH</th>
<th>Number</th>
<th>Network</th>
<th>GroupTyp</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHC</td>
<td>1</td>
<td>CERN</td>
<td>EXCLUSIVE</td>
<td>1</td>
</tr>
<tr>
<td>SFS</td>
<td>2</td>
<td>CERN</td>
<td>BIT-PATTERN</td>
<td>2</td>
</tr>
<tr>
<td>CPS</td>
<td>3</td>
<td>CERN</td>
<td>NUMERIC</td>
<td>4</td>
</tr>
</tbody>
</table>

**Ready Telegram**: is a MachineEvent with the code = 0xFE

**MachineEvent**

| MCH | 4   | 0xFE |

---
### 3.3 TABLE OF CTRTEST COMMANDS

#### Table 5. ctrtest commands

<table>
<thead>
<tr>
<th>Category</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENERAL</strong></td>
<td><strong>h</strong></td>
<td>HELP</td>
</tr>
<tr>
<td></td>
<td><strong>s</strong></td>
<td>Sleep for #seconds</td>
</tr>
<tr>
<td></td>
<td><strong>his</strong></td>
<td>Shows command history</td>
</tr>
<tr>
<td></td>
<td><strong>rst</strong></td>
<td>Reads status</td>
</tr>
<tr>
<td></td>
<td><strong>reset</strong></td>
<td>Activates the global reset line of the FPGA</td>
</tr>
<tr>
<td></td>
<td><strong>load</strong></td>
<td>Installs CTMs, installs PTMs and configures the HPTDC device if mounted</td>
</tr>
<tr>
<td></td>
<td><strong>memtst</strong></td>
<td>Tests RAM</td>
</tr>
<tr>
<td></td>
<td><strong>jtag</strong></td>
<td>Sends a new configuration bit-stream to the CTR</td>
</tr>
<tr>
<td><strong>TIMING</strong></td>
<td><strong>enb</strong></td>
<td>Enables/disables timing. Disabled by default</td>
</tr>
<tr>
<td></td>
<td><strong>ind</strong></td>
<td>Displays gmt input delay/ Writes gmt input delay to module</td>
</tr>
<tr>
<td></td>
<td><strong>utc</strong></td>
<td>Displays UTC/Sets UTC</td>
</tr>
<tr>
<td></td>
<td><strong>reh</strong></td>
<td>Displays last event history entries</td>
</tr>
<tr>
<td></td>
<td><strong>rtg</strong></td>
<td>Reads telegram</td>
</tr>
<tr>
<td><strong>INTERRUPTS</strong></td>
<td><strong>wi</strong></td>
<td>Help</td>
</tr>
<tr>
<td></td>
<td><strong>c</strong></td>
<td>Waits for a given CTM</td>
</tr>
<tr>
<td></td>
<td><strong>p</strong></td>
<td>Waits for a given PTM</td>
</tr>
<tr>
<td></td>
<td><strong>mask</strong></td>
<td>Connects to interrupt. Type “wi ?” to see possible interrupt sources</td>
</tr>
<tr>
<td></td>
<td><strong>0</strong></td>
<td>Disconnects from all interrupts, PTMs and CTMs included</td>
</tr>
<tr>
<td></td>
<td><strong>imo</strong></td>
<td>Displays/ Disables/Set timeout value</td>
</tr>
<tr>
<td></td>
<td><strong>qf</strong></td>
<td>Displays qf status/ Enables interrupt queuing/ Disables interrupt queuing</td>
</tr>
<tr>
<td></td>
<td><strong>cl</strong></td>
<td>Clients connected to interrupts</td>
</tr>
<tr>
<td></td>
<td><strong>lat</strong></td>
<td>Driver latency</td>
</tr>
<tr>
<td><strong>CTM/PTM</strong></td>
<td><strong>ctm</strong></td>
<td>Opens the CTM editor</td>
</tr>
<tr>
<td></td>
<td><strong>ctmr</strong></td>
<td>Reads CTMs</td>
</tr>
<tr>
<td></td>
<td><strong>ptm</strong></td>
<td>Opens the PTM editor</td>
</tr>
<tr>
<td></td>
<td><strong>ptmrf</strong></td>
<td>Reads PTMs from a file</td>
</tr>
<tr>
<td></td>
<td><strong>ptmwf</strong></td>
<td>Writes PTMs to a file</td>
</tr>
<tr>
<td></td>
<td><strong>la</strong></td>
<td>Lists PTMs</td>
</tr>
<tr>
<td></td>
<td><strong>ea</strong></td>
<td>Edits PTMs actions</td>
</tr>
<tr>
<td><strong>PLL</strong></td>
<td><strong>pll</strong></td>
<td>Displays/ Writes new parameters into the pll and resets it</td>
</tr>
<tr>
<td></td>
<td><strong>plot</strong></td>
<td>Plots the PLL error versus UTC time onto a file and optionally to a graphic window</td>
</tr>
<tr>
<td></td>
<td><strong>TimeLength, [1/0]</strong></td>
<td>Writes new parameters into the pll, resets it and plots the PLL error onto a file and optionally to a graphic window</td>
</tr>
<tr>
<td><strong>COUNTERS</strong></td>
<td><strong>ch</strong></td>
<td>Displays current counter/ Jumps to counter</td>
</tr>
<tr>
<td></td>
<td><strong>nch</strong></td>
<td>Jumps to next counter</td>
</tr>
<tr>
<td></td>
<td><strong>pch</strong></td>
<td>Jumps to previous counter</td>
</tr>
<tr>
<td></td>
<td><strong>omsk</strong></td>
<td>MASK, [1 -logic/0 +logic] Displays/Writes channel output mask, and polarity</td>
</tr>
<tr>
<td></td>
<td><strong>chis</strong></td>
<td>Shows current counter history (trigger, start, and interrupt times)</td>
</tr>
<tr>
<td></td>
<td><strong>rem</strong></td>
<td>Counter in remote mode/not in remote mode</td>
</tr>
<tr>
<td></td>
<td><strong>cnf</strong></td>
<td>Edit counter configuration (Delay, Width, Start, Mode, Clock, Out enabled, Interrupt Enabled) Counter must be in remote mode</td>
</tr>
<tr>
<td></td>
<td><strong>rcm</strong></td>
<td>Help/0x01=Load 0x02=Stop 0x04=Start 0x08=Out 0x10=Bus Counter must be in remote mode</td>
</tr>
</tbody>
</table>
4. REFERENCES

[1] Nanosecond level UTC timing generation and stamping in CERN’s LHC
J. Serrano, P. Álvarez, D. Dominguez, J. Lewis, ICALEPCS-03 Korea

[2] The evolution of the CERN SPS timing system for the LHC era, ICALEPCS-03
J. Lewis, Jean-Claude Bau, Javier Serrano, David Dominguez, Pablo Alvarez.
ICALEPCS-03 Korea

[3] PLL usage in the general machine timing system for the LHC
P. Álvarez, D. Dominguez, J. Lewis, Q. King, J. Serrano, B. Todd. ICALEPCS-03 Korea
http://icalepcs2003.postech.ac.kr/Proceedings/PAPERS/MP532.PDF