# MCB developer notes

Otger Ballester, Laia Cardiel Sas

August 9, 2010

DARK ENERGY
SURVEY

Institut de Física
d'Altes Energies

**Document Version**: 1.02

**MCB Hardware described**: 1.2

**MCB firmware version described**: 5.12

# Contents

## Preface

This document describes the hardware implementation of the Master Control Board made at Barcelona, Revision 1.2, that boots the firmware revision 5.0 and later. Minor revisions and modifications to the capabilities and functionality of this board can be added as an appendix to this document.

## Document Scope

This document provides an overall description, as well as detailed information, on the architecture, configuration, testing and functionality of the MONSOON Master Control Board (MCB). It is intended that this document be read by anyone who needs to consider, build, use, or test a MONSOON system that requires this hardware module.

This document is an adaptation of MNSN-AD-08-0001 to the MCB HW and FW versions developed at IFAE (Barcelona) for the DES project.

If you find some error or some section is not well explained contact Otger Ballester (otger@ifae.es) or Laia Cardiel (laia@ifae.es), we will do our best to help.

To diferentiate between boards, when I write about the old MCB board developed at NOAO I will call it NOAO MCB.

# 1 Board Functional Description

## 1.1 Purpose of Board

The Master Control Board (MCB) provides the essential control and communication functions required by the detector head electronics (DHE). Every MONSOON DHE chassis requires just one MCB that must reside in the 'SYSTEM' slot or 'Slot 1' of the PCI backplane.

This board provides the interface for communication between the Pixel Acquisition Node (PAN) and the DHE electronics. It controls the unidirectional Sequencer Bus on the backplane and provides a programmable sequencer (SEQUENCER MPU) to control DHE functions independently of the PAN to support detector functions. It also provides for DHE clock distribution and a mechanism to synchronize multiple DHE chassis. All communication, control and sequencing functions are built from firmware contained in a Xilinx virtex 4 fpga.

## 1.2 Interface Description

### 1.2.1 Slink Interface

The MCB provides a seat for an Slink fiber optic CMC module using connector J6. The Slink module provides a bi-directional fiber optic link to the Pixel Acquisition Node (PAN). All communication with the PAN is through this link.

**ToDo:** Add more info about Slink implementation, data paths,....

### 1.2.2 Clock Distribution

The MCB controls the activity of peripheral boards by enabling or disabling the system clock to them. This allows boards that are not specifically required during some periods to shut down to conserve power or reduce induced noise from their activities.

Control of this interface is by directly writing to the clock enable register of the MCB (at address `0x0100`).

Because of some non-standardization within the cPCI specification, some peripheral boards share common clocks. Tables 1 and 2 map the control word written to the clock enable register to physical peripheral boards.

Writing a 'one' to the respective bit in the clock enable register turns that clock source on.

Table 1: Clock Distribution Control Bit Assignments for 8-Slot Backplanes

| Control Word Bit | Backplane Signal | Peripheral Board Clocks |
|---|---|---|
| 0 | | Not used, always active |
| 1 | CLK23 | Board 2 and 3 active |
| 2 | CLK23 | Board 2 and 3 active |
| 3 | CLK45 | Board 4 and 5 active |
| 4 | CLK45 | Board 4 and 5 active |
| 5 | CLK6 | Board 6 active |
| 6 | CLK7 | Board 7 active |
| 7 | CLK8 | Board 8 active |
| 15 | BKPLN_SLCT | Set high for 8-slot backplane mapping |

Table 2: Clock Distribution Control Bit Assignments for 6-Slot Backplanes

| Control Word Bit | Backplane Signal | Peripheral Board Clocks |
|---|---|---|
| 0 | | Not used, always active |
| 1 | CLK2 | Board 2 and 3 active |
| 2 | CLK3 | Board 2 and 3 active |
| 3 | CLK4 | Board 4 and 5 active |
| 4 | CLK5 | Board 4 and 5 active |
| 5 | CLK6 | Board 6 active |
| 15 | BKPLN_SLCT | Set low for 6-slot backplane mapping |

The appropriate mapping for 8-Slot or 6-Slot backplanes is set by bit 15 of the register: Setting this bit high enables mapping for 8-Slot backplanes. When this bit is set low, the mapping is enabled for 6-Slot backplanes.

### 1.2.3 Sequencer Bus Interface

The Master Control Board controls all activity on the Sequencer Bus. This bus is unidirectional, away from the MCB, and controls all peripheral DHE boards on the backplane. Bus activity is synchronous to the rising edge of the system clock.

There are four groups of signals that make up the Sequencer Bus: eight bits of board select (`/SEL2` TO `/SEL8`), two bits of mode (`SEQ_MODE[1:0]`), six bits of device address (`DEV_ADDR[5:0]`), and 32 bits of data (`SEQ_DATA[31:0]`).

The `SEQ_MODE[1:0]` signals allow three types of bus transactions, reset, write and read. These are detailed in Table 3.

Table 3: Sequencer Bus Mode Bit Definitions

| Mode Bits | Transaction | Mode | Description |
|---|---|---|---|
| 00 | Reset | Hard or Soft | Reboot FPGA or soft reset the addressed board depending on SEQ_DEVADDR[2:0] bits. |
| 01 | Read | 32 Bits | Read a word from the addressed board |
| 10 | Write | 16 Bits | Write a 16-bit word to the addressed board |
| 11 | Write | 32 Bits | Write a 32-bit word to the addressed board |

The board select bits (`/SEL2` TO `/SEL8`) are used to activate a board for a bus transaction. It is legal for multiple board select lines to be active for a reset or write transaction but only a single board select can be active for a read transaction.

A reset or write transaction takes one clock cycle and a read transaction takes three clock cycles to complete.

#### 1.2.3.1 Reset Transactions

Reset mode is similar in timing to a write transaction.

Setting all bits of the `SEQ_DEVADDR[5:0]` signals high results in a reboot of the FPGA on the selected board and all configuration data is set to default values.

Setting the `SEQ_DEVADDR[5:0]` signals low during a reset transaction performs a 'soft reset' of the board, only the functionality is reset (i.e. state machines, etc.) and current

configuration data is preserved.

**NOTE:** At this time Sequencer bus data content has no significance in the reset transaction.

#### 1.2.3.2 Write Transactions

Only the lowest sixty four memory locations (`0x0000` to `0x003F`) of a peripheral board can be written to in 32-bit mode. In this mode the `SEQ_DEVADDR[5:0]` signals carry the address information to the peripheral board. Data is carried on signals `SEQ_DATA[31:0]`. The data is written into the peripheral board on the rising edge of the board `SYSCLK` when `/SELn` is set true.

When writing 16-bit data, the address on the peripheral board is carried in the most significant 16 bits of the sequencer data bus (`SEQ_DATA[31:16]`) signals and data contained in the least significant bits (`SEQ_DATA[15:0]`).

**ToDo:** Add diagram showing 16bit and 32 bit writes.

#### 1.2.3.3 Read Transactions

During a read transaction, a 'write' is performed to the peripheral board with the required board address contained in the least significant 16 bits of the sequencer bus signals (`SEQ_DATA[15:0]`). Two clock cycles are allowed for the peripheral board to decode the address and put the requested data onto the Pixel Data Bus; Then the `topPixFpga` module input control line `seqWrite_n` is taken true by the PAN command decode logic and data from the pixel data bus is latched into the Pixel Bus FIFO.

An additional set of signals, Peripheral Board Acknowledge Strobes (`bp_ack(8:2)`), is defined for this bus. Currently, these signals are not used.

### 1.2.4 Pixel Data Bus Interface

The Pixel Data Bus (`PIX_DATA[47:0]`) transfers data from peripheral boards to the MCB and from there to the PAN by way of the Slink link. It is a unidirectional 48-bit bus that connects to a 128-word FIFO (Pixel Bus FIFO) within the pixel FPGA. Data loaded into this FIFO is transmitted directly to the PAN by way of the Slink Interface. Data present on the Pixel Data Bus is loaded into this FIFO by two methods:

- If the data is in response to a decoded read command from the PAN, the signal `seqWrite_n` is used to strobe data into the FIFO with timing controlled by the Pan Decode logic.

- If the data is pixel data generated by a peripheral board then data is loaded into the FIFO on the rising edge of the system clock when any of the seven `/pipe_req_n[6:0]` signals and signal `/pipe_wrt_n` are active.

This facility allows a peripheral board (Video Acquisition Board) to burst transfer pixel data to the PAN. There is a priority scheme associated with the use of this feature that is handled by the peripheral boards themselves.

### 1.2.5 Synchronization Interface

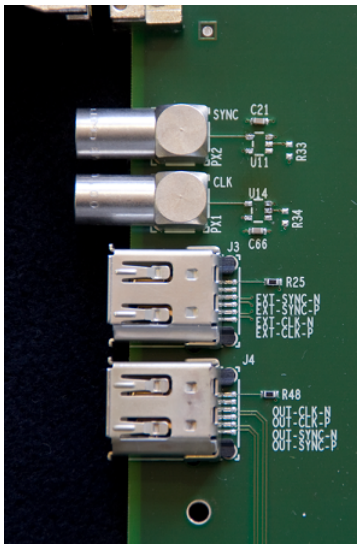


Figure 1: Synchronization Connectors

To allow multiple DHE chassis to act in unison, two connectors are provided to enable tight, clock cycle-level synchronization. J3 accept externally generated synchronization signals and J4 drive synchronization signals out to other DHE chassis in the daisy chain.

There are two synchronization signals, `EXTCLK` and `EXTSYNC`, that are driven and received as PECL logic levels. In a multiple DHE configuration, one DHE acts as master and generates the `EXTCLK` from the system clock source.

The driven signals from a DHE can be adjusted in phase writing to MCB registers to provide phase match of DHE chassis. Both sync and clock signals can be adjusted. Each one has its own register:

- To adjust clock phase, register `ADDR_CLKCTRLDEC` must be used.
- To adjust sync phase, register `ADDR_SYNC_DELAY` must be used.

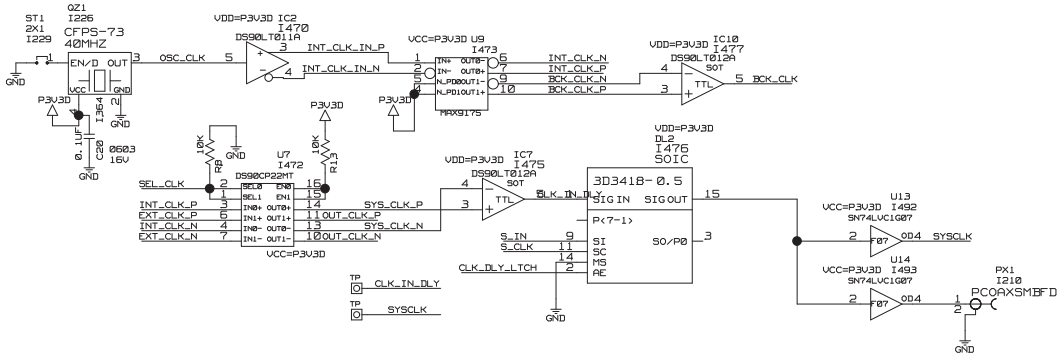


Figure 2: Clock Path

Setting an MCB as master or slave is controlled by setting bit 4 inthe MCB control register (`ADDR_SEQCTLREG`)[1]. This provides the basic mechanism to begin control processes run by the MPU Sequencer in synchronization with each other.

[1]In old releases of the MCB (both NOAO or Barcelona) this was selected using jumpers on the board.

This bit automatically selects which signals arrives to the fpga. When MCB is set as master, internal oscillator clock feeds fpga sysclk. Else, when set as slave, sysclk is feed by external clock. We've added a backup clock which goes to another clock buffer of the fpga[2]. From figure 2 we can see that the clock that can be seen on the front panel lem0 connector is the same that goes to the fpga. This is to be able to adjust clock phase between crates with an oscilloscope.

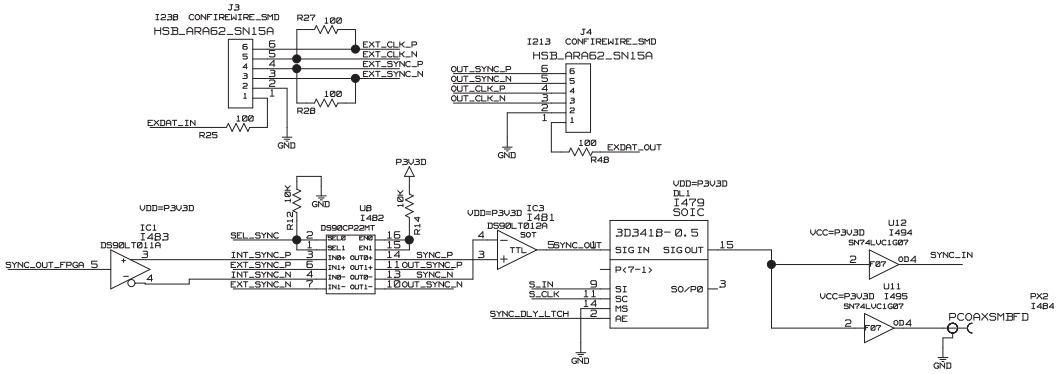


Figure 3: Sync Path

The sinc signal path is similar to the clock path, but when the MCB is set as Master, the signal it is generated inside of the fpga. Front panel lem0 connector is to see the signal which goes to the fpga after the delay chip.[3]

In addition, the SEQUENCER MPU code should be able to use the sync signal that appears in the sequencer command register (`ADDR_SEQCMDREG`) and set the master flag in the MCB control register.

A slow and simple daisy-chained serial data interface is available for inter-DHE communication. There is one serial data in and one serial data out port assigned to the respective synchronization connectors. The protocol, and therefore internal logic, to control these has not yet been defined. However, there is a simple one bit interface available through the `ADDR_EXTDATALATCH` register which can be used to set the output bit and read the input bit.

[2]In a near future firmware release is planned to implement a backup system which will track if system is receiving clock signal while set as slave. To be able to recover the system in case it is set as slave but it doesn't get clock through ext_clk. This will be configurable (it could be disabled, set how many seconds it must wait without clock signal before changing from extern clock to intern oscillator clock,...)

[3]In future firmware release we plan to add a delay setting mode which will output an slow clock through the sync path, just to be able to adjust the delay.

### 1.2.6 JTAG Interface, Reset and Power on Boot.

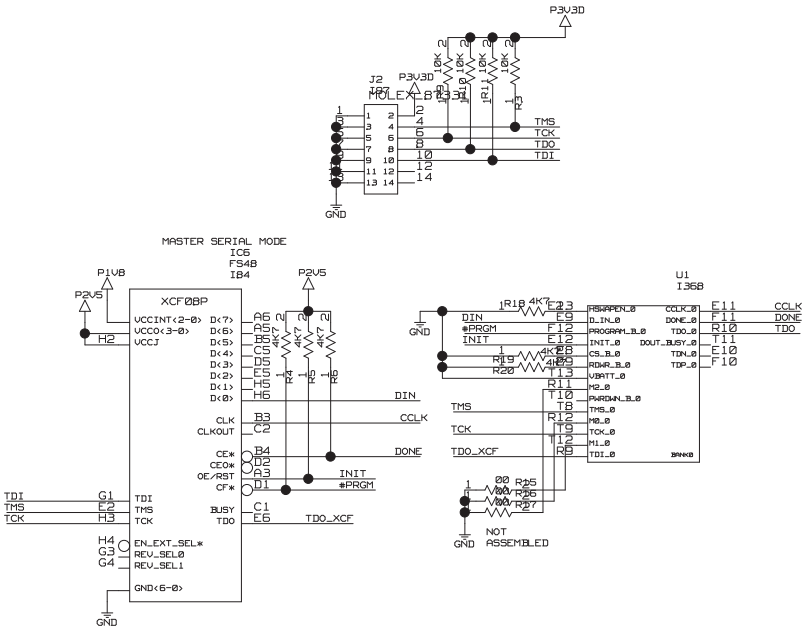


Figure 4: Jtag interface

The programmable logic device that contain all the logic for the MCB is accessible for programming and limited diagnostics by the JTAG connector J7.

Firmware reloading and initialization cycle can be initiated manually by pressing the front panel reset button (U5) and by PAN command by writing to the (`ADDR_RELOADFW` or `ADDR_REPROGRAM`) register. Figure 5 shows hardware implementation of the reload firmware.

When a reprogram command is received by the MCB `#PROGRAM` signal is set after 100ms. `#PROGRAM` is captured by a latch and it goes to a monostable which will maintain this signal active enough time to reload the firmware.

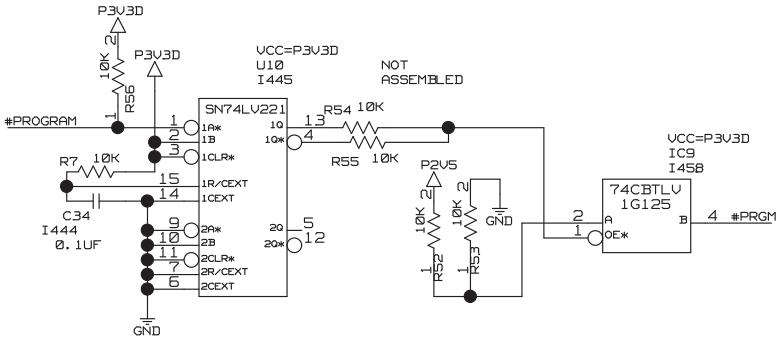


Figure 5: Firmware reload from fpga

### 1.2.7 Front Panel Indicators

There are two front panel single LED indicators used for basic diagnostics of the board. D4 is controlled by firmware in the Pixel FPGA and D3 by firmware in the Sequencer FPGA.

During power up, each indicator is off until the firmware for the respective FPGA has been booted and started. After booting their respective code, the indicators will be steadily on until an asynchronous command has been sent from the PAN node to synchronize the communication link. At this time both indicators will go out until programmed to indicate some

functionality through their respective control registers (ADDR_LEDLATCH, ADDR_SEQ_LED). Table 4 lists the functionality that is provided.

Table 4: Front Panel Indicator Functions

| ADDR_LEDLATCH | | ADDR_SEQ_LED | |
|---|---|---|---|
| BIT | Function | BIT | Function |
| 0 | FPDP TM Write Request Active | 0 | FPDP /TA_DVALID Flag True |
| 1 | Funct. Cmd. Decoder Active | 1 | PAN Seq. Bus Access Request |
| 2 | Pixel Data Request Active | 2 | MPU Seq. Bus Access Request |

**NOTE:** right now, 3 led tower leds are assigned to some debug signals (If red led is ON don't worry). Next firmware release will come with more functionality (disable leds, show disabled slink leds,..)

## 1.3 Register Map / Control Functions

All control functions that the MCB is capable of are initiated by writing to specific memory locations on the MCB. The board address for the MCB is always 1. For details on the format of the data written to the DHE from the PAN, please refer to document MNSN-AD-01-0004 R 1.0 ICD 6.0 Generic DHE Command and Data Stream Interface.

A summary of the memory space for the MCB is dispalyed in Table 5 followed by a more detailed description of each function.

Table 5: MCB Memory Map

| Address | Range | FW Name | Description |
|---|---|---|---|
| 0x0000 | | ADDR_SEQINTEGTIME | Requested Integration time |
| 0x0001 | | ADDR_COUNTREG | Test Data Generator |
| 0x0100 | | ADDR_CLOCKEN | Clock Enable Register |
| 0x0101 | | ADDR_SEQACTINTTIM | Actual Integration Time |
| 0x0102 | | ADDR_SEQCMDREG | Sequencer MPU Command Reg |
| 0x0110 | 0x011f | ADDR_RANGE_SEQLOOP | Sequencer MPU loop Registers |
| 0x0200 | 0x03ff | ADDR_RANGE_PIXCMD | Pixel command Range |
| 0x0200 | 0x027f | ADDR_RANGE_SCONF | Serial config Port (Write to 0x027f triggers write to eeprom) |
| 0x0280 | | ADDR_SERFETCH | Serial config fetch (Write triggers a read cycle of eeprom) |
| 0x0281 | | ADDR_CLKCTRLDEC | Clock Phase Port |
| 0x0282 | | ADDR_LEDLATCH | LED Port enable register |
| 0x0286 | | ADDR_EXTDATALATCH | External data port |
| 0x02d0 | | ADDR_PIXROWREG | Simulation |
| 0x02d1 | | ADDR_PIXCOLREG | Simulation |
| 0x02d2 | | ADDR_PIXTIMEREG | Simulation |
| 0x02d3 | | ADDR_PIXCHANREG | Simulation |
| 0x02d4 | | ADDR_PIXDLYREG | Simulation |
| 0x02fc | | ADDR_PIXCTLREG | Pix FPGA Control register |

| Address | Range | FW Name | Description |
|---|---|---|---|
| 0x02ff | | ADDR_FWVERHEX | Firmware version (makes sense in hex) |
| 0x0300 | 0x031f | ADDR_RANGE_SLINK | Address range for Slink Statistics registers |
| 0x0300 | | ADDR_SLINK_SRDW_L | Received data words (16 LSB) |
| 0x0301 | | ADDR_SLINK_SRDW_H | Received data words (16 MSB) |
| 0x0302 | | ADDR_SLINK_SRCW_L | Received Control Words (16 LSB) |
| 0x0303 | | ADDR_SLINK_SRCW_H | Received Control Words (16 MSB) |
| 0x0304 | | ADDR_SLINK_SXDW_L | Sent Data Words (16 LSB) |
| 0x0305 | | ADDR_SLINK_SXDW_H | Sent Data Words (16 MSB) |
| 0x0306 | | ADDR_SLINK_SXCW_L | Sent Control Words (16 LSB) |
| 0x0307 | | ADDR_SLINK_SXCW_H | Sent Control Words (16 MSB) |
| 0x0308 | | ADDR_SLINK_SXDP_L | Sent Data Pages (16 LSB) |
| 0x0309 | | ADDR_SLINK_SXDP_H | Sent Data Pages (16 MSB) |
| 0x030a | | ADDR_SLINK_SXCP_L | Sent Control Pages (16 LSB) |
| 0x030b | | ADDR_SLINK_SXCP_H | Sent Control Pages (16 MSB) |
| 0x030c | | ADDR_SLINK_SXDCP_L | Sent Data Complete Pages (16 LSB) |
| 0x030d | | ADDR_SLINK_SXDCP_H | Sent Data Complete Pages (16 MSB) |
| 0x030e | | ADDR_SLINK_SXCCP_L | Sent Control Complete Pages (16 LSB) |
| 0x030f | | ADDR_SLINK_SXCCP_H | Sent Control Complete Pages (16 MSB) |
| 0x0310 | | ADDR_SLINK_SLFFE | Full Flag errors |
| 0x0311 | | ADDR_SLDE | Link Down errors |
| 0x0312 | | ADDR_SLINK_SUTM | Minutes since power up |
| 0x0313 | | ADDR_SLINK_SUTS | Seconds since change in ADDR_SLINK_SUTM |
| 0x0314 | | ADDR_SLINK_IPL | Configuration of Slink, Page length in Words |
| 0x0315 | | ADDR_SLINK_IMW | Configuration of Slink, Max Wait in clock cycles before closing an incomplete page |
| 0x0600 | | ADDR_PCB_VERSION | PCB revision |
| 0x0601 | | ADDR_SYNC_DELAY | Sync Delay register |
| 0x0666 | | ADDR_REPROGRAM | When write MCB FPGA is reprogramed |
| 0x1000 | 0x1bff | ADDR_RANGE_SEQPATMEM | Sequencer pattern memory |
| 0x4000 | 0x43ff | ADDR_RANGE_SEQPRGMEM | Sequencer program memory |
| 0xfff8 | | ADDR_SWITCHES | Register to configure clock and sync switches |
| 0xfff9 | | ADDR_SEQ_LED | Led indicator select |
| 0xfffa | | ADDR_BOARDSERIAL | Circuit Board Serial number |
| 0xfffb | | ADDR_BOARDTEMP | Temperature register |
| 0xfffc | | ADDR_SEQCTLREG | Control Register |
| 0xfffd | | ADDR_SEQSTATUS | Status Register |
| 0xfffe | | ADDR_BOARDIDREG | Board ID Register (Read) |
| | | ADDR_SOFTRESET | Issue a soft reset to fpga firmware (Write) |

| Address | Range | FW Name | Description |
|---|---|---|---|
| 0xffff | | ADDR_FWVERDEC | Firmware version (makes sense in decimal) (Read) |
| | | ADDR_RELOADFW | Reload fpga firmware from eprom (Write) |

### 1.3.1 FW constants

Next, a listing of the current constantsPkg.vhd file can be found. This is a live include of the working copy of the file, nothing is more up to date than this listing[4].

```
library ieee;
use ieee.std_logic_1164.all;

package constantsPkg is

  -- Constants
  constant PCB_VERSION : std_logic_vector(15 downto 0) := x"0102";

  -- MASTER CONTROL BOARD PART NUMBER (0400)
  constant BOARDID : std_logic_vector(15 downto 0) := x"0190";
  -- VERSION 5.11
  constant FWVERDEC  : std_logic_vector(15 downto 0) := x"0201";
  constant FWVERHEX  : std_logic_vector(15 downto 0) := x"0513";



  ------------------------------------------------------------------------------
  -- adresses (pixel registers: 0x0200->0x03FF)
  ------------------------------------------------------------------------------
  -- Slink Stats Range (0x0300 -> 0x31f) (Some spares)
  constant ADDR_RANGE_SLINK   : std_logic_vector(15 downto 5) := x"03" & "000";
  -- statsRcvdDatWrd
  constant ADDR_SLINK_SRDW_L  : std_logic_vector(15 downto 0) := x"0300";
  constant ADDR_SLINK_SRDW_H  : std_logic_vector(15 downto 0) := x"0301";
  -- statsRcvdCtlWrd
  constant ADDR_SLINK_SRCW_L  : std_logic_vector(15 downto 0) := x"0302";
  constant ADDR_SLINK_SRCW_H  : std_logic_vector(15 downto 0) := x"0303";
  -- statsXferDatWrd
  constant ADDR_SLINK_SXDW_L  : std_logic_vector(15 downto 0) := x"0304";
  constant ADDR_SLINK_SXDW_H  : std_logic_vector(15 downto 0) := x"0305";
  -- statsXferCtlWrd
  constant ADDR_SLINK_SXCW_L  : std_logic_vector(15 downto 0) := x"0306";
  constant ADDR_SLINK_SXCW_H  : std_logic_vector(15 downto 0) := x"0307";
  -- statsXferDatPag
  constant ADDR_SLINK_SXDP_L  : std_logic_vector(15 downto 0) := x"0308";
  constant ADDR_SLINK_SXDP_H  : std_logic_vector(15 downto 0) := x"0309";
  -- statsXferCtlPag
  constant ADDR_SLINK_SXCP_L  : std_logic_vector(15 downto 0) := x"030a";
  constant ADDR_SLINK_SXCP_H  : std_logic_vector(15 downto 0) := x"030b";
  -- statsXferDatCPag
  constant ADDR_SLINK_SXDCP_L : std_logic_vector(15 downto 0) := x"030c";
  constant ADDR_SLINK_SXDCP_H : std_logic_vector(15 downto 0) := x"030d";
  -- statsXferCtlCPag
  constant ADDR_SLINK_SXCCP_L : std_logic_vector(15 downto 0) := x"030e";
  constant ADDR_SLINK_SXCCP_H : std_logic_vector(15 downto 0) := x"030f";
  -- statsLFFErr
  constant ADDR_SLINK_SLFFE   : std_logic_vector(15 downto 0) := x"0310";
  -- statsLDownErr
  constant ADDR_SLINK_SLDE    : std_logic_vector(15 downto 0) := x"0311";
  -- statsUpTimeMin
  constant ADDR_SLINK_SUTM    : std_logic_vector(15 downto 0) := x"0312";
  -- statsUpTimeSec
  constant ADDR_SLINK_SUTS    : std_logic_vector(15 downto 0) := x"0313";
  -- infoPageLength
  constant ADDR_SLINK_IPL     : std_logic_vector(15 downto 0) := x"0314";
  -- infoMaxWait
  constant ADDR_SLINK_IMW     : std_logic_vector(15 downto 0) := x"0315";

  -- SerConfigIFC (0x0200->0x027f)
  -- I've deleted serconfigIfc. when I implement it again
  -- but interfacing 1-wire protocol chip I will use it again
  constant ADDR_RANGE_SCONF : std_logic_vector(15 downto 7) := x"02" & '0';
  -- serial eeprom control register
  constant ADDR_SERFETCH    : std_logic_vector(15 downto 0) := x"0280";

  -- ExtDataLatch
  constant ADDR_EXTDATALATCH : std_logic_vector(15 downto 0) := x"0286";

  -- LedLatch
  constant ADDR_LEDLATCH : std_logic_vector(15 downto 0) := x"0282";

  -- pixCmdDec
  constant ADDR_PIXROWREG  : std_logic_vector(15 downto 0) := x"02d0";
  constant ADDR_PIXCOLREG  : std_logic_vector(15 downto 0) := x"02d1";
```

[4] In case of doubt, if the register you are looking for has not the same address in the table above and in this listing, trust the listing (and please inform me so I can correct it).

```
  constant ADDR_PIXTIMEREG : std_logic_vector(15 downto 0) := x"02d2";
  constant ADDR_PIXCHANREG : std_logic_vector(15 downto 0) := x"02d3";
  constant ADDR_PIXDLYREG  : std_logic_vector(15 downto 0) := x"02d4";
  constant ADDR_PIXCTLREG  : std_logic_vector(15 downto 0) := x"02fc";
  constant ADDR_FWVERHEX   : std_logic_vector(15 downto 0) := x"02ff";

  -- clk phase adjustment / clock delay chip
--  constant ADDR_CLKCTRLDEC : std_logic_vector(15 downto 0) := x"0281";
  constant ADDR_CLKPHASE : std_logic_vector(15 downto 0) := x"0281";

-------------------------------------------------------------------------
-- CONSTANTS FOR RESET MODES
-------------------------------------------------------------------------
  constant ADDR_RELOADFW : std_logic_vector(15 downto 0) := x"FFFF";  -- BOARD CODE REVISION IDENTITY REGISTER
  constant ADDR_SOFTRESET : std_logic_vector(15 downto 0) := x"FFFE";  -- BOARD FUNCTIONAL IDENTITY REGISTER

  ---------------------------------------------------------------------------
  -- adresses (sequencer registers)
  ---------------------------------------------------------------------------

  constant ADDR_PCB_VERSION  : std_logic_vector(15 downto 0) := x"0600";
  constant ADDR_SYNC_DELAY   : std_logic_vector(15 downto 0) := x"0601";
  constant ADDR_REPROGRAM    : std_logic_vector(15 downto 0) := x"0666";
--  constant ADDR_EVENTREG     : std_logic_vector(15 downto 0) := x"fff0";
  constant ADDR_SWITCHES     : std_logic_vector(15 downto 0) := x"fff8";
  constant ADDR_SEQ_LED      : std_logic_vector(15 downto 0) := x"fff9";
  constant ADDR_BOARDSERIAL  : std_logic_vector(15 downto 0) := x"fffa";
  constant ADDR_BOARDTEMP    : std_logic_vector(15 downto 0) := x"fffb";
  constant ADDR_SEQCTLREG    : std_logic_vector(15 downto 0) := x"fffc";
  constant ADDR_SEQSTATUS    : std_logic_vector(15 downto 0) := x"fffd";
  constant ADDR_BOARDIDREG   : std_logic_vector(15 downto 0) := x"fffe";
  constant ADDR_FWVERDEC     : std_logic_vector(15 downto 0) := x"ffff";
  constant ADDR_SEQINTEGTIME : std_logic_vector(15 downto 0) := x"0000";
  constant ADDR_COUNTREG     : std_logic_vector(15 downto 0) := x"0001";
  constant ADDR_CLOCKEN      : std_logic_vector(15 downto 0) := x"0100";
  constant ADDR_SEQACTINTTIM : std_logic_vector(15 downto 0) := x"0101";
  constant ADDR_SEQCMDREG    : std_logic_vector(15 downto 0) := x"0102";
  constant ADDR_STARTEXP     : std_logic_vector(15 downto 0) := x"0602";

  ---------------------------------------------------------------------------
  -- Sequencer adresses ranges
  ---------------------------------------------------------------------------
  -- Sequencer loop range 0x0110 -> 0x011f
  constant ADDR_RANGE_SEQLOOP   : std_logic_vector(15 downto 4)  := x"011";
  -- Sequencer pattern memory range (0x1000->0x1bff) can not be described this
  -- way. It is hardcoded on Personality module.
  constant ADDR_RANGE_SEQPATMEM : std_logic_vector(15 downto 12) := x"1";
  -- sequencer programm memory 0x4000 -> 0x43ff
  constant ADDR_RANGE_SEQPRGMEM : std_logic_vector(15 downto 10) := x"4"&"00";
  -- pixel command range 0x0200 -> 0x03ff
  constant ADDR_RANGE_PIXCMD    : std_logic_vector(15 downto 9)  := x"0" & "001";


end constantsPkg;
```

### 1.3.2 Former Sequencer fpga registers description

**1.3.2.1 Integration Time Register (`ADDR_SEQINTEGTIME`)** This is a 32-bit register that holds the absolute time required for an integration. The value is used by the Integration Time Counter to determine the end of integration for an exposure. If necessary, the register can be written to with a new value during the integration phase to reflect a change of integration time. The value written to this register is in units milliseconds. The minimum value is zero and the maximum 4294967295 (i.e. 1193 hours).

**1.3.2.2 Test Counter Register (`ADDR_COUNTREG`)** A 32-bit value written to this register arms the test counter function. When a `START EXPOSURE` command is received from the PAN, the written value of pixels will be sent to the PAN at intervals of 1µs. The data for each pixel will be the current 32-bit counter value, which is decremented after each pixel is sent. When the test counter reaches zero, it will become inactive and another value must be written to the register to repeat the test. Its primary purpose is to provide data path verification between the MCB to PAN.

It is necessary to prevent the MPU SEQUENCER from responding to the `START EXPOSURE` command and sending its own data to the PAN by disabling the MPU SEQUENCER. Disabling the MPU SEQUENCER can be done by writing a zero to the MCB Control Register (`ADDR_SEQCTLREG`, see 1.3.2.12 on page 21) bit 0 (`SEQ_ENABLE`).

**1.3.2.3 Clock Enable Register (`ADDR_CLOCKEN`)** A 16-bit value written to this register enables the system clock to each peripheral board slot on the backplane that has a bit set in the value. The mapping for this function is detailed in Table 1.

Although the backplane and peripheral boards are designed to be synchronous, that is, they require a clock to perform any transaction or function, peripheral boards have a limited ability to access their Board Identity Register and Firmware Code revision register without activating the appropriate clock to that backplane slot. This enables a PAN to determine which boards and what functionalities are present in the DHE without activating a clock to each peripheral backplane slot in turn.

**1.3.2.4 Integration Timer Counter (`ADDR_SEQACTINTTIM`)** The integration timer counter is a 32-bit read only register. The counter can be enabled and disabled from the MPU Sequencer Enables register using microcode. When enabled, the value will increment each millisecond until the value stored in the Integration Time Register is reached or surpassed. At this time a flag (`SEQ_CMD_ITC`) is set in the MPU Sequencer Command register (see 1.3.2.5).

The counter will continue to increment until it is disabled by an MPU Sequencer microcode instruction. In this way, and assuming the microcode is designed for this, the PAN may read the actual integration time from this register after integration is complete. This counter can be 'paused' by the PAN by using the appropriate bit (bit 1) in the `ADDR_SEQCTLREG` register (`SEQ_INTPAUSE`) (see 1.3.2.12 on page 21).

**1.3.2.5 Sequencer Command Register (`ADDR_SEQCMDREG`)** This 16-bit register provides the mechanism for conditional branching within the MPU SEQUENCER microcode.

The MPU SEQUENCER instruction set provides one conditional jump command (JCB) that requires a destination for the jump and a bit number to test. The instruction branches if the specified bit in the Sequencer Command register is set. Therefore there are 16 possible branches that can be specified.

The lowest 4 bits (3:0) of the command register are set by hardware. The middle 8 bits (11:4) are set from the `START EXPOSURE` command as the required start vector within the sequencer program code. The last 4 bits (15:12) are written directly from the data value written to this register from the PAN. Figure 6 illustrates the relative bit positions for the branch conditions.


Figure 6: MPU SEQUENCER Command Register Structure

- Bit 0 is tied to '1' to provide an unconditional branch instruction (JMP).
- Bit 1 reflects the state of the external SYNC signal used for multiple DHE synchronization.
- Bit 2 is set true whenever the value of the Integration Time Counter is greater than or equal to the Integration Time Register.
- Bit 3 is set whenever a `START EXPOSURE` command is received from the PAN. This indicates that the condition of bits (11:4) has been established. This bit can be reset by a microcode instruction to the `SEQ_ENBLREG` (see 1.3.5.1 on page 26).
- Bit 11 to 4 is set by the `START EXPOSURE` command vector and is used to evaluate what type of sequence should be carried out by the MPU Sequencer microcode.

- Bit 15 to 12 are directly written from the least significant four bits of data during a PAN write command to this register. These bits may be tested and used to control modes of functionality within the MPU Sequencer microcode.

**1.3.2.6 Sequencer Loop Counters (`ADDR_RANGE_SEQLOOP`)** To provide for looping structures within the MPU Sequencer microcode, sixteen 16-bit registers are available. Each register can be tested and branching controlled by the LRB (Loop Register Begin) and LPE (Loop End) instructions. These registers are static, that is, their value remains unchanged by the effect of microcode instruction execution.

**1.3.2.7 Sequencer Pattern Memory Space (`ADDR_RANGE_SEQPATMEM`)** The MPU SEQUENCER functionality is controlled by the microcode stored in the Sequencer Program Code Memory space, however, the data that is actually put onto the Sequencer Bus during MPU SEQUENCER control comes from the Sequencer Pattern Memory space. This memory space appears to the PAN as a 4K x 16-bit word block but is used by the sequencer as 2K x 32-bit words. This pattern memory contains data that is to be written to peripheral boards at certain times under control of the microcode. Any memory location on any peripheral board can be written to with data from the pattern memory. Memory locations on the MCB cannot be written to under microcode control with the exception of the SEQ_ENBLREG which can only be written to under microcode control. It alsodoes not appear in MCB Memory space visible from the PAN.

**1.3.2.8 Sequencer Program Code Memory Space (`ADDR_RANGE_SEQPRGMEM`)** The microcode used to control DHE functionality is stored within this 1K memory block. This memory is written and read as 16 bits wide, however, the microcode is based on a four-bit word slice so there is effectively 4K word memory for the MPU SEQUENCER microcode. This memory space is blocked to the PAN when the MPU sequencer is running (bit 0 of the MCB Control Register is 1).

**1.3.2.9 MCB Sequencer LED Indicator Control Register (`ADDR_SEQ_LED`)** There are three bits which control the function of the Sequencer FPGA indicator LED (D3). After power on or a reboot, the indicator will be on permanently until communication with the PAN has been established. After this, writing to one or more of the control bits will provide a 10ms indicator flash each time the specific function goes true. The functions that can be assigned for this are listed in Table 6.

Table 6: MCB Sequencer Led function codes

| BIT | Function |
|---|---|
| 0 | Flash each time a data word is received from the PAN |
| 1 | Flash each time the PAN command decode logic requests use of the sequencer bus |
| 2 | Flash each time the MPU Sequencer logic requests use of the sequencer bus |

**1.3.2.10 Silicon Serial Number Register (`ADDR_BOARDSERIAL`)** Reading this register returns a 32-bit word that is a unique serial number read from IC8, a ds2433[5] chip. This chip

[5] `http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2915`

consists of a factory-lasered registration number that includes a unique 48-bit serial number, an 8-bit CRC, and an 8-bit Family Code (23h) plus 4096 bits of user-programmable EEPROM.

With this register the 32 less significant bits can be read from PAN.

**1.3.2.11 Board Temperature Register (`ADDR_BOARDTEMP`)** Reading this register returns a 10-bit signed number that indicates the current ambient temperature of the board. Each LSB corresponds to 0.25ºC. The temperature is updated continuously and nothing has to be done to read it. The sensor used is an AD7814[6].

**1.3.2.12 MCB Control Register (`ADDR_SEQCTLREG`)** The MCB Control register provides overall control of the MPU SEQUENCER. The bit assignments for this register are as follows:

- Bit 0 `SEQ_ENABLE`: Setting this bit to '1' allows the sequencer to begin decoding and executing microcode instructions. Whenever the signal is de-asserted, the MPU enters into the reset state after executing the current instruction. All the board selects and the Sequencer Bus request signals are masked into non-active state. The program counter is reset to zero.
- Bit 1 `SEQ_INTPAUSE`: Setting this bit to '1' will stop the Integration Time Counter (ITC) from incrementing. Resetting this bit allows the ITC to continue counting up.
- Bit 3 to 2 `SEQ_CLKDIV`: This 2-bit pattern defines the rate at which the MPU SEQUENCER instructions are executed. Note that current firmware runs the MPU SEQUENCER at half the system clock frequency with `SEQ_CLKDIV` set to zero . Each microcode instruction takes exactly one sequencer clock cycle to complete.

Table 7: MPU Sequencer Instruction Cycle Time

| Bits | | |
|---|---|---|
| 3 | 2 | Instruction cycle time |
| 0 | 0 | 50ns (20 MHz) |
| 0 | 1 | 100ns (10 MHz) |
| 1 | 0 | 200ns (5 MHz) |
| 1 | 1 | 400ns (2.5 MHz) |

- Bit 4 DHE_MASTER: When set, this bit establishes the DHE as a master, enables the output of the external SYNC_OUT signal on J4, and sets the clock and sync selectors (U8 and U7) to use internal or external signals. In the default state of zero (1), the DHE is considered a master DHE, the SYNC bit in the CMD register reflects the state of the SYNC RS Flip Flop controlled through the EFR. When configured as a slave DHE the external SYNC_IN signal on J3 is routed to the CMD register SYNC bit.
- Bit 5 to7: Currently not used.
- Bit 8 to 15 SEQ_SYNCDELAY: This 8-bit value sets the delay (units of milliseconds) between the time the MPU SEQUENCER resets the SYNC bit in the EFR until the hardware signal is emitted by the master DHE. This delay allows sufficient time for

[6] `http://www.analog.com/en/sensors/digital-temperature-sensors/ad7814/products/product.html`

slave DHEs to be configured with a start exposure command by their relative PANS before looking for the SYNC event.

**1.3.2.13 MCB Status Register (ADDR_SEQSTATUS)** This read-only 16-bit register contains status information from various functions on the MCB. It is currently not used to any advantage, although the least significant bits of the MPU Sequencer program counter can be seen to provide limited feedback on what the microcode is doing.

**1.3.2.14 MCB Board Identity Code (ADDR_BOARDIDREG) (ADDR_SOFTRESET)** Reading from this address will return data that will identify this board as a Master Control Board (Code = 0x0190)

Writing to this address will result in the MCB going through a soft reset cycle. This will stop the MPU SEQUENCER, load zeros into the Clock Enable and Test Counter registers, and release the Sequencer Bus. Further information on board identity codes and firmware revision values can be found in Appendix I.

**1.3.2.15 MCB Firmware Code Revision (ADDR_FWVERDEC)(ADDR_RELOADFW)** Reading from this address will return data that will identify the version of firmware running on the MCB. The format of the data is binary that should be divided by 100d to find the correct revision number (XX.nn).

Writing to this address will result in the MCB going through a hard reset cycle. This will set signal BOOT_N true and that will result in a reloading of the firmware .

**1.3.2.16 System Clock Phase Control Register (ADDR_CLKCTRLDEC)** This register was formerly implemented on the pixel fpga firmware. Due to changes in Hardware in new MCBs, now this register is implemented inside sequencer firmware part.

Writing 8 bits of significant data to this register controls the clock phasing for the synchronization port clock recovery and clock distribution. This functionality is used when two or more DHE chassis need to be synchronized. Normally, each DHE sources its own clock to run the logic, however, when multiple DHE chassis are used to control one focal plane it is important to synchronize their operation to avoid impulse noise sources from each system interfering with the others.

When using multiple DHE chassis in synchronized operation, one DHE is nominated as a master and feeds the system clock to the slave DHEs in a daisy chain configuration of controlled impedance cables. Each slave has two clock skew adjustment regimes that can be used together to arrive at a sub-nanosecond phase alignment between all DHE chassis.

The DHE system clock is derived from a nominal 40MHz clock source and passed through a 3d3418-0.5[7], a digitally programmable delay line to drive the logic.

The clock source for a master DHE is the crystal oscillator. The clock source for a slave DHE is the LVDS logic receiver which is driven from the EXT_CLK_N and EXT_CLK_P signals at J3. Clock selection is achieved with chip U7 ( ds90cp22mt[8], see figure 2 on page 12).

This clock source selector drives the signal to the delay line and to the LVDS driver to sent out the system clock to the next DHE system using connector J4.

The 3D3418 Programmable 8-Bit Silicon Delay Line product family consists of 8-bit, user-programmable CMOS silicon integrated circuits. Delay values, programmed via the serial interface, can be varied over 255 equal steps of 500ps. Units have a typical inherent (address 0) delay of 19.5ns. The input is reproduced at the output without inversion, shifted in time as per user selection. The 3D3418 is CMOS-compatible, and features both rising- and falling-edge accuracy.

[7] http://www.datadelay.com/datasheets/3d3418.pdf
[8] http://www.national.com/pf/DS/DS90CP22.html

A programmable delay controlled by register `ADDR_CLKCTLDEC` can be applied to control system clock phase to this local DHE chassis. Delay applied can be calculated as: $t_d = RegValue * 0.5ns + 19.5ns$

Connector PX1 is a copy of the sysclk signal received by the fpga. This output is easily plugged into an osciloscope to adjust all DHE systems system clock phases.

**1.3.2.17 Sync Phase Control Register (`ADDR_SYNC_DELAY`)** Writing 8 bits of significant data to this register controls the sync phasing for the synchronization port. This functionality is used when two or more DHE chassis need to be synchronized.

Normally, each DHE sources its own sync to run the logic, however, when multiple DHE chassis are used to control one focal plane it is important to synchronize their operation to avoid impulse noise sources from each system interfering with the others.

When using multiple DHE chassis in synchronized operation, one DHE is nominated as a master and feeds the sync to the slave DHEs in a daisy chain configuration of controlled impedance cables. Each slave has two clock skew adjustment regimes that can be used together to arrive at a sub-nanosecond phase alignment between all DHE chassis.

The DHE sync signal is generated by logic in the fpga and passed through a 3d3418-0.5[9], a digitally programmable delay line to drive the logic.

Connector PX2 is a copy of the sync signal received by the fpga. This output is easily plugged into an osciloscope to adjust all DHE systems system sync phases.

**1.3.2.18 Synchronization ports selector controller (`ADDR_SWITCHES`)** This register is used to configure behavior of synchronization port selector.

Default value at power up sets sync and sysclk selector based upon slave/master configuration on register MCB control register bit 4. If MCB is set as Master, selectors are set to use local signals, else it is set to use external signals.

This configuration can be overset setting bit 0 of this register.

- **AutoSelect Bit** (`Switches[0]`): When set to '1', selectors are set to "autoselect". If unset ('0'), bits 1 and 2 are used as selectors for sync and sysclk.
- **LocalSysClk Bit** (`Switches[1]`): Selects whether we want local sysclk signal ('1') or we want external sysclk signal ('0')
- **LocalSync Bit** (`Switches[2]`): Selects whether we want local sync signal ('1') or we want external sync signal ('0')
- **CurrLocalSysClk Bit** (`Switches[3]`): (Read only) Current clock selector value
- **CurrLocalSync Bit** (`Switches[4]`): (Read only) Current sync selector value.

Resuming:

- For bits 4:1, a '1' means local and a '0' means external source (coming from sync interface).
- Bits 1:2 are used only when bit 0 is unset.
- Bits 3:4 may be diferent from bits 1:2 (only when bit 0 is set to '0')

[9] `http://www.datadelay.com/datasheets/3d3418.pdf`

### 1.3.3 Former Pixel fpga registers description

**1.3.3.1 Pixel FPGA LED Indicator Control Register (`ADDR_LEDLATCH`)** There are three bits that control the function of the Pixel FPGA indicator LED (D4). After power on or a reboot, the indicator will be on permanently until communication with the PAN has been established. After this, writing to one or more of the control bits will provide a 10ms indicator flash each time the specific function goes true. The functions that can be assigned for this are listed in Table 8.

Table 8: Pixel LED Indicator Function Codes

| MCB_PIXLEDCTL | |
|---|---|
| BIT | Function |
| 0 | Flash each time a data word (pixel or message) is written to the Slink FPDP port |
| 1 | Flash each time the Pixel FPGA is busy decoding a PAN CMD (functions with address ranges 0x0200 to 0x02FF). |
| 2 | Flash each time a pixel data value is written to the Slink FPDP port |

**1.3.3.2 Synchronization Link Serial Bit Register (`ADDR_EXTDATALATCH`)** There is a single bit available to be written and read by the PAN that is carried across the synchronization link ports. When reading this function, the data bit is that which appears on pin 1 of J3. When writing this bit, the data appears on pin 1 of J4. This enables a slow and primitive data exchange between DHE chassis belonging to the synchronized group.

**Note:** Currently this port has no function and is reserved for future use.

### 1.3.4 Slink Statistics registers

All 32 bits registers of slink statistics are splitted in 2 registers of 16 bits. In table 5, there are several registers address names finished in `_L` or `_H`. `_L` means that this register is the lower part (i.e. 16 LSB) while `_H` means that it is the high part (i.e. 16 MSB).

Slink has to type of data, “control” and “data”. “control” words are messages from pan or answers to messages, while “data” words are simply data (pixel data, data to write to memory,...)

Slink packs data into pages, which are saved into memory together. When Slink starts to send data it starts a new page. It closes this pages once it has to send a new type of data (“data” or “control”) or when it has been idle for some clock periods.

**1.3.4.1 Received Data Words (`ADDR_SLINK_SRDW`)** Number of data words received.
32 readable bits.

**1.3.4.2 Received Control Words (`ADDR_SLINK_SRCW`)** Number of control words received.
32 readable bits.

**1.3.4.3 Sent Data Words (`ADDR_SLINK_SXDW`)** Number of data words sent.
32 readable bits.

#### 1.3.4.4 Sent Control Words (ADDR_SLINK_SXCW)

Number of control words sent.

32 readable bits.

#### 1.3.4.5 Transferred Data Pages (ADDR_SLINK_SXDP)

Number of data pages sent.

32 readable bits.

#### 1.3.4.6 Transferred Control Pages (ADDR_SLINK_SXCP)

Number of control pages sent.

32 readable bits.

#### 1.3.4.7 Transferred Data Complete Pages (ADDR_SLINK_SXDCP)

Number of data complete pages sent.

32 readable bits.

#### 1.3.4.8 Transferred Control Complete Pages (ADDR_SLINK_SXCCP)

Number of control complete pages sent.

32 readable bits.

#### 1.3.4.9 Link Full flag Errors (ADDR_SLINK_SLFFE)

Contains the number of times, LFF has been set to true.

16 readable bits.

#### 1.3.4.10 Link Down Errors (ADDR_SLINK_SLDE)

Contains the number of times, LDOWN has been set to true.

16 readable bits.

#### 1.3.4.11 Up Time minutes (ADDR_SLINK_SUTM)

System uptime in minutes.

16 readable bits.

Cannot be write nor cleared.

#### 1.3.4.12 Up Time seconds (ADDR_SLINK_SUTS)

Seconds since last change in ADDR_SLINK_UTM.

16 readable bits.

Cannot be write nor cleared.

#### 1.3.4.13 Page Length Info (ADDR_SLINK_IPL)

How many words contains each Page. Transfers are splitted between pages of this length.

16 readable bits.

Cannot be write nor cleared through this address. To set this value, special functions of the slink must be used. See Slink Interface documentation.

#### 1.3.4.14 Max Wait Info (ADDR_SLINK_IMW)

Before closing a page, slink interface waits this value of clocks to see if more writes are coming.

16 readable bits.

Cannot be write nor cleared through this address. To set this value, special functions of the slink must be used. See Slink Interface documentation.

### 1.3.5 Microcode only registers

**1.3.5.1 MPU SEQUENCER Enables Register (`SEQ_ENBLREG`)** The MPU SEQUENCER Enables register can only be written to from microcode running on the MPU sequencer. It allows control over signals directly related to the MPU function. It cannot be read from. Table 9 outlines the functionality of this register.

Table 9: MPU Sequencer Enables Register Bit Definitions

| Bit | Function | Bit | Function |
|---|---|---|---|
| 31 | | 15 | Set SYNC_OUT signal high. |
| 30 | | 14 | Set SYNC_OUT signal low |
| 29 | | 13 | |
| 28 | | 12 | |
| 27 | | 11 | |
| 26 | | 10 | |
| 25 | | 9 | Set LSR Auto-cancel feature off |
| 24 | | 8 | Set LSR Auto-cancel feature on |
| 23 | | 7 | |
| 22 | | 6 | |
| 21 | | 5 | |
| 20 | | 4 | |
| 19 | | 3 | |
| 18 | | 2 | Stop Integration Time Counter |
| 17 | | 1 | Start Integration Time Counter |
| 16 | | 0 | Reset Start Exposure Flag |

- Bit 0 Reset SE Flag: Resets the Start Exposure flag in the Command register. This allows the microcode to avoid re-triggering on this flag when returning from a sequence.
- Bit 1 Start Integration Time Counter: Allows the microcode to elect when to start timing an exposure.
- Bit 2 Stop Integration Time Counter: Permits the microcode to elect when to stop the integration timer to accurately register the actual elapsed integration time.
- BIT 7 TO 3 NOT USED.
- Bit 8 Enable a feature of the MPU SEQUENCER that automatically cancels the board select line one clock cycle after the instruction LSR (Load Board Select Register) has been issued. This eliminates the need to specifically cancel the board select line in the microcode itself.
- Bit 9 Disable the automatic canceling feature described above. In this mode, after each LSR instruction, a zero would normally be written out to deselect the board to which the pattern memory data was sent. Disabling this feature can be useful to stream out patterns to a specific board / board address each MPU SEQUENCER clock cycle by using the INC and DEC microcodes to step through sequential pattern memory locations.
- BIT 13 TO 10 NOT USED.
- Bit 14 Set SYNC signal Low: Allows control over the sync signal used to synchronize multiple DHE chassis.
- Bit 15 Set SYNC signal High: Allows control over the sync signal used to synchronize multiple DHE chassis.
- Bit 31 to 16 Not Used.

# 2 Board Hardware Description

Check des-docdb#### document.

**ToDo:** Find MCB schematics document number at docdb. Include figures of schematics.

## 2.1 Power Supplies

**ToDoLaia:** ReadModifyWrite

The MCB requires three power supplies. The majority of the logic is powered by a 3.3v supply (+3.3VD). This supply can be either supplied from the backplane (P1) or generated internally from U38. The jumper JP5 selects the source for this supply. Note that the 'newer' Systran FPDP CMC modules require only a 3.3V supply, however, these models overstress the capabilities of the internal regulator and an external supply must be used. The internal logic cell of the FPGAs requires 1.8v (+1.8V) that is generated on board by U39. U38 is supplied from a 5v logic supply (+5VD) that must be present on the backplane through P1. U39 is supplied from the 3.3V supply. The nominal current draws on these supplies are shown in Table 10.

Table 10: Power Supply Current Draws

| Supply | Quiescent | Peak |
|---|---|---|
| 5v (with separate 3.3V supply) | 0.6 Amp | 1.7 Amp |
| 3.3v (Including Slink power) | 1.3 Amp | 2.0 Amp |
| 1.8v | | |

## 2.2 Logic Section

### 2.2.1 Clock Generation

All logic on the MCB and the DHE backplane is synchronous to the master clock generated by the MCB. If the MCB is mounted in a master DHE, then this 40MHz clock is generated by QZ1. If the MCB is in a slave DHE, then the master clock is fed through the synchronization interface at J3.

Selector U7 selects the clock source depending on master or slave use.

I476 provides a means to adjust the clock phase to enable close clock phase matching between master and slave DHE chassis. The MCB internal clock is buffered by U13 and fed to the FPGA through its global clock IO pin.

A backup clock from the oscillator QZ1 is bufered by IC10 and sent to another FPGA clock IO pin. This will let us implement a recovery mode in case a board is set as slave and it has no clock signal at connector J3.[10]

An external clock is fed out to slave DHE chassis through U7 and J4.

An analog of the DHE clock is available at the connector PX1 to provide a connection to an oscilloscope for clock synchronization.

Board clocks to the peripheral boards mounted in the backplane are gated versions of the internal clock generated within the Sequencer FPGA and controlled by the Clock Enable register.

The system clock is also supplied to Slink module.

[10]This is not implemented yet. Future releases of the firmware may have it.

### 2.2.2 Reset and FPGA Boot Logic

**ToDo:** Tell about jumper (ST3), monoestable, reprogram...

**2.2.2.1 Jumper ST3** This jumper changes functionality of front panel push button. It has three logical positions:

- Disconnected: When jumper is not placed, front panel push button is disabled.
- Connected to Voltage regulators: When push button is pressed it disables voltage regulators. When released, firmware is reloaded into fpga.
- Connected to fpga: This is an asynchronous reset of the firmware. When front panel push button is pressed it reloads default values to registers, state machines...

### 2.2.3 JTAG Interface

| Pin | Function |
|---|---|
| 1 | +3.3VD |
| 2 | DGND |
| 3 | KEY |
| 4 | TCLK |
| 5 | N.C. |
| 6 | TDO |
| 7 | TDI |
| 8 | TMS |

Table 11: JTAG Pin Assignments

### 2.2.4 Options for Building the MCB

Firmware package is released as a bundle which contains among other things, sources and a makefile.

This makefile can be tuned to change priorities (speed versus area) of synthesis.

Inside sources folder there is a file named `constantsPkg.vhd`. This file contains all defined constants of the firmware (mostly register addresses). If needed, this make easier to change default values or addresses (think it twice before changing a register address).

# 3 Board Specifications

**ToDo:** Update values

# 4 Appendix

## 4.1 Appendix I - Board Identity and Firmware Revision Codes.

THE BOARD IDENTIFIER IS CODED AS A 16-BIT UNSIGNED INTEGER (I.E. CODES 0 TO 65535 ARE AVAILABLE).

WHEN READ AS AN ATTRIBUTE, THE INTEGER DECIMAL VALUE OF THE ENCODED VALUE SHOULD REPRESENT THE PRODUCT CODE OF THE PHYSICAL HARDWARE.

CURRENT NOAO IMPLEMENTATIONS OF THIS CODE ARE RESERVED AND SHOWN BELOW:

- 100 = 36 CHANNEL IR ACQUISITION MOTHER BOARD REVISION A
- 200 = 18 CHANNEL IR ACQUISITION DAUGHTER BOARD REVISION A
- 300 = CLOCK AND BIAS BOARD REVISION A
- 400 = MASTER CONTROL BOARD REVISION A
- 500 = CCD ACQUISITION BOARD - PROTOTYPE

THE FIRMWARE REVISION IDENTIFIER CONTAINS A UNIQUE PATTERN TO IDENTIFY THE FPGA CODE VERSION CURRENTLY LOADED ONTO THE HARDWARE.

CURRENT IMPLEMENTATION USES THE SAME 16-BIT UNSIGNED INTEGER CODING SCHEME TO DESCRIBE A THREE DIGIT MAJOR + TWO DIGIT MINOR VERSION SET.

WHEN READ AS AN ATTRIBUTE, THE DECIMAL EQUIVELENT OF THE ENCODED VALUE SHOULD BE DIVIDED BY 100 TO SEPARATE THE MAJOR AND MINOR REVISION NUMBERS.

TWO SPECIAL CASES ARE RESERVED FOR CODE DEVELOPMENT AND TESTING.

VERSION 000.XX IS CONSIDERED UNDER DEVELOPMENT OR UNASSIGNED.

VERSION 9XX.XX IS CONSIDERED AS DEBUG CODE WHERE THE TWO REMAINING MAJOR AND TWO MINOR CODES REPRESENT THE SOURCE VERSION FOR THIS DEBUG CODE.

IT IS SUGGESTED THAT MAJOR CODES ARE RESERVED FOR CHANGES TO FUNCTIONS AND PROTOCOLS THAT AFFECT THE INTEROPERABILITY OF BOARDS WITHIN A DHE.

MINOR CODES ARE USED TO DIFFERENTIATE ON-BOARD FUNCTIONALITY CHANGES THAT DO NOT AFFECT OTHER BOARDS WITHIN A DHE.

## 4.2 Appendix II - Using the MONSOON SYNC Function

**ToDo:** Update to new MCB

### 4.2.1 Introduction

The use of the synchronized logic contained in the hardware / firmware of the MONSOON image acquisition system allows multiple Detector Head Electronics (DHE) chassis to operate in close proximity to each other. Multiple DHE systems are typically required for large focal planes where the number of video channels exceeds the capabilities of a single DHE. Without synchronization, each DHE will generate electrical fields that can potentially interfere with an adjacent DHE chassis leading to an increase in system noise.

To obtain synchronization two criteria must be met:

- Each DHE needs to have an internal system clock signal that is locked in phased to a master clock.
- Any activity that is sensitive to induced noise, such as reading out of a detector, must be 'lock stepped' to ensure that interference sources are seen as common mode noise sources by all DHE chassis.

### 4.2.2 MONSOON Implementation

The requirements for synchronous operation are met in MONSOON by the software configuration of several PAN attributes, two front panel connectors on the Master Control Board (MCB), and the use of a synchronization bit in the sequencer code running on the MCB.

The software attributes control the designation of each DHE as a master or slave and establish an appropriate delay that allows each DHE enough time to have received the 'Start Exposure' command from its local PAN. In addition, software attributes control the system clock phase control (by way of U19) to align the different DHE clock phases. These attributes are described further in Table 1.

The connectors are designated as "Sync In" (J9) and "Sync Out" (J18). The normal topology for the Master / Slave cabling is a daisy chain where a cable is plugged into the designated Master DHE 'Sync Out' connector and the first Slave DHE 'Sync In' connector. Successive Slave DHE chassis are then linked in the same way. Figure 1 shows this scheme. The requirement for the synchronization signals is that the combined delay associated to the cable length and line receivers / transmitters will not exceed 50ns total. If this condition is not met by the simple passive daisy chain topology, a 'Star' topology can be implemented using a powered repeater.

Routing of the hardware signals in each MCB is achieved through jumpers JP2, JP3, and JP11. Table 2 describes the appropriate jumper positions for each mode. Figure 2 shows a schematic of the signal routing on the MCB.

There is a dedicated bit in the testable Sequencer control register that acts as the synchronization flag for sequencer code running the readout and other sequencer programs. This bit (bit 1 of the SEQ_CMDS register at MCB address 0x0102) is controllable through the EFR register of the sequencer when the mode is set Master. Delaying sequencer program flow until this bit changes state allows both the master and slave DHE sequencers to initiate a program in 'lock step' with all other DHEs. Paragraph 4.2.4 details the basic mechanism of this.

### 4.2.3 Sample Assembler Code for Synchronized DHE Function

```
*****************************************************
** PROJECT : MONSOON ENGINEERING
** TITLE : EXEC PORTION OF TEST ALADDIN CODE MODIFIED TO ACHIEVE SYNCRONIZATION BETWEEN
** : TWO OR MORE DHE MODULES.
** VERSION : 002
** DATE : 08/30/2006
** AUTHOR : PETER MOORE
** HISTORY : 12/09/2003 PCM ORIGINAL VERSION SCULPTED FROM EXEC ON ALADDIN.
** : 08/30/2006 PCM ADOPTED CODE TO REFLECT CCD USE OF ASM4. UPDATED JUMPER
** : DESIGNATIONS FOR MCB REVISION A HARDWARE.
**
** DESCRIPTION :
** : DUMMY SEQUENCER EXEC PORTION TO EXERCISE AND DEMONSTRATE THE SYNC
** : FUNCTIONALITY.
** :
** : NORMAL READOUT CODE IS REPLACED BY A DUMMY STUB THAT JUST RETURNS.
** : FOR ACTUAL USE YOU WOULD INSERT YOUR CODE AT THE STUBS
** :
** : TO USE THIS CODE THE MASTER AND SLAVE DHES NEED TO BE LINKED WITH THE SYNC
** : CABLE. MASTER SYNC OUT => SLAVE SYNC IN. THE SLAVE DHE SHOULD BE SET TO
** : USE THE EXTERNAL CLOCK (JP3 2=>3) AND ADJUSTED IN PHASE (VIA THE MCB_CLKINCTL
** : AND MCB_CLKOUTCTL FUNCTIONS) TO BE COINCIDENT WITH THE MASTER SYSTEM CLOCK.
** : THE SYNC CABLE CONNECTING THE MASTER AND SLAVE DHES MUST HAVE LESS THAN
** : 45ns OF DELAY IN ITS ENTIRE LENGTH (approximately 7 meters physical length).
** :
** : JP2 NEEDS TO BE JUMPERED FOR EXTERNAL SYNC IN THE SLAVE DHE AND REMOVED
** : IN THE MASTER DHE.
** :
** : THE PAN SETS THE MASTER AND SLAVE MODE IN THE RELEVENT DHES VIA
```

```
** : THE MCB CONTROL REGISTER BIT 15. BIT SET INDICATES MASTER MODE.
** :
** : THE PAN THEN ISSUES A NORMAL "START␣EXPOSURE" COMMAND WITH A VECTOR (1 IN THIS
** : EXAMPLE). ALL SEQUENCERS JUMP OUT OF THEIR IDLE MODE AND AWAIT THE ARRIVAL OF SYNC.
** : SYNC IS DELAYED BY A FIRMWARE DELAY GENERATOR BY A VALUE THAT EXCEEDS THE
** : ESTIMATED MAXIMUM LATENCY OF THE START EXPOSURE COMMAND REACHING EACH
** : INDIVIDUAL PAN NODE. THE SYNC IS EMMITED BY THE MASTER TO ALL SLAVES AFTER THE
** : DELAY. THE MASTER RECEIVES ITS OWN SYNC SIGNAL IN THE SAME FASHION (AND AT THE SAME
** : TIME) AS THE SLAVES. WHEN SYNC IS RECEIVED AND RECOGNIZED BY THE SEQUENCER BRANCH
** : INSTRUCTION, ALL SEQUENCERS RUN THE SAME CODE ON THE SAME CLOCK EDGE.
** :
**
** RESOURCES :
**
************************************************************
** JUMP FLAG DEFINITIONS - BIT POSITION IN COMMAND REGISTER **
************************************************************
DEF #SYNC 1 * Jump on sync bit high
DEF #ITC 2 * Jump on Integration time expired
DEF #START 3 * Jump on start exposure flag set
DEF #CCDREADOUT 4 * STRVCTR = 1: DO COMPLETE CCD CLEAR - INTEGRATE - READ
DEF #CCDCLEAR 5 * STRVCTR = 2: DO CCD CLEAR ONLY
DEF #CCDREAD 6 * STRVCTR = 4: DO CCD READ ONLY
DEF #VCTR_4 7 * STRVCTR = 8:
DEF #VCTR_5 8 * STRVCTR = 16:
DEF #VCTR_6 9 * STRVCTR = 32:
DEF #VCTR_7 10 * STRVCTR = 64:
DEF #VCTR_8 11 * STRVCTR =128:
DEF #CONT_RUN 12 * USER BIT 1: LOOP INDEFINATELY ON TESTS
DEF #USR_2 13 * Jump on user mode bit value 2
DEF #USR_3 14 * Jump on user mode bit value 4
DEF #CONT_CLR 15 * Jump on user mode bit value 8
**
*****************************
** EFR REGISTER CONSTANTS **
*****************************
DEF #KILL_EXPFLG x00000001 * Cancel start exposure flag using the EFR register
DEF #STRT_INTCNTR x00000002 * Start the integration time counter using the EFR register
DEF #STOP_INTCNTR x00000004 * Stop the integration time counter using the EFR register
DEF #LSR_AUTO_KILL x00000100 * Enables automatic board select cancel after an LSR instruction
DEF #LSR_MAN_RESET x00000200 * Disables automatic board select cancel. LSR will remain active
*until an LSR #DSLCT is issued
DEF #SET_SYNC_LOW x00004000 * If enabled as master DHE, set sync bit low after sync delay
DEF #SET_SYNC_HIGH x00008000 * If enabled as master DHE, set sync signal high immediately
DEF #KILLANDSYNC x00004001 * Set sync low and kill start exposure flag
**
*****************************
** MODE REGISTER CONSTANTS **
*****************************
DEF #RESET 0 * Mode register constant for board reset operation
DEF #READ 1 * Mode register constant for read operation
DEF #WRITE16 2 * Mode register constant for 16 bit write operation
DEF #WRITE32 3 * Mode register constant for 32 bit write operation
**
**********************************
** LSR BOARD SELECT ASSIGNMENTS **
**********************************
DEF #DSLCT 0
DEF #MCB 1
*******************************
** PATTERN MEMORY DEFINITIONS **
*******************************
** PATTERN MEMORY SEGMENTATION MAP
DEF #PATSEG_00 x0000 * Pattern memory origin - Initialization stuff
**
** ADDITIONAL DEFINITIONS USUALLY GO HERE
**
**
***************************************
** INSTANTIATE PATTERN MEMORY VALUES **
***************************************
ORG PAT #PATSEG_00 * ********* GENERAL CONSTANTS KEPT IN PAT MEM *********
PAT #KILL_EXPFLG #KILL_EXPFLG * We can use the same label name here because
*they are in different contexts
PAT #STRT_INTCNTR #STRT_INTCNTR * The definition earlier points to a data value. The
PAT #STOP_INTCNTR #STOP_INTCNTR * label here points to a pattern memory address
PAT #SET_SYNC_LOW #SET_SYNC_LOW * CODE VALUE TO WRITE TO EFR TO SET SYNC PIN LOW
PAT #SET_SYNC_HIGH #SET_SYNC_HIGH * CODE VALUE TO WRITE TO EFR TO SET SYNC PIN HIGH
***********************************
* PROGRAM CODE SEGMENT STARTS HERE **
***********************************
** EXECUTIVE ROUTINE
#INIT CAL #INIT * SET DETECTOR UP FOR INITIAL CONDITIONS
LMR #WRITE32
LPP #SET_SYNC_HIGH
LSR #MCB * SET SYNC BIT HIGH AS INITIAL CONDITION
#EXEC JCB #GOBABY #START * IF START EXPOSURE FLAG SET
JMP #EXEC * ELSE JUMP BACK TO EXEC
#GOBABY LPP #KILL_EXPFLG * KILL THE START EXP FLAG PATTERN = 0x00000001
LSR #MCB * WRITE TO THE EFR
LPP #SET_SYNC_LOW
LSR #MCB * TRIGGER THE SYNC SIGNAL. NOTHING HAPPENS HERE
                        * UNTIL THE DELAY HAS EXPIRED.

#WAIT JCB #WAIT #SYNC * SYNC GOES FALSE SYNCHRONOUSLY ON ALL DHES AFTER DELAY
LPP #SET_SYNC_HIGH
LSR #MCB * RESET SYNC BIT HIGH
JCB #JRDOUT #CCDREADOUT * DO CLEAR - INTEGRATE - READOUT OPERATION
JCB #JCLEAR #CCDCLEAR * DO CLEAR ONLY
```

```
JCB #JREAD #CCDREAD * DO READ ONLY
JMP #EXEC * NO FLAGS SET, GO HOME.
#JRDOUT CAL #CLEAR
LPP #STRT_INTCNTR
LSR #MCB * START INTEGRATION TIMER RUNNING
NOP * A bug in the MPU .. needs time to reset the flag
NOP
#RD1 JCB #RD2 #ITC * WAIT FOR INTEGRATION TIME
JMP #RD1
#RD2 LPP #STOP_INTCNTR
LSR #MCB * STOP INTEGRATION TIMER
CAL #ARRAY_READ
JCB #JRDOUT #CONT_RUN * DO IT ALL AGAIN IF TOLD TO DO SO
JMP #EXEC * CALL IT A DAY
#JCLEAR CAL #CLEAR
JMP #EXEC
#JREAD CAL #READ
JMP #EXEC
```

Listing 1: Sample assembler file

## 4.3 Appendix III – Master Control Board Sequencer MPU Description

**ToDo:** Update to new MCB. Performance, interface, architecture and register is the same but it changes small details (like which fpga we've used or the implementation of the memories.

### 4.3.1 Introduction

The sequencer for the Master Control Board (MCB) is implemented as a completely imbedded 4-bit (16 instructions) micro-controller. A micro-code program loaded into the sequencer program memory controls the logic flow of the sequence. A fixed instruction decode and execution cycle is of great value, particularly for timing critical waveform generation. Therefore, the sequencer is designed with a uniform one-clock-cycle execution time for all 16 instructions. The assembly micro-code for the sequencer can be easily customized to change functionality and adjust the required timing.

### 4.3.2 Sequencer Performance

The current sequencer design implementation performs satisfactorily at 20 MIPS on the Virtex XCV 300 PQ 240 and meets all worst-case timing requirements including clock skew analysis with a positive slack of more than 1 ns. The current performance of 20 MIPS is very modest but meets the objective of single-clock-cycle execution without any branching latency penalties. It should be noted that the sequencer currently operates at the maximum frequency of 20 MHz = ½ system clock frequency of 40 MHz.

### 4.3.3 Sequencer Interface

The sequencer interface symbol is shown in Figure 1. The sequencer derives its clock internally from the sysclk. Only the decode execution unit of the sequencer operates on a seqclk derived from the global system clock sysclk. All internal registers and memory elements except for the stacks are accessible through a common data write port and individual read port which is multiplexed external to the sequencer. The address decode of all components of the sequencer is also implemented external to the sequencer and individual input write strobes are asserted when the corresponding components are addressed for writes. The data in all sequencer registers based on Select-RAM is always available for read during the same clock cycle in which they are addressed. However, the Block RAM-based program memory and pattern memory elements incur one-clock-cycle latency before the data corresponding to their address is available on their output ports. The sequencer accesses the Sequencer Bus by asserting a bus request signal and waiting for an acknowledge signal. This wait blocks the sequencer execution and is critical for timing sensitive delay instructions. However, if the acknowledge signal is asserted before the next sequencer rising edge, no additional delay is incurred.

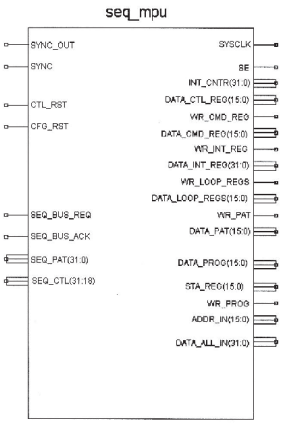


Figure 7: Sequencer Interface Symbol

### 4.3.4 Architecture Overview

The sequencer is based on Harvard architecture with separate program memory and pattern memory implemented using the Block RAM resources. Apart from the conventional branching instructions, dedicated looping instructions are also included in the sequencer instruction set, therefore the sequencer also includes an address stack, loop address stack and loop count stack memories to support the nested looping and branching functionality. Five types of special function registers, Enable, Command, Loop Count, Integration and Control, are also included for their specific functions. A central Decode Execution Unit (DEU) processes the sequencer micro-code to implement the programmed control flow and timing. A clock control unit, which includes a sequencer clock divider and clock buffer, is included for programmable clock cycle and low skew operations. The detailed functionality and implementation of each of these sequencer components is described later in this section.

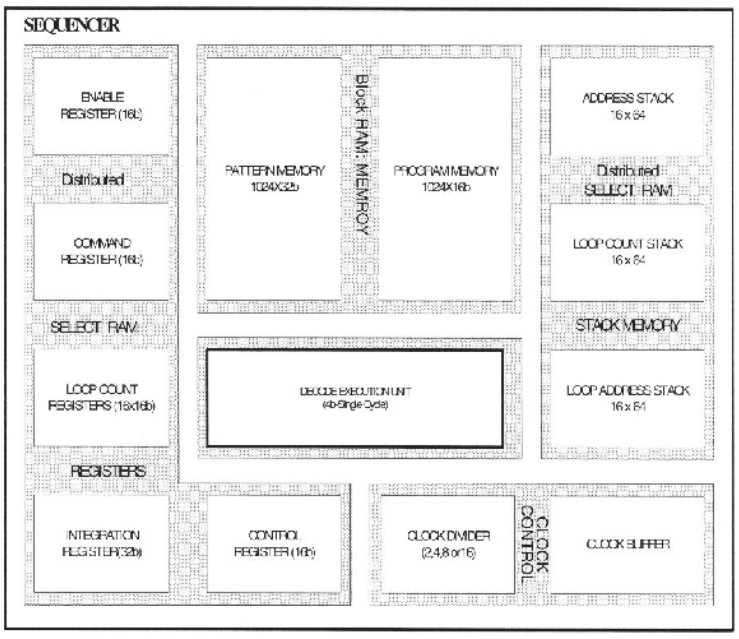


Figure 8: Sequencer Components

### 4.3.5 Program Memory (BRAM)

The program memory is implemented using the Block RAM resources. The program memory is dual port, 16 bits wide and 1K (1024 words deep, read-write RAM). Both ports are clocked with the same system clock (SYSCLK 40 MHZ). Conventional pipeline-based architecture results in pipeline refill latencies while executing branching instructions that are unacceptable for single clock execution. To avoid such latencies, one port of the program memory is dedicated to substitute for pipelining operation. This is achieved by performing a Program counter + 1 read at the second port of the program memory. Access to this port is limited to the DEU only. Access to the other port of the program memory is time multiplexed between the PAN and the sequencer DEU, where the PAN is given the higher priority. The result is that the DEU's access to program memory is suspended when the PAN is reading or writing to the sequencer memory. This inserts a blocking delay in sequencer execution so the timing information encoded into the sequencer micro-code gets altered. Because of this, program memory read and writes should be avoided during timing-critical normal mode of sequencer operation.

### 4.3.6 Pattern Memory (BRAM)

The pattern memory is dual port 32K bits of read-write RAM. The two ports of pattern memory include independent read-write PAN port and read only DEU port but differing in width and depth. From the PAN port the program memory is 16 bits wide and 2K words deep whereas from the DEU port it is 32 bits wide and 1K patterns deep. The even and odd addressed 16-bit words correspond to the least and most significant 16 bits of a pattern respectively. The dual port access to the pattern memory is not time multiplexed so both PAN and DEU can randomly access the program memory independently. The pattern memory is clocked using the same system clock (SYSCLK 40 MHZ). The pattern memory is implemented using the Block RAM resources. It should be noted that patterns in the program memory are not cleared on reset and require explicit writes (with 0 data) from the PAN to clear the contents of the Pattern Memory.

### 4.3.7 Stack RAMs (Select-RAM)

Program Address Stack RAM and Loop begin address Stack RAM provides 12 bits wide and 64 words deep buffer spaces each to implement the Program Pointer Stack and Loop Begin Address Stack respectively, whereas the loop count Stack RAM is 16 bit wide and also 64 words deep. The Program Address is pushed into the Program Address Stack RAM whenever a sub-routine is called and is popped back at the end of sub-routine execution. Similarly, to support the nested looping instructions at the beginning of the loop, the next program memory address and previous contents of the loop counter are pushed into their respective stacks, whereas the loop counter is loaded with the new count value. After execution of iteration, if the loop counter is greater than zero, it is decremented and program pointer is re-loaded with the loop beginning address stored at the top of the stack. Otherwise the program pointer is incremented to point to the next instruction and loop count and loop starting address are both popped out from the sequencer stack. All three Stack RAMs are clocked from the system clock (SYSCLK 40 MHZ). The Stack RAMs are not transparent to the PAN and are controlled by the DEU only. Stack RAMs are implemented using Distributed Select RAM resources.

### 4.3.8 Special Function Registers

As mentioned earlier, the current implementation of the sequencer includes the following five different types of special function registers:

- Command Register
- Control Register
- Loop Count Register
- Clock Control
- Decode Execution Unit

**4.3.8.1 Command Register Definition** This register is a combination of hardwired and software settable flags. The intended purpose is to provide a conditional jump capability within the sequencer micro code by using the sequencer instruction "Jump if Control Bit Set" (JCB). This instruction allows any of the 16 flags to be tested for an active condition.

**4.3.8.2 Bit 0: Unconditional Jump** Always tied to '1' to facilitate the implementation of unconditional jump.
Default 1 Active 1.

**4.3.8.3 Bit 1: SYNC_IN** This bit reflects the condition of the EXT_SYNC signal when MCB jumper JP17 is correctly configured for slave mode. In Master mode this bit reflects the status of the SYNC_OUT RS flip flop controlled by the Enable Function Register (EFR).
Default 0 Active 1.

**4.3.8.4 Bit 2: Integration Timer Terminal Count** This bit is set when the Integration Time counter (ITC) reaches the value stored in the Required Integration Time (RIT) register (at MCB address 0x0000). The flag will remain set until either the RIT register is loaded with a larger value than the ITC contents or asserting the enable bit via the EFR activates the integration timer. Default condition (after reset) is active since the ITC and the RIT are reset to zero. If this bit is to be used to determine if the integration timer is active, the PAUSE bit in the control register should also be checked.
Default 1 Active 1.

**4.3.8.5 Bit 3: Start Exposure Active High** This bit is set when the DHE receives a "Start Exposure" command from the PAN. It remains set until a reset start exposure flag bit is set via the EFR. The purpose of this bit is to act as a qualifier for bits 4 to 11 of this register.
Default 0 Active 1.

**4.3.8.6 Bit 4 to Bit 11: Start Exposure vector** These bits are latched from the address portion of the start exposure command at the same time as bit 3. They are meant to enable a variety of sequence actions to be selected based upon the bits set and the use of the JCB instruction in the sequencer micro code.
Default 0 Active 1.

**4.3.8.7 Bit 12 to Bit 15: User Defined Bit Pattern** These bits are written directly to the command register from the PAN by writing to the register address. They are transposed from bits 0 to 3 in the data written to the register.
Default 0 Active 1.

### 4.3.9 Control Register Definition

This register controls the general behavior of the Master Control Board. The principal functions are listed below.

**4.3.9.1 Bit 0: MPU Run:** Asserted by writing to the MCB at address 0xFFFC. Asserting this bit allows the sequencer to begin decoding and executing micro code. Whenever the signal is de-asserted, the MPU enters into the Hold State after executing the current instruction. All the board selects and the request signal are masked during the non-active hold state. The program counter is reset to zero.

Default 0 Active 1.

**4.3.9.2 Bit 1: PAUSE ITC:** Setting this bit to 1 will stop the Integration Time Counter (ITC) from incrementing. Resetting this bit will allow the ITC to continue counting up.

Default 0 Active 1.

**4.3.9.3 Bit 3 & bit 2: Sequencer Clock Pre-Scalar** This 2-bit pattern defines the base sequencer clock frequency. Note that the current maximum frequency is half the system clock frequency. Each micro code instruction takes one sequencer clock cycle to complete.

- 00 SYSCLK / 1 20 MHz
- 01 SYSCLK / 2 10 MHz
- 10 SYSCLK / 4 5 MHz
- 11 SYSCLK / 8 2.5 MHz

Default 00.

#### 4.3.9.4 Bits 4-14: Currently Not Used

**4.3.9.5 Bits 15: Slave DHE** When set this bit disables the output of the SYNC_OUT signal and enables the reception of SYNC_IN signals to the CMD register. When this bit is not set the SYNC bit in the CMD register reflects the state of the SYNC RS Flip Flop controlled through the EFR.

Default 0 Active 1.

### 4.3.10 Loop Count Registers

There are 16 Loop Counters. These 16-bit registers can be loaded directly from the PAN and are intended for use where repetitive cycles of clock sequences are needed. The sequencer instruction "Loop Register Begin" (LRB) identifies the first instruction to be repeated and one of the 16 loop registers. The "Loop End" (LPE) instruction identifies the end of the repeated block of micro code and will repeat the sequence until the count stored in the loop register has been completed.

**4.3.10.1 Enable Function Register (EFR)** The EFR is accessible only through the micro code. This register controls strobes that are set and reset by executing a "Load Select Register" (LSR) instruction with the bit 0 set in the operand and the MODE Register set to a 32-bit write code, that is, writing to the MCB board itself with a WRT32 mode. The contents of the current pattern pointer are written to this register with the following effects:

Table 12: Enable Function Register

| Bit | Function | Bit | Function |
|---|---|---|---|
| 31 | | 15 | Set SYNC_OUT signal high. |

| Bit | Function | Bit | Function |
|---|---|---|---|
| 30 | | 14 | Set SYNC_OUT signal low |
| 29 | | 13 | |
| 28 | | 12 | |
| 27 | | 11 | |
| 26 | | 10 | |
| 25 | | 9 | Set LSR Auto-cancel feature off |
| 24 | | 8 | Set LSR Auto-cancel feature on |
| 23 | | 7 | |
| 22 | | 6 | |
| 21 | | 5 | |
| 20 | | 4 | |
| 19 | | 3 | |
| 18 | | 2 | Stop Integration Time Counter |
| 17 | | 1 | Start Integration Time Counter |
| 16 | | 0 | Reset Start Exposure Flag |

**4.3.10.2 Integration Time Register and Control** The Required Integration Time Register is 32 bits wide and thus can be programmed by PAN with integration time of minimum of 1 millisecond up to a maximum of approximately 50 days. The integration period delay is implemented using an internal up counter. When the start bit of the Enable Function Register is asserted, the internal counter is reset to zero and it starts counting up every millisecond until a terminal count equal to the value loaded into the RIT register is exceeded. The TC flag in the command register is negated (0) until the ITC is less than RIT value. The ITC is paused by the pause ITC bit in the control register is asserted.

### 4.3.11 Clock Control

The sequencer clock is programmable by configuring the control register appropriately. Refer to the description of the Control Register. Thus for debugging purposes the sequencer can by slowed by a factor of 2, 4 or 8. However, the Virtex FPGA doesn't provide BUFGMUX modules to multiplex multiple clocks. To minimize the clock skew across combinational logic, the sequencer clock divider and multiplexer are implanted by using the clock enable based implementation. To route the sequencer clock on high speed fast global resources of the FPGA, the output of the clock divider is buffered through on-chip clock buffers. Note that the delays implemented are currently with respect to the fastest sequencer clock frequency of 20 MHz so if the sequencer clock is divided by a factor, all delays specified will get multiplied by the same factor.

### 4.3.12 Decode Execution Unit (DEU)

The decode execution unit is the central control logic of the sequencer. Based on the functionality performed by various instructions, the control logic internal to the Decode execution unit is grouped into various control and data path elements, which collectively perform various functions as briefly explained in sequel.

**4.3.12.1 Program Address Multiplexer** As mentioned earlier, the DEU's access to the program memory Multiplexed Port is time-shared with the PAN Local address. The Program Address Multiplexer implements this priority multiplexed port.

**4.3.12.2 Program Counter Control Logic** Increments the program counter by the size of the instruction (nibble) executed while performing sequential (non-branching) instructions. For branching and sub routine call instructions the counter is loaded with the address specified by the operands. While executing subroutine return and looping instructions, the program counter is loaded with address on the top of the program pointer address stack and loop begin address stack respectively. Apart from this program counter output, another output equal to "Program Counter + 1" is also provided to address the program memory Pipeline Port.

**4.3.12.3 Instruction Multiplexer** The current instruction set implementation supports a variable instruction size from 1 to a maximum of 5 nibbles or 20 bits. So the maximum size instruction consists of 1 nibble opcode and 4 nibbles operand. The 32-bit data read per clock cycle from the two ports of the Program memory includes 16 bit data at the current memory location addressed by the program counter and the next 16 bits located at the next memory location. The opcode can be the first, second, third or fourth significant nibble of the 16-bit multiplexed port program word if the value of the least significant 2-bits of the program counter is b'00, b'01, b'10 or b'11 respectively. The operand for the maximum size instruction is the following 4 nibbles. Therefore, from the 32 bits program memory data, the Instruction Multiplexer selects the 20 bits valid instruction based on the least significant 2 bits of the program counter at the beginning of every execution cycle.

**4.3.12.4 Pattern Pointer Control Logic** Logic initializes the pattern pointer to 0 when Reset is asserted. Decodes and executes LPP, IPP and DPP instructions when the sequencer run flag is asserted.

**4.3.12.5 Output Registers and Control Logic** Apart from the pattern pointer sequencer include three other static registers, namely Select Register, Device Address Register and Output Register to control the sequencer bus. The control logic decodes and executes the LSR, LMR and LDA instructions to load the operand data into Select registers, Mode register and Device Address register on respective execution.

**4.3.12.6 Program Counter Stack Control Logic** Decodes sub-routine Call and Return instruction and performs Stack Memory push (write) and pop (read) operations respectively.

**4.3.12.7 Loop Control Logic** Controls the Loop Counter and selects the addressed Loop Register from the bank of 16 loop registers using an internal Loop register Multiplexer while executing looping instructions. Also controls the push and pop operations on Loop Count Stack and Loop Begin Address Stack for implementing nested loop functionality.

**4.3.12.8 EFR Write Decode** Asserts a write to the enables register when local select bit in the select register is asserted and the pattern of the mode register bits correspond to the write mode.

**4.3.12.9 Delay Control Logic** Three delay counters, clock-counter, microsecond counter and millisecond counter are designed to insert the explicitly specified blocking timing delays apart from the non-blocking integration time controller. The delay control logic decodes the delay instruction and loads the corresponding resolution down counter with specified value in the operands. Once the counters are loaded, they start decrementing to implement the specified delay and block the sequencer from executing the next instruction until all counters are reduced to inactive zero state. On reset, all three counters are initialized to zero value.

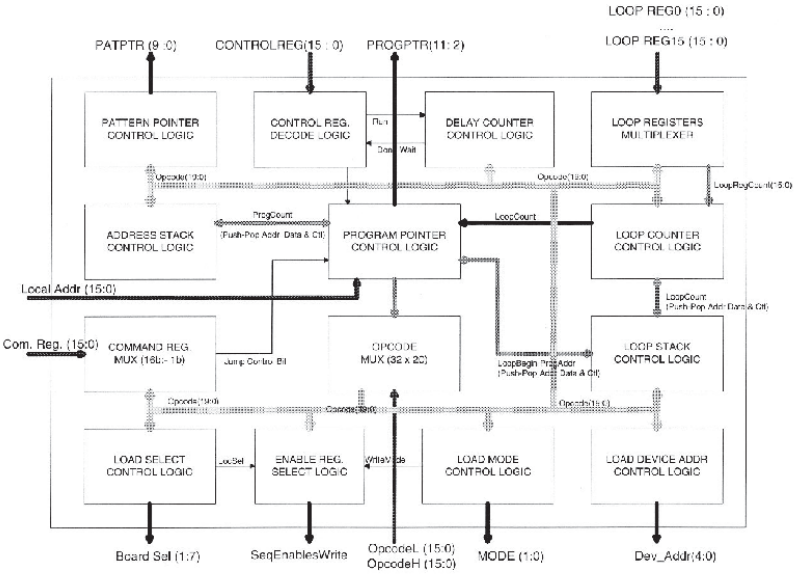


Figure 9: Decode Execution Unit

### 4.3.13 Instruction Set for the Sequencer

Instead of controlling the 32 sequencer signals (max) on the Sequencer address and data bus individually, the Sequencer Data Bus is grouped and is controlled as a 32-bits wide binary vector including the address and data bus. All the 32-sequencer signals are controlled simultaneously by changing the 32-bit word, defined as a Pattern, instead of switching one signal individually. Any arbitrary desired waveform can be expressed as a sequence of pattern with appropriate delay for each pattern. The output of the Sequencer can be divided into such elementary control-waveforms as Row-Shift Column-Shift Waveform. The Sequencer architecture includes a 32-bit wide and 1K deep memory called the Pattern Memory implemented using FPGA Block RAM. All valid 32-bit patterns required for generating the desired waveforms are loaded into the Pattern Memory and are accessed by Pattern Memory address. By sequentially changing the address of the Pattern Memory after the required amount of delays desired, waveform can be generated on the Sequencer Address Data-Bus.

One can observe that for any defined waveform, the Pattern Memory access will be generally sequential except for some exceptions. The Sequencer CPU indirectly addresses the Pattern Memory by a Pattern Pointer (similar to the Data Pointer in 8051). All instructions have a unique binary Opcode 1-nibble wide and 0-4 nibble wide operand based on the function executed by the instruction. Thus an instruction set of maximum 16 instructions with the longest instruction being 5 nibble or 20 bits wide and the shortest instruction being 1 nibble or 4 bits wide. However, all instructions will be executed in a single SEQCLK Based on the requirements and functionality executed, the instructions are classified as:

1. Output Instructions

2. Control Instructions

3. Delay Instructions

### 4.3.14 Output Instructions

Six different instructions are included to control all the output signals to the back-plane Sequence Bus. The first 3 instructions are designed to mange the pattern pointer efficiently with goals to optimize the program size and at the same time provide enough flexibility to the programmer to control the complete Sequence Bus. The next 4 instructions provide the programmer with necessary instructions to control the Sequence-Mode bits, Device Address bits and Board Select signals.

**4.3.14.1 LPP: Load Pattern Pointer: 4 Nibble (16 bit)** This instruction loads the Pattern Pointer of the Sequencer CPU with the immediate 10-bit address specified by the next three nibbles in the program memory. Note that since the address space of the Pattern Memory for the current design is limited to 1K, the most significant 2 bits of Nib1 are don't care and the other two bits specify the most significant bits of the pattern address PA9 PA0.

LPP: "Pattern Address"

| Nib 00 | Nib 01 | Nib 02 | Nib 03 |
|---|---|---|---|
| 0001b | XX | 10-bit Pattern Address. PA9-PA0 | |

**4.3.14.2 IPP: Increment Pattern Pointer: 1 Nibble (4-bit)** On execution, the Pattern Pointer is incremented by one. It is useful when waveform is running through a sequence of consecutive patterns in ascending order.

IPP:

| Nib 00 |
|---|
| 0010b |

**4.3.14.3 DPP: Decrement Pattern Pointer: 1 Nibble (4-bit)** Decrements the Pattern Pointer by one and is useful when waveform is running through a sequence of consecutive patterns in descending order.

DPP:

| Nib 00 |
|---|
| 0011b |

**4.3.14.4 LDA: Load Device Address (12-bit)** It loads the 5-bit Sequence Device Address Register, which is the output register for the DEV_ADDR [4:0] signals and the D3, D2 & D1 bits of the first nibble are don't care.

LDA: "Device Address"

| Nib 00 | Nib 01 | Nib 02 |
|---|---|---|
| 0100b | XXX | Dev_Addr(4-0) |

**4.3.14.5 LMR: Load Mode Register 3 Nibble (12-bit)** It loads the 2-bit Sequence Mode register, which is the output register for the SEQ_MODE [1:0] signals and the D3 & D2 bits of the first nibble are don't care.

LMR: "Mode Value"

| Nib 00 | Nib 01 | | | |
|---|---|---|---|---|
| 0101b | x | x | SQM1 | SQM0 |

**4.3.14.6 LSR: Load Select Register 3 Nibble (12-bit)** It loads the 8-bit board Select Register, which is the output register for the board select signals Sel#8:1. Note that the least significant select SEL1 is a sequencer internal select signal.

LSR: "Select Value"

| Nib 00 | Nib 01 | Nib 02 |
|---|---|---|
| 0110b | X | Sel(6-0) |

### 4.3.15 Control Instructions

**4.3.15.1 CAL: Call Subroutine: 4 Nibble (16 bit)** This is the classic CALL instruction similar to 8085; the address pointing to the sub-routine is an operand, which for the current implementation is (12-bit). The next address is pushed to the address stack and control is transferred to the specified address.

CAL: "Subroutine Address"

| Nib 00 | Nib 01 | Nib 02 | Nib 03 |
|---|---|---|---|
| 1000b | 12-bit Subroutine Address. A(11-0) | | |

**4.3.15.2 RET: Return 1 Nibble (4-bit)** This is the classic RETURN instruction similar to 8085; all sub-routines called must end with a return instruction. On execution of RET, return address is popped from the top of the stack and control is transferred to that address.

RET:

| Nib 00 |
|---|
| 1001b |

**4.3.15.3 JCB: Jump if Control Bit Set. 5 Nibble (20-bit)** For conditional branching, a conditional jump instruction is provided.

Unlike all other instructions, the JCB includes two types of operands and control is transferred to the specified address if the corresponding control register (16-bit wide) flag is set.

JCB: "Bit Number"; "Address"

| Nib 00 | Nib 01 | Nib 02 | Nib 03 | Nib 04 |
|---|---|---|---|---|
| 1010b | BN(3-0) | 12-bit Address. A(11-0) | | |

**4.3.15.4 LRB: Loop Begin 2 Nibble (8-bit)** The next program counter address and the loop counter value pointed to by the loop register index in nibble 01 are pushed onto their respective stacks. The stack implementation permits the use of nested loops of degree equivalent to stack depth.

LRB: "Index"

| Nib 00 | Nib 01 |
|---|---|
| 1010b | 4-bit loop register index 0:15 |

**4.3.15.5 LPB: Loop Begin 1 Nibble (4-bit)** The next program counter address and the value of the pattern memory location pointed to by the pattern pointer are pushed onto their respective stacks. The stack implementation permits the use of nested loops of degree equivalent to stack depth.

LPB:

| Nib 00 |
|---|
| 1011b |

**4.3.15.6 LPE: Loop End 1 Nibble (4-bit)** The loop count on the top of the stack, if not zero, is decremented and address stored on the top of the program counter address stack is loaded into the address pointer. If the loop count is zero, the count and program counter address are popped off the stack and program counter address is incremented.

LPE:

| Nib 00 |
|---|
| 1100b |

### 4.3.16 Delay Instructions

With a combination of the following instructions, any arbitrary delay (less than a second) can be implemented. A corresponding down counter is loaded with the count specified by the operand. The delay values for every instruction are currently limited to 255*least count. This is based on the assumption that most frequently required delays will be within this range. However, if the delay values required are in the complete range of 0-999 *Least count, either the size of the operand or the least count can be scaled to provide the complete range accordingly.

**4.3.16.1 DMS: Delay Milliseconds 3 Nibble (12-bit)** A delay with resolution (least count) of a millisecond and of magnitude equivalent to the value specified by the operand "Delay mili seconds count" (8-bit) is inserted. None of the output registers are altered while this instruction is executed. The program counter is halted until the delay counter is reset.

DMS: "ms count"

| Nib 00 | Nib 01 | Nib 02 |
|---|---|---|
| 1101b | DmsC(7-0) | |

**4.3.16.2 DuS: Delay Microseconds 3 Nibble (12-bit)** A delay with resolution of a microsecond and of magnitude equivalent to counter value specified by the DusC Delay micro seconds count (8-bit) is inserted. None of the output registers are altered when this instruction is executed. The program counter is halted until the delay counter is reset.

DUS: "us count"

| Nib 00 | Nib 01 | Nib 02 |
|---|---|---|
| 1110b | DusC(7-0) | |

**4.3.16.3 DSC: Delay System Clock 3 Nibble (12-bit)** A delay with resolution of SEQCLK period and of magnitude equivalent to counter value specified by the DSCC Delay system clock count (8-bit) is inserted. None of the output registers are altered when this instruction is executed. The program counter is halted until the delay counter is reset.

DSC: "delay system clock count"

| Nib 00 | Nib 01 | Nib 02 |
|---|---|---|
| 1111b | DSCC(7-0) | |

**4.3.16.4 NOP: No Operation 1 Nibble (4-bit)** A delay with resolution of SYSCLK period and of unit magnitude is inserted. None of the output registers are altered when this instruction is executed.

NOP:

| Nib 00 |
|---|
| 0000b |

## 4.4 Appendix IV – Synthetic Pixel Generator

Revision 4.63 of the Pixel FPGA firmware (McbPixFpgaV463.mcs) contains a comprehensive pixel data generator that allows testing of the MONSOON data paths and software. The data generator produces synthetic pixel data in an amount and cadence that represents data produced by physical acquisition boards. This can be used to test and optimize the pixel data path without having to use the intrument or detectors. Also, the data generator can be programmed to inject synthetic data into an existing DHE data path to replace one or ore disabled ormissing acquisition baords. The pixel generator suppoorts both visible and infrared readout schemes. Synthetic pixel dta contains a channel number tag and an 8-bit sythetic data field. The data field can be programmed as a sequential ramp or as pseudo noise values. The generator is controlled by the attributes described in Table 19.

| Address | Function Name | Function Description |
|---|---|---|
| 0x02FC (15:13) | SIM_MODE | 3-bit field of the pixel control register. Controls the pixel generator operational mode. |
| 0x02D0 | SIM_PIXROWS | Number of pixel rows to synthesize. |
| 0x02D1 | SIM_PIXCOLS | Number of pixel columns to synthesize. |
| 0x02D2 | SIM_PIXTIME | Period between pixel bursts |
| 0x02D3 | SIM_CHANNELS | Number of pixels to generate each burst. |
| 0x02D4 | SIM_DELAY | Time to delay before initial burst is generated. |
| 0x02D5 | SIM_INTEGRATE | Integration time for IR mode 2. |
| 0x02D6 | SIM_FOWLER | Number of Fowler samples to take in IR mode 2. |
| 0x02D7 | SIM_PIPE | Pipeline priority time slots to generate in mode 3. |

Table 13: Pixel Generator Memory Map

### 4.4.1 Simulation Mode Register (SIM_MODE)

This is a 3-bit field in the pixel control register. The value in this register controls the moe o the pixel generator. Table 20 shows the availabel mode values.

| SIM_MODE | Function |
|---|---|
| 0 | Pixel generator disabled. No synthetic data generated. (default value). |
| 1 | Generate pixels in visible mode. (emulate CCD) - sequential data. |
| 2 | Generate pixels in IR mode - sequential data. |
| 3 | Generate replacement pixels on time slot - sequential data. |
| 4 | Pixel generator disabled. No synthetic data generated. |
| 5 | Generate pixels in visible mode (CCD) - noise data. |
| 6 | Generate pixels in IR mode - noise data |
| 7 | Generate replacement pixels on time slot - noise data. |

Table 14: Pixel Generator Mode Description

*Generate pixels in visible mode* – The pixel genrator will be triggered on the Start Exposure command sent to the MCB. There will be (SIM_PIXROWS x SIM_PIXCOLS) bursts of data separated by (SIM_PIXTIME x 100ns) time. For each burst, SIM_CHANNELS worth of pixels will be genreated. The data will contain an 8-bit channel ID field (repeated each burst)

and either an 8-bit incremental pixel value (SIM_MODE bit 2 = 0) or an 8-bit pseudo noise value. There will be a dealy of (SIM_DELAY x 1ms) before the first burst of pixels is sent to the PAN. Total number of pixels sent to the PAN will be (SIM_PIXROWS x SIM_PIXCOLS x SIM_CHANNELS).

*Generate pixels in IR mode* – The pixel generator will be triggered on the Start Exposure command sent to the MCB. There will be an initial delay of (SIM_ELAY x ims) before the data begins to arrive at the PAN. For the value of SIM_FOWLER, there will be (SIM_PIXROWS x SIM_PIXCOLS) bursts of data separated by (SIM_PIXTIME x 100ns) time sent to the PAN. Before each Fowler smaple, there will be a delay of (SIM_DELAY x 1ms). After the completion of these data frames, the integration timer will waitr (SIM_INTEGRATE x 1ms) before starting the second readout. The second readout will follow exactly the same process as pre-integration. Total number of pixels sent to the PAN will be (2 x SIM_PIXROWS x SIM_PIXCOLS x SIM_CHANNELS x SIM_FOWLER). The data will contain an 8-bit channel ID field (repeated each burst) and either an 8-bit incremental pixel value (SIM_MODE bit 2 = 0) or an 8-bit pseudo noise value.

*Generate replacement pixels on time slot* – this mode value is used to inject synthetic pixel data into the normal acquisition data stream generated by a set of MONSOON acquisition boards. In this mode, the 8-bit SIM_PIPE attribute is set up to indicate into which sequence time slots the data is to be placed. Table 21 shows the releveance of this register's bit positions.

| Pipeline Priority Value | Bit Position |
|---|---|
| 0 | Not available |
| 1 | Bit 1 – value 0b00000010 |
| 2 | Bit 2 – value 0b00000100 |
| 3 | Bit 3 – value 0b00001000 |
| 4 | Bit 4 – value 0b00010000 |
| 5 | Bit 5 – value 0b00100000 |
| 6 | Bit 6 – value 0b01000000 |

Table 15: Pixel Generator Pipeline Priority Register

In this mode it is essential that the pipeline prioity time slot being emulated by the pixel generator be preceded by either a real acquisition board pixel transfer time slot or a previous pixel generator time slot. This is because the pixel generator uses the edges of the pipeline prioity signal to "know" when the correct time slot is available so it can begin the data transfer. Examples follow.

Example: Acquisition boards in DHE slots 3, 4, 5 and 6 – The acquisition board in slot 5 has been disabled )pipeline enable = 0). Pipeline priority values are programmed as 0, 2, 4 and 6 on the respective boards. Setting the SIM_PIPE register to 3 will produce the reuqired result since the pixel generator data will be triggered at the end of the data transfer from the acquisition board in slot 4 at pipeline priority time slot 2. Setting the SIM_PIPE register to 4 will not produce data since there will be no end of transfer activity on priority 3 to trigger the pixel generator.

For simlar reasons pertaining to the above, it is not possible to command the pixel generator to initiate pipeline transfers with a pipeline priority value of 0.

In this mode, for each pipeline priority bit set in the SIM_PIPE register, the number of pixels gerneated will be equal to the value of SIM_CHANNELS.

### 4.4.2 Simulation Pixel Rows Register (SIM_PIXROWS)

This is a 16-bit register that controls the number of synthetic pixel data that is generated. The product of SIM_PIXROWS x SIM_PIX COLS is equal to the number of bursts that will be generated for each data frame. This register is not used in simulation modes 3 or 7.

### 4.4.3 Simulation Pixel Columns Register (SIM_PIX COLS)

This is a 16-bit register that controls the number of synthetic pixel data that is generated. The product of SIM_PIXROWS x SIM_PIX COLS is equal to the number of bursts that will be generated for each data frame. This register is not used in simulation modes 3 or 7.

### 4.4.4 Simulation Pixel Time Register (SIM_PIXTIME)

This 8-bit register controls the time between pixel bursts in units of 100ns. For example, to emulate a pixel data rate of 250Kpix/sec, set this register to a value of 40. This register is not used in simulation modes 3 or 7.

### 4.4.5 Simulation Channels Register (SIM_CHANNELS)

This 8-bit register controls the number of synthetic pixels generated for each burst. For example, to simulate four CCD acquisition boards, each with eight channels, set this register to 32. This register is used in all active simulation modes.

### 4.4.6 Simulation Delay Register (SIM_DELAY)

This 8-bit register controls two dealys. In modes 1, 2, 5 and 6, it allows specifying an initial delay from the arrival of the Start Exposure command to when the first pixel burst is generated and sent to the PAN computer.This delay is specified in this register in lilliseconds. In mode 2, the register also controls the timebetween Fowler sample data frames. This register is not used in simulation modes 3 or 7.

### 4.4.7 Simulation Integration Register (SIM_INTEGRATE)

In mode 2 and 6 (IR mode), this 16-bit register sets the simulated integration time between the pre-integration readout and post-integration readout data streams. Set this register to the required integartion time in milliseconds. This register is not used in modes 1, 3, 5 or 7.

### 4.4.8 Simulation Fowler Register (SIM_FOWLER)

In simulation modes 2 and 6, this 8-bit register determines the number of data frames sent ot the PAN in the pre-integration and post-integration simulation. Each data frame wil transmit (SIM_PIXROWS x SIM_PIXCOLS x SIM_CHANNELS) pixels to the PAN. This register is not used in modes 1, 3, 5 or 7.

### 4.4.9 Simulation Pipe Register (SIM_PIPE)

This 8-bit register is only used in modes 3 and 7. It specifies the pipeline prioity time slots to be used by the pixel generator to send data to the PAN. The significance of this register is hsown in Table 21.

### 4.4.10 Limitations of the Pixel Generator

There are several combinatins of mode and operating configuration that cannot be simulated by the pixel generator. These combinatins are listed by their respective SIM_MODE values.

- Modes 1 and 5 - SIM_CHANNEL = 1 and (SIM_PIXROWS x SIM_PIXCOLS) = odd number.
- Modes 2 and 6 - SIM_CHANNEL = 1 and (SIM_PIXROWS x SIM_PIXCOLS) = odd number.
- Modes 3 and 7 - SIM_CHANNEL = 1; Set the SIM_CHANNEL value to 2. It will operate correctly.

### 4.4.11 Sample Configuration File Setup

The following ten lines indicate a usable configuration file structure to include in the PAN.cxv file.

```
  SimPipe,SIM_PIPE,0x00102D7,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1.0,0.0,LINEAR,0.0,128.0,Timeslot,PIP
RPIORITY TIMESLOT MASK. I = GENERATE DATA, 0 = DONT GENERATE
  SimIntegrate,SIM_INTEGRATE,0x00102D6,1,0x0E000000,SIMPLE,SIMPLE,FLOATUINT,1.0,0.0,LINEAR,0.0,65535.0,mi
INTEGRATION TIME IN MILLISECONDS
  SimDelay,SIM_DELAY,0x00102D4,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1.0,0.0,LINEAR,0.0,255.0,millisec,I
DELAY BEFORE FIRST PIXEL BURST
  SimTime,SIM_TIME,0x00102D2,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1.0,0.0,LINEAR,10.0,255.0,Time,PERIOD
BETWEEN PIXEL BURSTS IN UNITS 100ns
  SimFowler,SIM_FOWLER,0x00102D5,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1.0,0.0,LINEAR,0.0,255.0,Samples,
OF FOWLER SAMPLES PER READOUT IN IR MODE
  SimChannels,SIM_CHANNELS,0x00102D3,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1.0,0.0,LINEAR,0.0,255.0,Pixe
OF PIXELS TO GENERATE FOR EACH BURST
  SimCols,SIM_COLS,0x00102D5,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UINT,1.0,0.0,LINEAR,0.0,65535.0,Pixels,NUMB
OF SIMULATED PIXEL COLUMNS
  SimRows,SIM_ROWS,0x00102D0,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UINT,1.0,0.0,LINEAR,0.0,65535.0,Pixels,NUMB
OF SIMULATED PIXEL ROWS
  SimData,SIM_DATA,0x00102FC,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,32768.0,0.0,LINEAR,0.0,1.0,Value,0
= INCREMENTAL DATA, 1 = NOISE
  SimMode,SIM_MODE,0x00102FC,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,8192.0,0.0,LINEAR,0.0,3.0,Value,0
= DISABLED,1 = CCD MODE,2 = IR MODE,3 = TIMESLOT REPLACEMENT MODE
```

## 4.5 Appendix V – csv file for MCB

```
## PAN Attribute Name,DHE Func Name,Base Address,Number of elements,Creg,Set Method,Read Method,PANType,DHEType,
     Coef1,Coef2,Function ID,Min Value,Max Value,UnitName,HelpText,,
intTime,SEQ_ITR,0x0010000,1,0x02000000,SIMPLE,SIMPLE,FLOAT,UINT,1000,0,LINEAR,0,4294967.295,Second,Set this
     register to the desired total integration time of an exposure. Timing resolution is 1 millisecond,,
mcbBkplnSlct,MCB_BCKPLN_SLCT,0x0010100,1,0x05000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,32768,0,LINEAR,0,1,Boolean,Set
     this function to 1 when using MCB in an 8 slot Schroff Backplane,,
mcbClkEnables,MCB_CLKENABLES,0x0010100,1,0x05000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,255,Value,Writing bits
     to this register controls the global clock sourcing of SYSCLK to peripheral boards,,
clkClkEnbl,MCB_CLK_ENABLE_3,0x0010100,1,0x05000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4,0,LINEAR,0,1,Boolean,Writing a
     one to this function enables the SYSCLK to board 3,,
acqClkEnbl,MCB_CLK_ENABLE_6,0x0010100,1,0x05000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,32,0,LINEAR,0,1,Boolean,Writing a
      one to this function enables the SYSCLK to board 6,,
actIntTime,SEQ_ITC,0x0010101,1,0x02000000,NOMETHOD,SIMPLE,FLOAT,UINT,1000,0,LINEAR,0,4294967296,Second,Reading
     this register returns the agregate integration time. Timing resolution is 1 millisecond.,,
mpuStrtVctr,SEQ_CMD_STARTVCTR,0x0010102,1,0x06000000,NOMETHOD,RDMSKWRT,FLOAT,UINT,16,0,LINEAR,0,255,Boolean,
     Reading this function returns the start exposure vector in the sequencer control register,,
seqCmds,SEQ_CMDS,0x0010102,1,0x06000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Reading this register
      returns the global data in the sequencer control register,,
Usr1Flag,SEQ_CMD_USER1,0x0010102,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,8192,0,LINEAR,0,1,Boolean,This
     function sets or reads the sequencer user bit 1 in the sequencer control register,,
contRun,SEQ_CMD_USER0,0x0010102,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4096,0,LINEAR,0,1,Boolean,This function
      sets or reads the sequencer user bit 0 in the sequencer control register,,
Usr2Flag,SEQ_CMD_USER2,0x0010102,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,16384,0,LINEAR,0,1,Boolean,This
     function sets or reads the sequencer user bit 2 in the sequencer control register,,
Usr3Flag,SEQ_CMD_USER3,0x0010102,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,32768,0,LINEAR,0,1,Boolean,This
     function sets or reads the sequencer user bit 3 in the sequencer control register,,
```

```
loopregs,SEQ_LOOPREGS,0x0010110,16,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write all
     loop registers,,
loopreg0,SEQ_LOOPREG[0],0x0010110,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 0,,
loopreg1,SEQ_LOOPREG[1],0x0010111,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 1,,
loopreg2,SEQ_LOOPREG[2],0x0010112,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 2,,
loopreg3,SEQ_LOOPREG[3],0x0010113,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 3,,
loopreg4,SEQ_LOOPREG[4],0x0010114,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 4,,
loopreg5,SEQ_LOOPREG[5],0x0010115,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 5,,
loopreg6,SEQ_LOOPREG[6],0x0010116,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 6,,
loopreg7,SEQ_LOOPREG[7],0x0010117,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 7,,
loopreg8,SEQ_LOOPREG[8],0x0010118,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 8,,
loopreg9,SEQ_LOOPREG[9],0x0010119,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write to
     loop register 9,,
loopreg10,SEQ_LOOPREG[10],0x001011A,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 10,,
loopreg11,SEQ_LOOPREG[11],0x001011B,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 11,,
loopreg12,SEQ_LOOPREG[12],0x001011C,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 12,,
loopreg13,SEQ_LOOPREG[13],0x001011D,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 13,,
loopreg14,SEQ_LOOPREG[14],0x001011E,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 14,,
loopreg15,SEQ_LOOPREG[15],0x001011F,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read / write
     to loop register 15,,
mcbClkDelay,MCB_CLKCTRL,0x0010281,1,0x05000000,SIMPLE,SIMPLE,FLOAT,UCHAR,1,0,LINEAR,0,255,Value,Control register
     for Sysclk Control delay. Delay applies is 19.5ns + Value * 0.5ns,,
mcbFpdpWrt,MCB_PIX_LED_1,0x0010282,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,This function
      provides control of the pixel indicator led source 1 -Flash each time a data word (pixel or message) is
     written to the Slink FPDP port,,
mcbFpdpBusy,MCB_PIX_LED_2,0x0010282,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,2,0,LINEAR,0,1,Boolean,This
     function provides control of the pixel indicator led source 2 - Flash each time the Pixel FPGA is busy
     decoding a PAN CMD,,
mcbPixDataRdy,MCB_PIX_LED_3,0x0010282,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4,0,LINEAR,0,1,Boolean,This
     function provides control of the pixel indicator led source 3 - Flash each time a pixel data value is
     written to the Slink FPDP port,,
slinkStats, SLINK_FULLSTATS, 0x0010300,22, 0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,values,List
     all register related to slink stats
slinkRcvdDataWL,SLINK_RCVDDATWL,0x0010300,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many data words has been received (16 LSB),,
slinkRcvdDataWH,SLINK_RcvdDatWH,0x0010301,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many data words has been received (16 MSB),,
slinkRcvdCtrlWL,SLINK_RcvdCtlWL,0x0010302,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many control words has been received (16 LSB),,
slinkRcvdCtrlWH,SLINK_RcvdCtlWH,0x0010303,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many control words has been received (16 MSB),,
slinkXferDataWL,SLINK_XFERDatWL,0x0010304,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many data words has been sent (16 LSB),,
slinkXferDataWH,SLINK_XFERDatWH,0x0010305,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many data words has been sent (16 MSB),,
slinkXferCtrlWL,SLINK_XFERCtlWL,0x0010306,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many control words has been sent (16 LSB),,
slinkXferCtrlWH,SLINK_XFERCtlWH,0x0010307,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How
     many control words has been sent (16 MSB),,
slinkXferDataPL,SLINK_XFERDataPL,0x0010308,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How
     many pages of data words has been sent (16 LSB),,
slinkXferDataPH,SLINK_XFERDataPH,0x0010309,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How
     many pages of data words has been sent (16 MSB),,
slinkXferCtrlPL,SLINK_XFERCtrlPL,0x001030A,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How
     many pages of control words has been sent (16 LSB),,
slinkXferCtrlPH,SLINK_XFERCtrlPH,0x001030B,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How
     many pages of control words has been sent (16 MSB),,
slinkXferDCPL,SLINK_XFERDCPL,0x001030C,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How many
     complete pages of data words has been sent (16 LSB),,
slinkXferDCPH,SLINK_XFERDCPH,0x001030D,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How many
     complete pages of data words has been sent (16 MSB),,
slinkXferCCPL,SLINK_XFERCCPL,0x001030E,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How many
     complete pages of control words has been sent (16 LSB),,
slinkXferCCPH,SLINK_XFERCCPH,0x001030F,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,How many
     complete pages of control words has been sent (16 MSB),,
slinkFullFlagErr,SLINK_FFERR,0x0010310,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,Contains
     the number of times FullFlag has been set to true.,,
slinkLinkDownErr,SLINK_LDERR,0x0010311,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,pages,Contains
     the number of times LDOWN has been set to true.,,
slinkUpTimeMin,SLINK_UTM,0x0010312,1,0x0E000000, NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,min,System uptime
     in minutes,,
slinkUpTimeSec,SLINK_UTS,0x0010313,1,0x0E000000, NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,sec,Seconds since
     last change in slinkUptimeMin,,
slinkPLI,SLINK_PLI,0x0010314,1,0x0E000000, NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,words,How many words
     contains each complete Page,,
slinkMWI,SLINK_MWI,0x0010315,1,0x0E000000, NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,clocks,How many clocks
     does Slink wait before closing an incomplete page,,
mcbPCBVer,MCB_PCB_VER,0x0010600,1,0x0C000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Hardware
     revision of the MCB board layout,,
mcbSyncDelay,MCB_SYNC_DELAY,0x0010601,1,0x0E000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,255,nsec,Used to set/
     read delay applied to sync. Delay = 19.5ns + value * 0.5ns,,
reloadMcb,MCB_RELOAD,0x0010666,1,0x0E000000,SIMPLE,NOMETHOD,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,Reload MCB fpga
     firmware from eprom,,
mcbSeqPatMem,SEQ_PAT_MEM,0x0011000,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Sequencer
     pattern memory space (Note actual mem size is 4096 locations),,
```

```
mcbSeqPgmMem,SEQ_PGM_MEM,0x0014000,1,0x06000000,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Sequencer
     program memory space (Note actual mem size is 1024 locations),,
mcbSwitchAuto,MCB_SWITCHAUTO,0x001FFF8,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,1,0,LINEAR,0,1,BOOLEAN,If this
     bit is set to '1' switches for sync an clock are autogenerated according to master slave setting (pan att
     get dheMstr),,
mcbLocalSysClk,MCB_SYSCLKSWITCH,0x001FFF8,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,2,0,LINEAR,0,1,BOOLEAN,When
     mcbSwitchAuto is unset we can set to use local ('1') or external ('0') Sysclock signal. External means that
     it will use a clock coming from the synchronization interface.,,
mcbLocalSync,MCB_SYNCSWITCH,0x001FFF8,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4,0,LINEAR,0,1,BOOLEAN,When
     mcbSwitchAuto is unset we can set to use local ('1') or external ('0') Sync signal. External means that it
     will use a sync coming from the synchronization interface.,,
mcbCurrSysClk,MCB_CURRSYSCLKSWITCH,0x001FFF8,1,0x0E000000,NOMETHOD,RDMSKWRT,FLOAT,UINT,8,0,LINEAR,0,1,BOOLEAN,
     Current value of the SysClk Selector. ('1') -> local; ('0') -> external,,
mcbCurrSync,MCB_CURRSYNCSWITCH,0x001FFF8,1,0x0E000000,NOMETHOD,RDMSKWRT,FLOAT,UINT,16,0,LINEAR,0,1,BOOLEAN,
     Current value of the Sync Selector.('1') -> local ('0') -> external,,
mcbFpdpRcvDataValid,MCB_SEQ_LED_1,0x001FFF9,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,This
      function provides control of the MCB status indicator led source 1 - Flash each time a data word is
     received from the PAN,,
mcbPanBusReq,MCB_SEQ_LED_2,0x001FFF9,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,2,0,LINEAR,0,1,Boolean,This
     function provides control of the MCB status indicator led source 2 - Flash each time the PAN command decode
     logic requests use of the sequencer bus,,
mcbMpuSeqBusReq,MCB_SEQ_LED_3,0x001FFF9,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4,0,LINEAR,0,1,Boolean,This
     function provides control of the MCB status indicator led source 3 - Flash each time the MPU Sequencer logic
      requests use of the sequencer bus,,
serialNumMcb,MCB_SERNUM,0x001FFFA,1,0x0C000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,4294967295,Value,Read the
     silicon serial number of the board,,
temperatureMcb,MCB_TEMPERATURE,0x001FFFB,1,0x0C000000,NOMETHOD,SIMPLE,FLOAT,TWLVBIT,10,0,LINEAR,-128,128,Deg.C,
     Read the actual temperature of the board,,
controlMcb,MCB_CONTROL,0x001FFFC,1,0x0E000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,1,0,LINEAR,0,65535,Boolean,Read /
     write the global control register for the board,,
mstrSyncDly,SEQ_SYNCDELAY,0x001FFFC,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,256,0,LINEAR,0,255,Value,Sets the
     delay time after receiving a start exposure command issuing a master sync pulse between 0 and 256
     milliseconds,,
dheMstr,DHE_MASTER,0x001FFFC,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,16,0,LINEAR,0,1,Boolean,Writing a one to
     this function establishes the DHE as Master.,,
mpuSeqClkDiv,SEQ_CLKDIV,0x001FFFC,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,4,0,LINEAR,0,3,Value,Read / write to
     the sequencer MPU master clock divider,,
mpuIntPause,SEQ_INTPAUSE,0x001FFFC,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,2,0,LINEAR,0,1,Boolean,Writing a one
      to this function pauses the integration timer writing a zero allows the timer to continue,,
mcbSeqEnable,SEQ_ENABLE,0x001FFFC,1,0x06000000,RDMSKWRT,RDMSKWRT,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,Writing a one
     to this function enables the sequencer MPU to run.,,
statusMcb,MCB_STATUS,0x001FFFD,1,0x0E000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Boolean,Read the status
      register for the board,,
resetMcb,MCB_RESET,0x001FFFE,1,0x0E000000,SIMPLE,NOMETHOD,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,Soft reset the board
     firmware,,
boardIdentityMcb,MCB_IDENT,0x001FFFE,1,0x0C000000,NOMETHOD,SIMPLE,FLOAT,UINT,1,0,LINEAR,0,65535,Value,Read the
     board identity register,,
firmwareVerMcb,MCB_FIRMVERS,0x001FFFF,1,0x0C000000,NOMETHOD,SIMPLE,FLOAT,UINT,100,0,LINEAR,0,655.35,Value,Read
     the board firmware version register,,
rebootMcb,MCB_REBOOT,0x001FFFF,1,0x0E000000,SIMPLE,NOMETHOD,FLOAT,UINT,1,0,LINEAR,0,1,Boolean,Reload MCB fpga
     firmware from eprom,,
```

## 4.6 Appendix V – Firmware History

```
--------------------------------------------------------------------------------
2010.08.09 - Last changes before holydays
--------------------------------------------------------------------------------
- I've added a copy of _acq12 fpa to adapt csv files to our new HW/FW into this package
- Corrected errors in documentation
- Decreased images size in documentation (pdf size from ~7MB down to 1.5MB)
- Added attribute on csv file to see all slink stats registers at same time
- Added attribute on csv file to see all loop registers at same time
- Added csv file into documentation

FWVersion 5.13 (r2063):
-----------------
- Changed slink statistics. Instead of contain kibiwords now it shows words sent or received.

--------------------------------------------------------------------------------
2010.03.18 - Last steps of production Testbench
--------------------------------------------------------------------------------
FWVersion 5.12 (r1100):
-----------------
- Removed PULLDOWN modifier from resetInHw pin on mcb.ucf (constrains file). I suspect that this pullup doesn't
     allow the push button to put resetInHw signal to a High state. I've checked it works.

--------------------------------------------------------------------------------
2010.01.08 - Starting to document all this stuff
--------------------------------------------------------------------------------
- Copying Peter Moore pdf documentation to Lyx, so I can start documentation of new MCB based on his docs

--------------------------------------------------------------------------------
2010.01.07 - Welcome back to work
--------------------------------------------------------------------------------
FWVersion 5.11 (r1100):
-----------------
- On the PCB, reset push button is active high, on firmware it was active low. I Change some names (resetInHw_n
     to resetInHw) and polarity.

- Changed time between temperature sensor reads from 1ms to 500ms.
```

```
--------------------------------------------------------------------------------
2009.12.28 - kill all the useless pins!
--------------------------------------------------------------------------------
FWVersion 5.10 (r1093):
----------------
        - Led_Fifo_Full (the red one in the led tower) shows isMaster signal (just for debug)
                - added port led_fifofull at mcbFull.vhd
                - added led_fifofull at pin E14 in mcb.ucf
                - recreated isMaster signal (I deleted it some time ago)

        - Removed internal fpga pullup from sysclk pin


FWVersion 5.09 (r1092):
----------------
        - sync_in and sync_out comes back to normality.
        - Removed seq_data_ack port from pixdatamux (I've never seen it connected)
        - I've checked all warnings from synthesis report. Most of them are bus ports which comes from an upper
             level and not all its bits are used. This ones doesn't worry me.
                - on topSeq, mpuEvents is not connected. It was connected to unused signals (check log from
                     2009.12.22). topSeq.mpuEvents comes from topSeq/mpu_seq/Event_register module which only
                     generates this register. I delete Event_register.
                - I've deleted some unused signals (declared, assigned but not readed)


FWVersion 5.08 (r1090):
----------------
        - Solved HW problems, I start to use normal clock instead of backup clock. Changed xilinx/mcb.ucf
        - To debug int_sync and ext_sync paths I temporarily change how sync_out port is generated. Also
             invalidated sync_in.
        - FW should be reloaded when a write is done to 0xffff (ADDR_FWVERDEC = ADDR_RELOADFW) or to 0x666 (
             ADDR_REPROGRAM)


FWVersion 5.07:
----------------
        As explained somewhere, in new mcb boards (either HW1.1 or HW1.2) there is no eprom serial memory as in
             the NOAO boards. For this reason sda and scl ports can be deleted as they don't interface anything,
             and serConfigIfc module can be deleted.
        If some day somebody needs non volatile memory to store something, there is a serial number chip with a
             1-wire protocol which has non volatile memory. I add to my todo list implement something like what
             we had here but using this small memory.
        - topPixFpga:
                - removed sda and scl:
                        - removed ports
                        - removed ser config interface signals
                        - removed serConfigIfcV42 instance
                - removed components IBUF and OBUF
                - removed prevously commented code
                - removed all reset aliases
                - removed all sysclk aliases
        - deleted serConfigIfcV42.vhd



--------------------------------------------------------------------------------
2009.12.28 - The joke is to be working without net access (spanish April's fool day)
--------------------------------------------------------------------------------
FWVersion 5.06:
----------------
        - Deleted some previously comented code
        - Created externSelec module. This module is used to select if we want to use external clk and sync or
             use the ones on the board. By default if the board is set to master it uses internal sync and clock,
              if the board is set as slave it uses external signals. If bit 0 of this register is set to 0 then
             clock switch is configured setting bit 1 of switchSelec and switch sync is set by bit 2. When read,
             bit 4 tells us the current value of clock selector, and bit 4 current value of sync selector.
        - Yuhuu! We've got net again.
        - Added constant ADDR_SWITCHES 0xfff8 to write and read switchSelec register.
        - Connected led_activity signal to led_fib

--------------------------------------------------------------------------------
2009.12.23 - Coding instead of go shopping
--------------------------------------------------------------------------------
-Change syncDelay component name to delayChipCtrl as it is going to be used to control both sync and clock delay
     chips. Also renamed file.
-Created delayChips component
-Instantiated components in sequencer
-Modified ports in topSeq and mcbFull
-Modified pins in mcb.ucf
-Changed tp_xx pins in mcbFull top to testPoints(7:0)
-I've checked all this works. The only thing that doesn't work is read register 0x601 and register 0x281 (the
     ones of the delay chips) But when one writes to this registers, delay chips are programmed and changes its
     delay.
-Registers 0x601 and 0x281 output ports were unconnected so I guess that was the problem when reading this
     registers

--------------------------------------------------------------------------------
2009.12.22 - Merry Christmas
--------------------------------------------------------------------------------
-Change seq/serno_interface.vhd/sn_data output port from 48 bit wide to 32. This made me change ports on
     seqPackage, topSeq and PersonalityModule. sn_data is a register which is read through slink and it only
     reads the 32LSB.

-Removed ports expose_start, frame_start and temp_rd_stb from personality module. They were unconnected.

-seq/personality_ModuleV45.vhd:
        -event_reg was used to generate Expose_start and frame start (both of them were deleted because they were
```

```
              unconnected)
        -event_reg is used only to generate event_hist_reg
        -event_hist_reg is used only to generate itself
        Then I can delete event_reg and event_hist_reg (I've checked in mcbFull.nrg and this signals are
             simplified on synthesis)
        -MPU_EVENTS is an input to Personality_Module and is used only in event_hist_reg (which I'm going to
             delete) so I can delete also this port.
        -Removed event_cs, ADDR_EVENTREG, event_reg, event_hist_reg,

Changes specified above had been tested in a real system and nothing is broken (this is rev 1076 in IES svn)

-At constantsPkg changed constant name from ADDR_CLKCTRLDEC to ADDR_CLKPHASE
-Changed Personality_Module to implement clkPhase Register in topSeq
-Right now I think that it doesn't compile (I'm pretty sure), but I want to go home. Tomorrow I will finish this
     and some other things I have on the queue.


--------------------------------------------------------------------------------
2009.12.18 - Refactoring is my second name (part III)
--------------------------------------------------------------------------------
These days I've been working in other things and checking that all this changes were correct:
        -Propagated all changes due to errors on the code from revision 984 to revision 1022.
        -Revision 1053 has been checked and it works.
        -Tests done are:
                -Exposure at 1Kx4K
                -ACQ DAC tests (noise test 6)
                -ACQ12 All registers test

I continue with changes:
-src/seq/clockenableregister.vhd:
        -Changed  ports clk2,..clk8 to bp_clk(8 :2)
        -Changed some registers to sync reset to async.
        -Changed clockenableregister component in packageseq and topseq
-src/seq/topSeq:
        -Changed some signal names (xlxn_70 -> clkDataOut,..)
        -Removed some thing.
-src/seq/Pan_comm...:
        -Generate seqWrite_n (add this port)
        -change output port pixfifo_wrt_n to pixfifo_wrt (not only the name, also its polarity)
        -LocSel register was inside a big process and the ojly things it was doing was delay intboardsel(0) one
             cycle, now it has its own process.

-Generation of cmdRdy and SeqDataReq of the old fpga interface are now generated inside the personality module
     instead of being generated on the topseq module
-src/seq/topSeqFpga.vhd:
        -XLXN_9                 -> seq_cmd_sel
        -XLXN_10                -> seq_cmd_reg
        -XLXN_11                -> wr_int_reg
        -XLXN_12                -> seq_int_reg
        -XLXN_14                -> seq_lpp_reg
        -XLXN_15                -> seq_pat_sel
        -XLXN_16                -> seq_pat_mem
        -XLXN_18                -> seq_pgm_mem
        -XLXN_19                -> status_reg
        -XLXN_439               -> led_data
        -XLXN_438               -> led_reg_sel
        -XLXN_43                -> seq_addrdata_in
        -XLXN_214               -> seq_int_ctr
        -XLXN_298               -> echoData
        -XLXN_450               -> tst_ctr_enable
        -XLXN_465               -> tempData
        -XLXN_466               -> mpuEvents
        -XLXN_464 deleted (temp_rd_stb not connected)



--------------------------------------------------------------------------------
2009.12.10 - Refactoring is my second name (part II)
--------------------------------------------------------------------------------
-Refactoring PanCommandDec:
        -intAddrData
        -added new state: waitAsync. When the state machine boots it does nothing until it receives an async
             command. Once it has received this async command it starts to process comands. Currently when it
             boots it goes to state idle, and there we can find a lot of concatenated "if". With this new state I
              avoid using this "if"s.
        -Changed cmd_decode to a typical async reset register
-Modified mcb.ucf (some errors on pin association)
-Modified mcbFull.vhd to use reset button instead of test point

--------------------------------------------------------------------------------
2009.12.09 - Refactoring is my second name
--------------------------------------------------------------------------------
-FPGA_CTL_6 is an alias of START_EXP in top_seq_fpga. In this last MCB layout it is not connected (pin left
     floating), and probably in MCB HW1.1 this also happens. I delete port FPGA_CTL_6 from top_seq_fpga and
     mcbFull, and also from ucf file.
-FPGA_CTL_2 is an alias of START_EXP in top_seq_fpga. It is connected to top_pix, and inside top_pix it only
     feeds sim_trigger port of pix_fifo. I change its name to simTrigger as I find it more easy to know what it
     is.
-FPGA_CTL_1 is an output port from topSeq. Change its name to seqWrt_n

-Refactoring PanCommandDecode:
        -locWrt
        -intBoardSelect
        -PixFifo_Wrt_n
        -start_exp_s (added in comments recomendation to try once checked everything works this way)

--------------------------------------------------------------------------------
2009.12.04 - Here we go, again!
--------------------------------------------------------------------------------
```

```
-Modified Makefile and genProject
-deleted mcb.prj from subversion (it is automatically generated)
-created folder logs inside xilinx to store logs from make (added to subversion folder and files)

-Learned what is this Asyn stuff. When the fpga boots, sequencer remains in a halt state until it receives an
     asyn command (asyn commands = cmd(31:30) are "11"). If it receives another type of command Pan Command
     decode discards this commands until it receives one Asyn command. I don't know why it does this (probably
     something related to the implementation with 2 fpgas), but at least I'm going to clean some things. Right
     now there is a component called resetSync which does nothing except notify pancommanddec and ledcontroler if
      this asyn command has been received, but is pancommanddec who tells resetsync if an asyn command has been
     received. So I will integrate resetSync into Pan command dec.
        -Change PanCommandDec port Async_Pending from input to output (its the reset of LedController)
        -Delete PanCommandDec output port Sync_Flag (Thats a big mess, sync signal to state an Async has been
             received)
        -Removed resetSync component from topSeq
        -Changed connections between PanCommandDec and LedLatch

-PanCommandDec:
        -On state machine if deleted an if condition as it can't happen (simplifying the state machine). When we
             are in state idle, echoenable is set to '1', if in state idle one rd command is received, there was
             one option to go to state rdechowait if echoenable was set to '1' and another option, which of
             course is impossible that it happens.
(Changes only affect implementation, at the end signals should be the same, just to save mental illnes of fw
     programmer)
        -Changed how tmp_DATA_IN_RD is generated (to simplify state machine)
        -Changed how echoEnable is generated
        -Changed how echo_data is generated
        -Changed implementation of SEQ_BUS_REQ

--------------------------------------------------------------------------------
2009.12.03 - The Show must go on!
--------------------------------------------------------------------------------
-Long time ago there was problems on backplane reads. We found that the problem was that the fpga_ctl_1 signal
     between sequencer and pixel modules was too fast so we have to delay it one cycle (in Peter Moore version of
      MCB there was an intrinsic delay due to the fact of the time to transfer the signal between fpgas). I don't
      know why I put this delay (a simple register) in top mcbFull, I put it inside the sequencer module.

-Inside Pan_Command_Decode I've found two new constants, HARD_RST_ADDR and SOFT_RST_ADDR:
        -Changed its names to ADDR_RELOADFW and ADDR_SOFTRESET.
        -Moved definition of this constants to constants.vhd
        -updated constants excel file (Not new addresses, but new meaning for existing ones)
-Pan_Command_Decode.vhd:
        - Changed "MODE" port from inout to out
        - Changed DEVADDR port from inout to out
        - Changed ADDRDATA port from inout to out
        - Changed BOARD_SELECT port from inout to out

-Sequencer_Bus_Mux.vhd:
        -deleted previously commented code

-Pan_Command_decode.vhd:
        -Deleted ports HARD_RST and SOFT_RST. Right now they were disconected (I was unaware of this ports and I'
             ve implemented more or less the same functionality in Personality Module. I've added on my todo list
              to add to Personality Module that reloadFW must trigger also with this addresses.
        - Deleted all logic that previously implemented HARD_RST and SOFT_RST and also a documentation comment (
             WRITES TO ADDRESS 0xFFFF (NOMINAL BOARD CODE REVISION IDENT) ARE CONSIDERED TO BE A HARD RESET
             REQUEST. IN SIMILAR FASHION, WRITES TO 0xFFFE (BOARD FUNCTION IDENTITY) ARE DECODED TO PROVIDE A
             SOFT RESET TO THE ADDRESSED BOARD.)
-Top: changed bp_spare ports from inout to input.
-Modified genProject script to save some backup files into a tmp folder inside xilinx folder


--------------------------------------------------------------------------------
2009.12.02 - Ara resulta que no és urgent
--------------------------------------------------------------------------------
-Updated all known registers into excel file docs/registers.xls. In comments inside src/pix/pixCmdDecodeV45.vhd
     there are more registers explained, but I cannot find inside the code.
-I've caught my best axe (aka emacs), let's start the bleeding!!! delet'em all!!
-Changes at pix/topPixFpga.vhd:
        -Deleted BUFT component and instances
        -Deleted clkxfy ports on topPixFpga component
        -Deleted all signal definitions of CLK_CTRL_DEC_CLKxDy
        -Deleted ClkCntrlDecodeV40_1 instance
        -Deleted:
                -CLK_CTRL_DEC_SYSCLK (silly signal)
                -CLK_CTRL_DEC_CLK_CS (feed by PIX_CMD_DEC_CLK_CMD_STB)
                -CLK_CTRL_DEC_DATA_IN (feed by PIX_CMD_DEC_DATA_OUT(7:0))
-Changes at pix/PixCmdDecodeV45.vhd:
        -Deleted CLK_CMD_STB output port
        -deleted ClkCtl_cs select signal definition and implementation
        -deleted previously commented code sections
        -delected comments about address ranges as they were outdated
-Changes at topPixFpga:
        -Deleted port CLK_CMD_STB of instance PixCmdDecodeV461 (also deleted from components in packagePixel)
-Changes in pix/regMux.vhd:
        -Removed port clkcntrldecdatout and option to output this to dataOut

-Now clkControl is extirped, let's implant a new version for the new chip.
-Changes in seq/topSeqFpga.vhd:
        -Delete all code commented sections
        -Deleted attributes BOX_TYPE, BUFR_DIVIDE, IOSTANDARD, CAPACITANCE, SLEW and DRIVE. Not used.
        -created packageSequencer.vhd, which contains all component declarations
-PersonalityModule:
        -Instead of output the direct read of the temperature sensor , now we output two's complement of the
             temperature value.
-Deleted code commented out in lrl_decoder.
-removed pattern_mem.vhd and progmemcore.vhd from seq folder to be sure that the ones used are the ones inside
     cores folder
```

```
--------------------------------------------------------------------------------
2009.12.01 - I need to test new boards RIGHT NOW!
--------------------------------------------------------------------------------
-On MCBHw1.2 scl and sda lines are not connected, I've commented out this ports from top.
-Bug corrected from last changes in Personality_Module (There was one decode_vector left)
-topSeqFpga.vhd add new port isMaster (selfDescriptive)
-topSeqFpga.vhd deleted component OBUFT and attributes (there isn't instances of this component)
-topSeqFpga.vhd deleted component BUFR and attributes (there isn't instances of this component)
-topSeqFpga.vhd deleted component BUFG and attributes (there isn't instances of this component)
-topSeqFpga.vhd commented out component BUF and attributes. This is a simple buffer, change this to code.
-Added port to top, selClock. This is a selector for the clock ('1' -> external clock, '0' -> local clock). Added
      pin on constrains (B11).
-Added port to top, selSync. This is a selector for the clock ('1' -> external, '0' -> local). Added pin on
     constrains (A11).
-Commented out clkxfx ports on top.
-Changed default value for SEQCTLREG to set MCB as a Master DHE at boot.
-Added led_activity port on top. Added pin to constrains (D13)
-Added led_link_up port on top. Added pin to constrains (E7)
-led_link_up implemented
-led_activity right now is on when MCB is set as Master (used to debug clock and sync selectors)

-change the firmware version number to adjust to what Peter Moore was using. His last version was 4.61. I started
      from scratch (1.0) and using another register. I'm going to use same register as Peter FW and what I was
     calling version 1.04 now is going to be 5.04. Peter register makes sense when expressed in base10, my
     register will express the same number but is going to make sense in hex. (V5.05 -> PM way: 0x01f9, My way: 0
     x0505).
-Changed names of constants:
        -From FW_VERSION to FWVVERHEX at: pix/PixCmdDecodeV45.vhd, constantsPkg.vhd
        -From CODEID to FWVERDEC at seq/Personality_ModuleV45.vhd, constantsPkg.vhd
        -From ADDR_CODEIDREG to ADDR_FWVERDEC
        -From ADDR_FWVERSREG to ADDR_FWVERHEX


--------------------------------------------------------------------------------
2009.11.30 - changing firmware
--------------------------------------------------------------------------------
Although I'm saving each change I do to the code into our subversion server I start a log to easily know what
     changes I did and when. This is for future reference. All previous changes I did on the past can be tracked
     using diff between subversion revisions.

-Change in top mcbFull and in mcb.ucf, from clk2, clk3,...clk8 to a bus bp_clk(8 downto 2)
-Commented out ack_n in top ports. Added to todo list to check if in previous versions of the FW this port was
     connected, I don't remember. Right now it was completely disconnected.
-Changed name of board_slct_n to bp_boardSel_n (bp stands for backplane)
-Changed bus name from spr to bp_spare (backplane spare). They were floating, setted to 'Z'.
-Changed slink signals to xxx to sl_xxx in top
-Changed enable signal name of sync delay chip from delChipE to delChipSyncSel. Now both delay chips (sync and
     clock) use the same serial bus, so I must enfatize which delay chip are we selecting with the enable.

-Aesthetics changes in Pan_Command_Decode.vhd
-Removed old code which was commented out in Personality_ModuleV45.vhd
-Added constants CODEID and BOARDID in constants package, removed component IdentityModule from Peronality Module
     , Deleted IdentityModule file. Identity Module only implemented this two variables (CODEID and BOARDID).
-Some Aesthetics changes in Personality Module
-Commented out port CMD_ACK in PixCmdDecodeV461 as it was not connected.
-Commented out CsEnable signal in PixCmdDecodeV461 as it was not used in any process
-Some Aesthetics changes in PixCmdDecode.
-In Personality Module changed decode_vector type selector to simple selector which simplifies a lot adding new
     registers or memory sectors.
```