CAEN 可编程逻辑模块

发布 *V1.0*

Hongyi Wu(吴鸿毅)

2019 年 12 月 31 日

Contents:

如果您需要固件的支持，请联系吴鸿毅（wuhongyi@qq.com / wuhongyi@pku.edu.cn）

CHAPTER 1

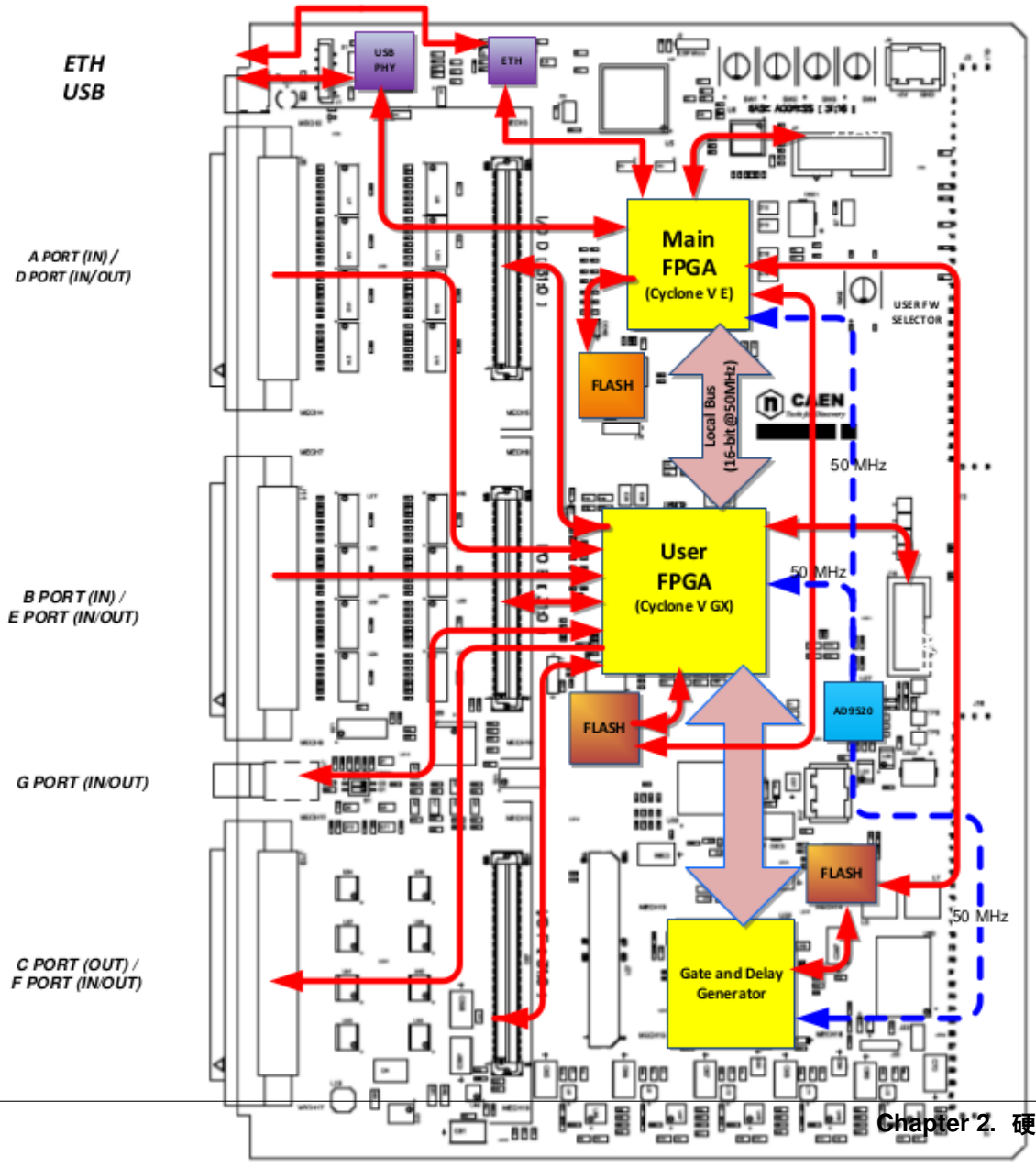
简介

CAEN DT5495 可编程逻辑单元使用笔记。可用于 V2495/DT5495。

阅读吴鸿毅的使用说明，请访问 http://wuhongyi.cn/CAENx495/

CHAPTER 2

硬件


ETH
USB
A PORT (IN) /
D PORT (IN/OUT)
B PORT (IN) /
E PORT (IN/OUT)
G PORT (IN/OUT)
C PORT (OUT) /
F PORT (IN/OUT)
USB PHY
ETH
Main FPGA (Cyclone V E)
USER FW SELECTOR
FLASH
Local Bus (16-bit@50MHz)
50 MHz
User FPGA (Cyclone V GX)
50 MHz
AD9520
FLASH
FLASH
50 MHz
Gate and Delay Generator

2.1 USB Interface

The DT5495 is equipped with a USB2.0 interface. The USB physical layer is managed by a high-speed transceiver controlled by the Main FPGA.

2.2 Ethernet Interface

In addition to the USB, the DT5495 provides a 10/100T Ethernet interface controlled by the Main FPGA.

2.3 Main FPGA

The MFPGA (Altera Cyclone V E) manages the Ethernet and USB interfaces and the connection with the UFPGA through a proprietary 16-bit 50 MHz local bus. The MFPGA has a dedicated external flash memory for configuration purposes.

It also pilots the flash memories dedicated to loading the firmware on the UFPGA and on the GDG.

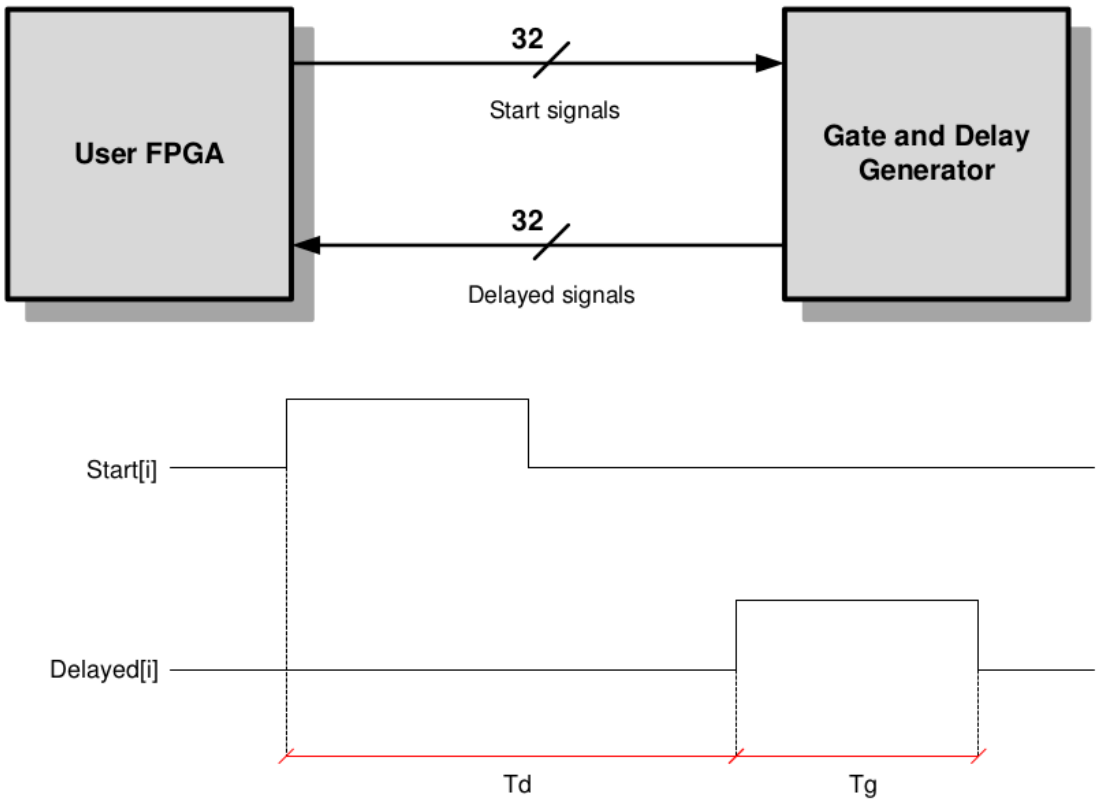
2.4 User FPGA

The User FPGA (Altera Cyclone V GX) manages the I/O peripherals (A/D, B/E, G, C/F ports) and communicates with the GDG. A dedicated external flash memory can store a set of firmware images to be loaded on the User FPGA. A dedicated JTAG connector allows to program the UFPGA "on-the-fly" for fast firmware prototyping and debugging.

2.5 Gate and Delay Generator

The DT5495 hosts a Gate and Delay Generator able to provide up to 32 gated and delayed signals("delayed signals") triggered by 32 inputs ("start signals"). The gate width and delay value are user programmable. The GDG is an external component implemented in a Xilinx Spartan-6 FPGA. It is connected through a serial bus (SPI) to the User FPGA for gate and delay register programming (refer to Sect. Gate and Delay Controller for detailed information).

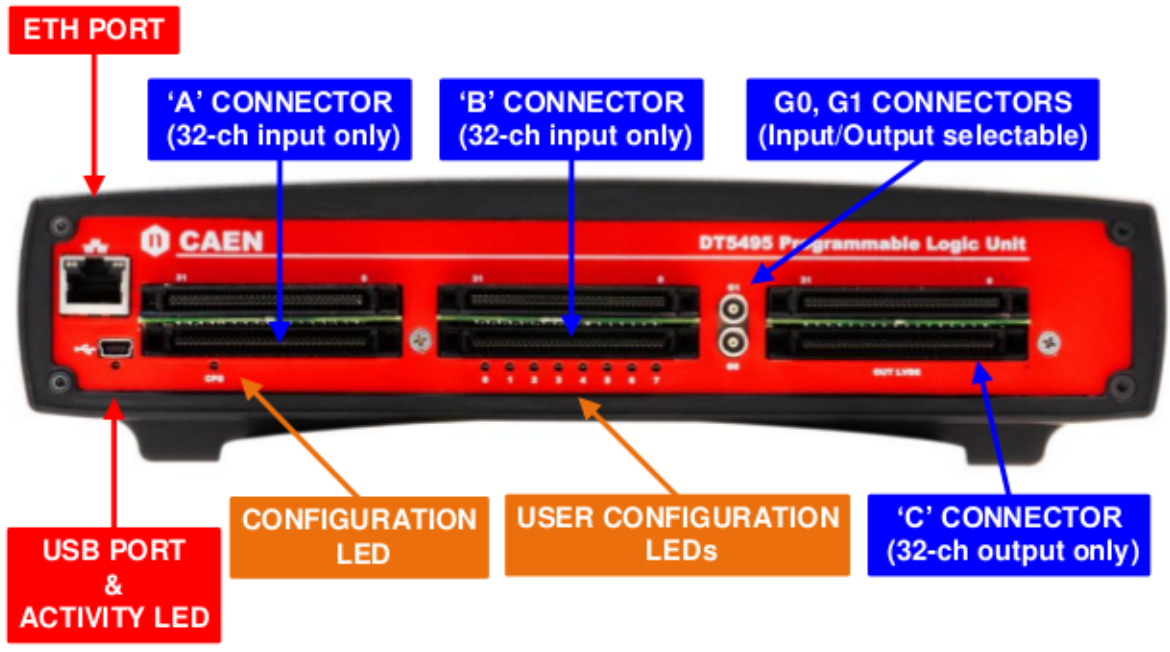
The GDG configuration is stored in a dedicated flash memory. The GDG firmware cannot be modified by the user.

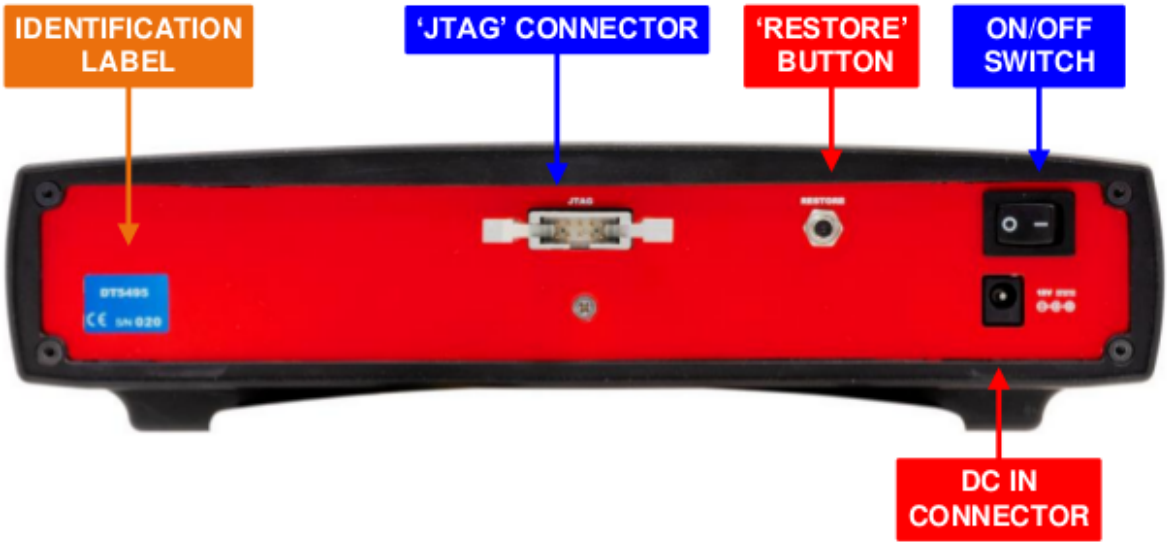


2.6 Clock Distribution

Each FPGA receives the same 50-MHz system clock generated by a common on-board oscillator.

2.7 Front/Rear Panel




IDENTIFICATION LABEL
'JTAG' CONNECTOR
'RESTORE' BUTTON
ON/OFF SWITCH
DC IN CONNECTOR

CHAPTER 3

扩展子板

3.1 User FPGA I/O ports

This section illustrates the I/O ports of the UFPGA. Port names are the same of the VHDL entity top-level ports used in the template and demo firmware.

3.2 User’s Code

3.2.1 G port

输入输出控制：

```
-- 输入
nOEG <= '1';

-- 输出
nOEG <= '0';
```

NIM/TTL 选择控制：

```
-- NIM
SELG <= '0';

-- TTL
SELG <= '1';
```

CHAPTER 4

系统固件

4.1 CAENUpgrader

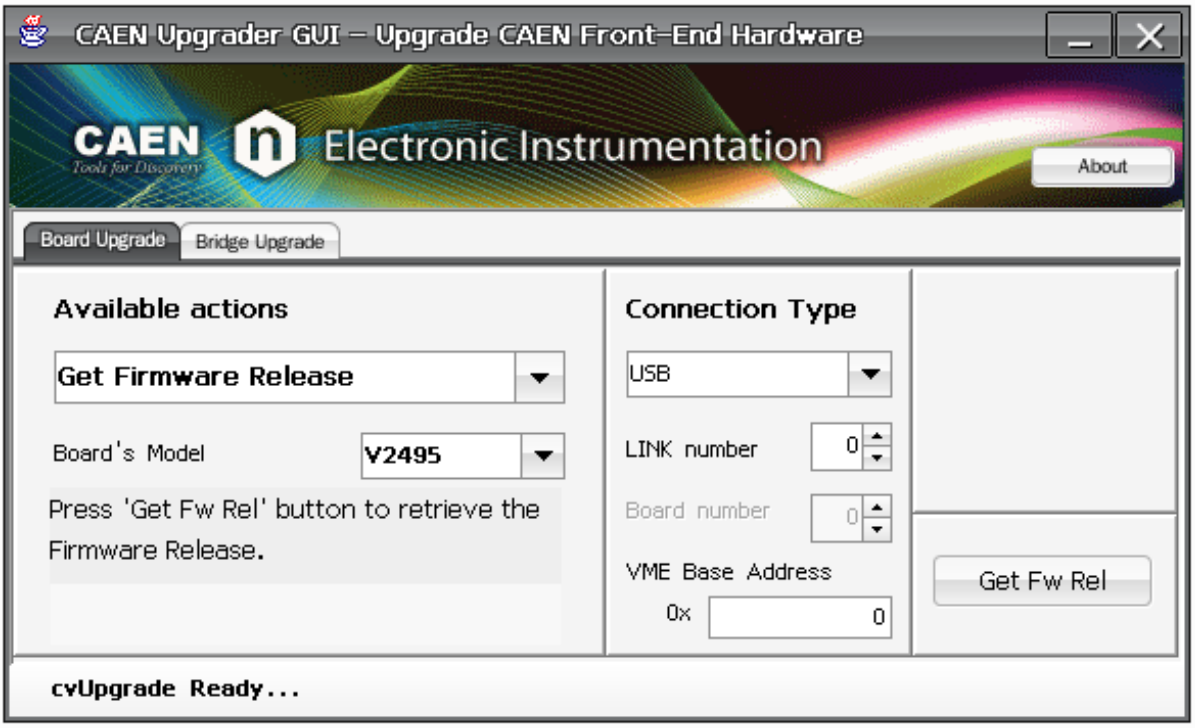
CAENUpgrader allows to perform the following operations on the DT5495 board:

- Upgrade the MFPGA firmware
- Load the User firmware on the UFPGA
- Verify the Main and User FPGA firmware
- Read the release number of the MFPGA firmware
- Store and get the Product Unlock Code in case of paid firmware
- Get the Board Info file (useful in case of support requests)

4.1.1 Get Main FPGA Firmware

The following instructions allow to read the release number of the MFPGA firmware:

- Select the “Get Firmware Release” option in the Board tab.
- Select the “V2495” item in the Board Model combo box.
- Set the connection parameters according to your communication link and hardware setup.
- Press the Get FW Rel button. The software displays the release number in a message window.

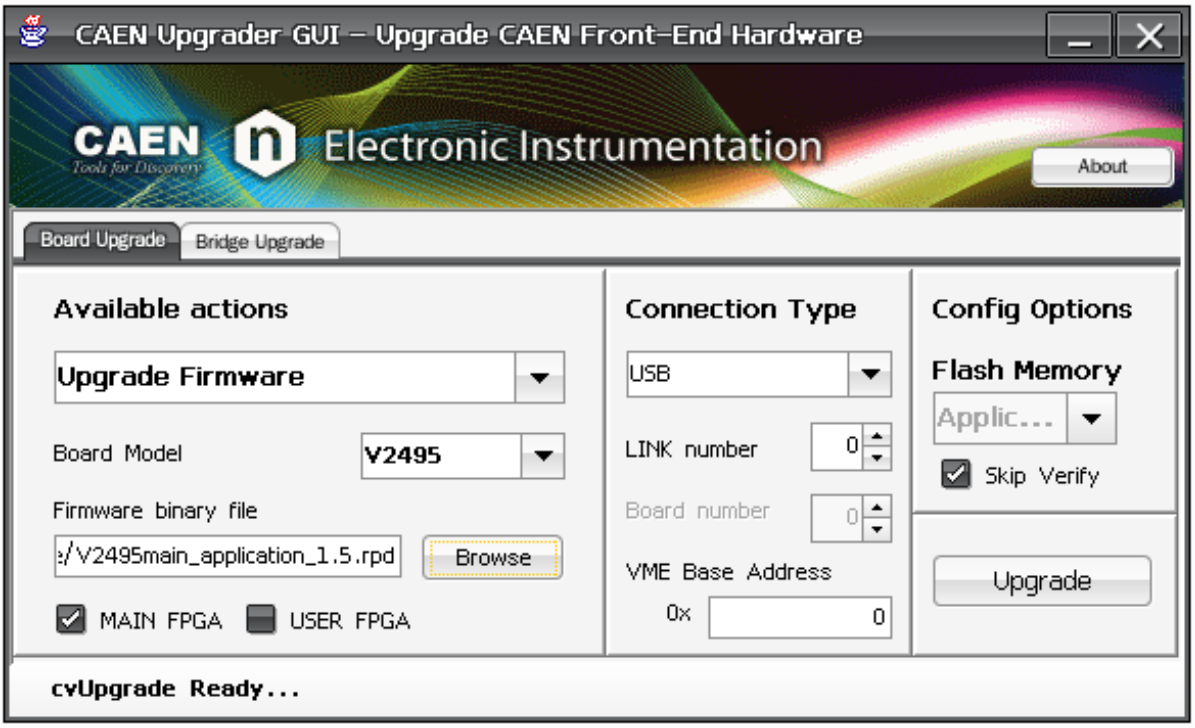


4.1.2 Upgrade Main FPGA Firmware

The MFPGA firmware can be downloaded from CAEN website at the DT5495 web page: The firmware file is available in Altera RPD format.

The following instructions allow to upgrade the MFPGA firmware:

- Select the “Upgrade Firmware” option in the Board tab.
- Select the “DT5495” item in the Board Model combo box.
- Set the connection parameters according to your communication link and hardware setup.
- Select the MFPGA by checking the relevant checkbox.
- Select the Main application firmware RPD file.
- Press the Upgrade button. The outcome of the upgrade process is shown in a message window.
- Power cycle the board to reconfigure the MFPGA with the firmware just updated on the flash. Please refer to Sect. Power-on Configuration Sequence to check the outcome of the process.



4.1.3 Upgrade User FPGA Firmware

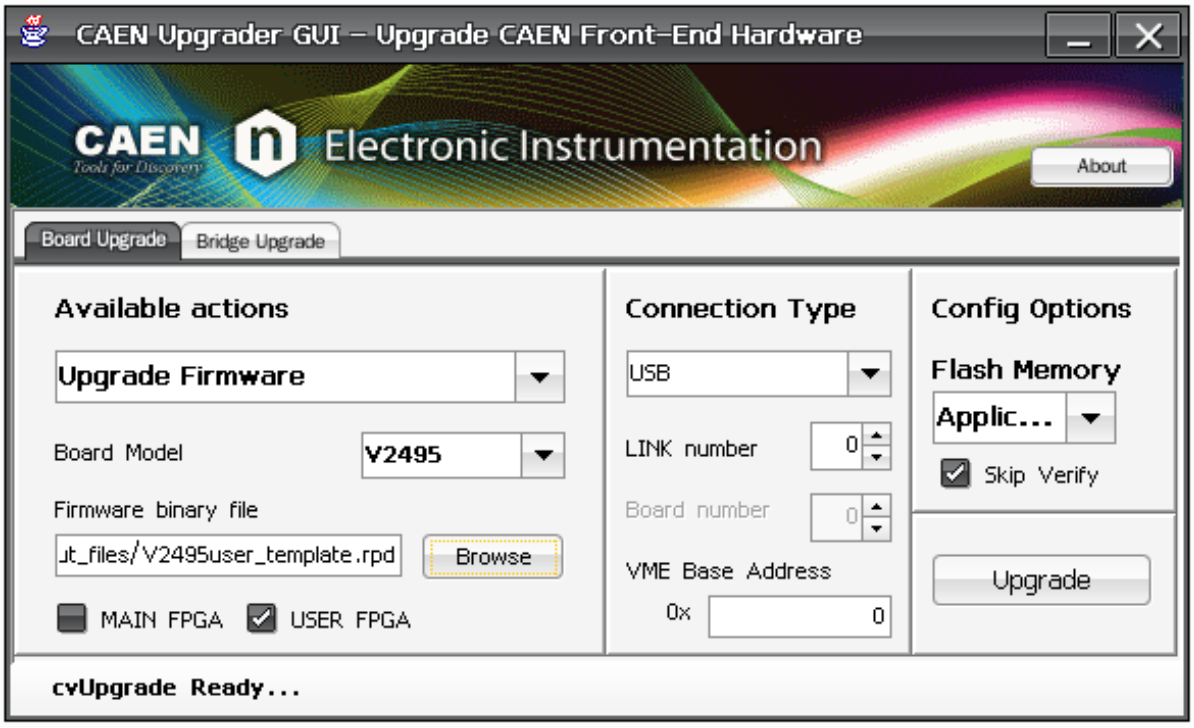
The UFPGA demo firmware projects are available on CAEN website at the DT5495 web page.

User demo firmware or custom user firmware can be recompiled and the generated RPD file can be uploaded into the FLASH memory of the User FPGA.

The custom or the Demo RPD firmware file can be uploaded only on the application 1 page of the USER FPGA FLASH of the DT5495!

The following instructions allow to upgrade the UFPGA firmware: - Select the “Upgrade Firmware” option in the Board tab. - Select the “V2495” item in the Board Model combo box. - Set the connection parameters according to your communication link and hardware setup. - Select the UFPGA by checking the relevant checkbox. - Select Application 1 as the only supported target application image. - Select your User firmware or a CAEN Demo firmware RPD file. - Press the Upgrade button. The outcome of the upgrade process is shown in a message window.

At the end of the upgrade process, the CAEN Upgrader will force a reconfiguration of the UFPGA from the selected userapplication image.



4.2 Address Map

The DT5495 board can be accessed via both Ethernet and USB interface. All registers are accessible from both communication interfaces.

The following table illustrates the DT5495 address map:

Address	Description
0x0000-0x0FFF	UFPGA data access
0x1000-0x7FFF	UFPGA register access
0x8000-0x80FF	*reserved*
0x8100-0x81FF	Configuration ROM
0x8200-0x83FF	Configuration and Status Registers
0x8400-0x84FF	Reserved
0x8500-0x86FF	MFPGA flash configuration
0x8700-0x88FF	UFPGA flash configuration
0x8900-0x8AFF	GDG flash configuration
0x8B00-x8BFF	Reserved
0x8C00-0x8FFF	Internal scratch RAM
0x9000-0xFFFF	reserved

4.2.1 User FPGA Data Access

The UFPGA data access space is allocated specifically for the readout of data produced in the UFPGA logic which need to exploit the maximum readout throughput available. This space can be accessed by using the available block transfer mechanism over the Ethernet or USB communication interfaces.

Block data transfer allows to implement a faster readout. Block transfers requests over Ethernet or USB do not trigger any transfer over the internal local bus between the MFPGA and UFPGA: a data prefetch mechanism, implemented in the MFPGA, stores User data into a local data queue (prefetch data queue). Consequently, data from the UFPGA data access space are read from the prefetch data queue in the MFPGA.

4.2.2 User FPGA Register Access

UFPGA register address range is allocated for User register read/write.

A read or write transfer (single cycle) can be triggered on the local bus by performing a corresponding read transfer in the 0x1000-0x7FFF address interval.

The local bus master in the MFPGA acts as a transparent bridge between the communication interface (ETH/USB) and the local bus in this interval. A couple of examples are given to illustrate the transparent bridge behaviour.

4.2.3 Configuration ROM

Only the 8 LSBs of each location of the Configuration ROM are significant.

Description	Address	Content
checksum	0x8100	An eight bit 2's complement binary checksum. The sum of the bytes starts from offset 0x8104 for the number of bytes specified in the length field (inclusive)
checksum_length2	0x8104	0x00
checksum_length1	0x8108	0x00
checksum_length0	0x810C	0x20
constant2	0x8110	0x84
constant1	0x8114	0x84
constant0	0x8118	0x01
c_code	0x811C	0x43
r_code	0x8120	0x52
oui2	0x8124	0x00
oui1	0x8128	0x40
oui0	0x812C	0xE6
version	0x8130	0x00
board2	0x8134	0x00
board1	0x8138	0x09
board0	0x813C	0xBF
revis3	0x8140	0x00
revis2	0x8144	0x00
revis1	0x8148	0x00
revis0	0x814C	PCB revision
sernum1	0x8180	Serial Number (MSB)
sernum0	0x8184	Serial Number (LSB)

Note: The oui0/1 fields represent CAEN Manufacturer identifier (IEEE OUI), which is equal to 0x40E6.

Note: The board serial number can be read with two accesses: if the serial number on the module front panel is 1245(hex 0x4DD), for instance, the Serial Number' s MSB (0x8180) will be 0x04, while the Serial Number' s LSB (0x8184) will be 0xDD.

4.2.4 Configuration and Status Registers

Address	Register/Content	Read/Write
0x8200	MFPGA firmware revision	R
0x8204 ÷ 0x8214	*reserved*	
0x8218	Software reset	W
0x8220	Scratch register	R/W
0x8224 ÷ 0x83FC	*reserved*	-

MFPGA Firmware Revision Register

Bit	Description
[31:16]	reserved
[15:8]	Major revision number
[7:0]	Minor revision number

Software Reset Register

Bit	Description
[31:0]	The value written into this register will determine the kind of software reset: 1 = generate a local bus reset only (nLBRES local bus signal) Others = reserved for future options

Scratch Register

Bit	Description
[31:0]	This register allows to perform 32-bit accesses for test purposes. Default value is 0xAAAAAAAA

4.2.5 Flash Configuration

This address range is reserved to flash remote programming.

Note: access to the FLASH is through the provided functions of the PLULib library.

4.2.6 Internal Scratch SRAM

This area is available either for test access or for volatile data storage. Any address in this interval is implemented by an internal RAM location.

CHAPTER 5

PLULib

5.1 PLULib Test

The connection parameters to correctly run the demo are:

- **connection type -> the identifier of the active communication link:**
  - 0 = USB direct link
  - 1 = Ethernet link
  - 2 = USB-to-VME through the V1718 CAEN Bridge
  - 3 = CONET-to-VME through the V2718 CAEN Brdige
- device_serial_number -> the serial number of the target board. This parameter is meaningful only in case of USB direct link; do not add it to the command string otherwise.
- ip -> the IP address of the target board. This parameter is meaningful only in case of Ethernet connection to the DT5495 board; do not add it to the command string otherwise.
- b -> the VME Base Address of the target board. This parameter is meaningful only in case of connection to the V2495 board through a CAEN Bridge (V2718/V1718); do not add it to the command string otherwise.

```
# 很奇怪的问题，我们插件的控制需要用以下的命令
./CAENPLUTest -c 0 -sn 0025;
```

PLULib 支持以下方式的通讯:

- PC → USB-to-miniUSB → V2495/DT5495
- PC → ETH → DT5495
- PC → PCI (A2818) → CONET → V2718 → VME → V2495
- PC → PCIe (A3818) → CONET → V2718 → VME → V2495
- PC → USB → V1718 → VME → V2495

5.2 Error Codes

```
enum CAEN_PLU_ERROR_CODE {
              CAEN_PLU_OK =0,
              CAEN_PLU_GENERIC =-1,
              CAEN_PLU_INTERFACE =-2,
              CAEN_PLU_FPGA =-3,
              CAEN_PLU_TRANSFER_MAX_LENGTH =-4,
              CAEN_PLU_NOTCONNECTED =-5,
              CAEN_PLU_NO_DATA_AVAILABLE =-6,
              CAEN_PLU_TOO_MANY_DEVICES_CONNECTED =-7,
              CAEN_PLU_INVALID_HANDLE =-8,
              CAEN_PLU_INVALID_HARDWARE =-9,
              CAEN_PLU_INVALID_PARAMETERS = -10
};
```

Error Code	Value	Description
OK	0	No errors.
GENERIC	-1	Generic (not specified) error.
INTERFACE	-2	Interface error while connecting to the target board.
FPGA	-3	FPGA internal error.
TRANSFER_MAX_LENGTH	-4	Transfer size in a read or write access exceeds 16 MB.
NOTCONNECTED	-5	Attempting to perform a read or write access while the target board is not connected.
NO_DATA_AVAILABLE	-6	No data available.
TOO_MANY_DEVICES_CONNECTED	-7	Attempting to connect to more than 100 devices.
INVALID_HANDLE	-8	Invalid handle used.
INVALID_HARDWARE	-9	Attempting to connect through an invalid hardware/interface.
INVALID_PARAMETERS	-10	At least one of the function parameters is not valid (value is not in the allowed range).

5.3 库函数

5.3.1 OpenDevice

The connection with a device in your network can be established using this function. It returns a handle that can be used later to interact with the system.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_OpenDevice(t_ConnectionModes connection_mode,
                                    char *IPAddress_or_SN_or_VMEBaseAddress,
                                    int TCPPort_or_VMElink,
                                    int UDPPort_or_VMEConetNode,
                                    int *handle);
```

Name	I/O	Description
connection_mode	Input	Indicates the physical communication channel. It can be: - *CAEN_PLU_CONNECT_DIRECT_USB* (direct USB connection)) - *CAEN_PLU_CONNECT_DIRECT_ETH* (direct Ethernet connection) - *CAEN_PLU_CONNECT_VME_V1718* (USB-to-VME connection through V1718) - *CAEN_PLU_CONNECT_VME_V2718* (Optical-to-VME connection through V2718) See **t_ConnectionModes**.
IPAddress_or_SN_or_VMEBaseAddress	Input	Pointer to the: - IP address of the unit in case of Ethernet connection; - Serial Number of the unit in case of direct USB connection; - VME Base Address of the unit (only V2495) in case of connection through CAEN Bridges.
VMElink	Input	Link number (VME). In case of USB and Ethernet connections, this paramenter is not significant.
VMEConetNode	Input	Conet node (VME with V2718 bridge). In case of USB and Ethernet connections, or VME V1718, this paramenter is not significant.
handle	Output	Returns a handle for subsequent library accesses. Can be NULL if connection is not possible.

CAEN_PLU_OK (0) in case of success. Negative numbers are error codes (see Error Codes).

```
// Connecting to a PLU module (V2495 or DT5495) via direct USB link:
ret = CAEN_PLU_OpenDevice(CAEN_PLU_CONNECT_DIRECT_USB, "4", 0, 0, &handle);

// A connection via a Ethernet is opened with:
ret = CAEN_PLU_OpenDevice(CAEN_PLU_CONNECT_DIRECT_ETH, "192.168.7.11", 0, 0, &
↪handle);

// V1718 access can be opened with:
ret = CAEN_PLU_OpenDevice(CAEN_PLU_CONNECT_VME_V1718, vme_base_address, 0, 0, &
↪handle);

// V2718 bridge connection can be opened with:
ret = CAEN_PLU_OpenDevice(CAEN_PLU_CONNECT_VME_V2718, vme_base_address, 0, 0, &
↪handle);

// In both the latter cases, a VME Base Address of the PLU module must be␣
↪specified.
```

5.3.2 CloseDevice

This function closes the connection with the programmable logic unit. The CloseDevice function must be called before to exit the application.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_OpenDevice(int handle);
```

Return Values : 0: Success. Negative numbers are error codes (see Error Codes).

```
// Close the device with:
ret = CAEN_PLU_CloseDevice(handle);
```

5.3.3 WriteReg

Generic write access to a register of the device.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_WriteReg(
     int handle,
     uint32_t address,
     uint32_t value
);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
address	Input	Register address (for the VME access, this is the lower 16-bit part of the VME address bus).
value	Input	32-bit data to write at the addressed register.

Return Values: 0: Success. Negative numbers are error codes.

5.3.4 ReadReg

Generic read access to a register of the device.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_ReadReg(int handle,
                 int32_t address,
                 int32_t *value);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
address	Input	Register address (for the VME access, this is the lower 16-bit part of the VME address bus).
value	Output	Pointer to the 32-bit data read at the addressed register.

Return Values: 0: Success. Negative numbers are error codes.

5.3.5 WriteData32

This function writes 32-bit data into memory.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_WriteData32(int handle,
                     uint32_t start_address,
                     uint32_t size,
                     uint32_t *value);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
start_address	Input	Start adddress for read operation. Address automatically incremented for each new value access.
size	Input	Size of transfert in 32-bit words.
value	Output	Pointer to the values to write.

Return Values: 0: Success; Negative numbers are error codes.

5.3.6 WriteFIFO32

This function writes 32-bit data at the same address (FIFO mode).

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_WriteFIFO32(int handle,
                     uint32_t start_address,
                     uint32_t size,
                     uint32_t *value);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
start_address	Input	Start address for write operation. Address is **NOT** incremented for each new value access.
size	Input	Size of transfert in 32-bit words.
value	Output	Pointer to the values to write.

Return Values: 0: Success. Negative numbers are error codes.

5.3.7 ReadData32

This function reads 32-bit data from memory.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_ReadData32(int handle,
                    uint32_t start_address,
                    uint32_t size,
                    uint32_t *value,
                    uint32_t *nw);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
start_address	Input	Start adddress for read operation. Address automatically incremented for each new value access.
size	Input	Size of transfer in 32-bit words.
value	Output	Pointer to read values.
nw	Output	Number of 32-bit words read. It is less or equal to *size*.

Return Values: 0: Success. Negative numbers are error codes.

5.3.8 ReadFIFO32

This function reads 32-bit data from the same adddress (FIFO mode).

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_ReadFIFO32(int handle,
                    uint32_t address,
                     uint32_t size,
                     uint32_t *value,
                     uint32_t *nw);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
start_address	Input	Start adddress for read operation. Address is **NOT** incremented for each new value access.
size	Input	Size of transfer in 32-bit words.
value	Output	Pointer to read values.
nw	Output	Number of 32-bit words read. It is less or equal to *size*.

Return Values: 0: Success. Negative numbers are error codes.

5.3.9 USBEnumerate

This function enumerates the boards connected via USB direct link.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_USBEnumerate(tUSBDevice *pvArg1,
                      uint32_t *numDevs);
```

Name	I/O	Description
pvArg1	Output	Pointer to USB devices enumerated.
numDevs	Output	Number of enumerated boards.

Return Values: 0: Success. Negative numbers are error codes.

5.3.10 USBEnumerateSerialNumber

This function enumerates the boards connected via USB direct link and returns a Serial Number as a string.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_USBEnumerateSerialNumber(unsigned int *numDevs,
                                  char *DeviceSNs,
                                  uint32_t buffersize);
```

Name	I/O	Description
numDevs	Output	Number of enumerated devices.
deviceSNs	Output	Serial number.
buffersize	Input	String length.

Return Values: 0: Success. Negative numbers are error codes.

5.3.11 InitGateAndDelayGenerators

This function performs the Gate and Delay initialization. **It MUST be called prior to any Gate and Delay function call.**

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_InitGateAndDelayGenerators(int handle);
```

Return Values: 0: Success. Negative numbers are error codes.

5.3.12 SetGateAndDelayGenerator

This function enables and sets **a single** gate and delay generator channel.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_SetGateAndDelayGenerator(int handle,
                                  uint32_t channel,
                                  uint32_t enable,
                                  uint32_t gate,
                                  uint32_t delay,
                                  uint32_t scale_factor);
```

Note: Gate+Delay parameters cannot exceed 65535.

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
channel	Input	Gate and Delay channel to set.
enable	Input	Channel enable.
gate	Input	Gate value in gate steps (valid range = 0-65535.)
delay	Input	Delay value in delay steps (valid range = 0-65535).
scale_factor	Input	Scale factor for delay (valid range = 0-255). 0 is the minimum gate and delay resolution (~10 ns).

Return Values: 0: Success. Negative numbers are error codes.

5.3.13 SetGateAndDelayGenerators

This function enables and set **ALL** gate and delay generators channels with a common value.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_SetGateAndDelayGenerators(int handle,
                                   uint32_t gate,
                                   uint32_t delay,
                                   uint32_t scale_factor);
```

Note: Gate+Delay parameters cannot exceed 65535.

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
gate	Input	Gate value in gate steps (Valid range = 0-65535).
delay	Input	Delay value in delay steps (Valid range = 0-65535).
scale_factor	Input	Scale factor for delay (Valid range = 0-255). 0 is the minimum gate and delay resolution (~10 ns).

Return Values: 0: Success. Negative numbers are error codes.

5.3.14 GetGateAndDelayGenerator

This function gets the Gate and Delay channel parameters.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_GetGateAndDelayGenerator(int handle,
                                  uint32_t channel,
                                  uint32_t *gate,
                                  uint32_t *delay,
                                  uint32_t *scale_factor);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
channel	Input	Gate and Delay channel.
gate	Output	Gate value in gate steps.
delay	Output	Delay value in delay steps.
scale_factor	Output	Fine tune value in fine tune steps.

Return Values: 0: Success. Negative numbers are error codes.

5.3.15 EnableFlashAccess

By this function, it is possible to enable the Flash access. **It MUST be called prior to any Flash access function call.**

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_EnableFlashAccess(int handle,
                           t_FPGA_V2495 FPGA);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
FPGA	Input	The possible target FPGA: - *FPGA MAIN* (MAIN FPGA); - *FPGA USER* (USER FPGA); - *FPGA DELAY* (GATE AND DELAY FPGA). See **t_FPGA_V2495**.

Return Values: 0: Success. Negative numbers are error codes.

5.3.16 DisableFlashAccess

By this function, it is possible to disable the Flash access. **It MUST be called prior to any flash access function call.**

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_DisableFlashAccess(int handle,
                            t_FPGA_V2495 FPGA);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
FPGA	Input	The possible target FPGA: - *FPGA MAIN* (MAIN FPGA); - *FPGA USER* (USER FPGA); - *FPGA DELAY* (GATE AND DELAY FPGA). See **t_FPGA_V2495**.

Return Values: 0: Success. Negative numbers are error codes.

5.3.17 DeleteFlashSector

This function deletes a single Flash sector.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_DeleteFlashSector(int handle,
                           t_FPGA_V2495 FPGA,
                           uint32_t sector);
```

Note: Please, BE AWARE that some sectors are reserved for factory and user firmware. User storage area is in sectors 106-510 for MAIN Flash and sectors 458-510 for USER Flash. DELAY Flash should not be used for user data.

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
FPGA	Input	The possible target FPGA: - *FPGA MAIN* (MAIN FPGA); - *FPGA USER* (USER FPGA); - *FPGA DELAY* (GATE AND DELAY FPGA). See **t_FPGA_V2495**.
sector	Input	Flash se tor to delete (64 KB). MAIN and USER Flash (N25Q256 model) have 512x64KB sectors; GATE AND DELAY Flash (W25Q64 model) has 128x64KB sectors.

Return Values: 0: Success. Negative numbers are error codes.

5.3.18 WriteFlashData

This function allows to write data into the Flash.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_WriteFlashData(int handle,
                        t_FPGA_V2495 FPGA,
                        uint32_t address,
                        uint32_t *data,
                        uint32_t length);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
FPGA	Input	The possible target FPGA: - *FPGA MAIN* (MAIN FPGA); - *FPGA USER* (USER FPGA); *FPGA DELAY* (GATE AND DELAY FPGA). See **t_FPGA_V2495**.
address	Input	Flash start address.
data	Input	Pointer to data to write into the Flash.
length	Input	Data length in 32-bit words.

Return Values: 0: Success. Negative numbers are error codes.

5.3.19 ReadFlashData

This function allows to read data from the Flash.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_ReadFlashData(int handle,
                       t_FPGA_V2495 FPGA,
                       uint32_t address,
                       uint32_t *data,
                       uint32_t length);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
FPGA	Input	The possible target FPGA: - *FPGA MAIN* (MAIN FPGA); - *FPGA USER* (USER FPGA); *FPGA DELAY* (GATE AND DELAY FPGA). See **t_FPGA_V2495**.
address	Input	Flash start address.
data	Output	Pointer to data read from the Flash.
length	Input	Data length in 32-bit words.

Return Values: 0: Success. Negative numbers are error codes.

5.3.20 GetInfo

This function retrieves the module information.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_GetInfo(int handle,
                 tBOARDInfo *HWOPTIONS;
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
HWOPTIONS	Output	Pointer to a *tBOARDInfo* structure (see **tBOARDInfo)**: - *checksum*; - *checksum_length2*; - *checksum_length1*; - *checksum_length0*; - *checksum_constant2*; - *checksum_constant1*; - *checksum_constant0*; - *c_code*; - *r_code*; - *oui2*; - *oui1*; - *oui0*; - *version*; - *board2*; - *board1*; - *board0*; - *revis3*; - *revis2*; - *revis1*; - *revis0*; - *reserved[12]*; - *sernum1*; - *sernum0*.

Return Values: 0: Success. Negative numbers are error codes.

5.3.21 GetSerialNumber

This function retrieves the module serial number stored into the Configuration ROM

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_GetSerialNumber(int handle,
                         char *sn,
                         uint32_t buffersize);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
sn	Output	Serial number.
buffersize	Input	Serial number string length.

Return Values: 0: Success. Negative numbers are error codes.

5.3.22 ConnectionStatus

This function gets the current connection status from the unit.

```
CAEN_PLU_API CAEN_PLU_ERROR_CODE
CAEN_PLU_ConnectionStatus(int handle,
                          int *status);
```

Name	I/O	Description
handle	Input	Library handle (as returned by *CAEN_PLU_OpenDevice()*).
status	Output	Connection status with ID (0 = USB, 1 = ETH, 2 = V1718, 3 = V2718).

Return Values: 0: Success. Negative numbers are error codes.

5.4 数据结构和类型描述

5.4.1 t_ConnectionModes

Enumerated type for the kind of connection link.

```
typedef enum
{
      CAEN_PLU_CONNECT_DIRECT_USB,
      CAEN_PLU_CONNECT_DIRECT_ETH,
      CAEN_PLU_CONNECT_VME_V1718,
      CAEN_PLU_CONNECT_VME_V2718,
} t_ConnectionModes;
```

Value	Type	Description
CAEN_PLU_CONNECT_DIRECT_USB	enum	USB direct connection type.
CAEN_PLU_CONNECT_DIRECT_ETH	enum	Ethernet direct connection type.
CAEN_PLU_CONNECT_VME_V1718	enum	USB-to-VME connection type through the V1718 Bridge.
CAEN_PLU_CONNECT_VME_V2718	enum	CONET-to-VME connection type through the V2718 Bridge.

5.4.2 t_FPGA_V2495

Enumerated type for the kind of V2495/DT5495 target FPGA.

```
typedef enum
{
      FPGA_MAIN = 0,
      FPGA_USER = 1,
      FPGA_DELAY = 2
} t_FPGA_V2495;
```

Value	Type	Description
FPGA_MAIN	enum	MAIN FPGA.
FPGA_USER	enum	USER FPGA.
FPGA_DELAY	enum	GATE AND DELAY FPGA.

5.4.3 tBOARDInfo

This structure defines the board generic information from the Configuration ROM.

```
typedef struct _tBOARDInfo
{
      uint32_t checksum;
      uint32_t checksum_length2;
      uint32_t checksum_length1;
      uint32_t checksum_length0;
      uint32_t checksum_constant2;
      uint32_t checksum_constant1;
      uint32_t checksum_constant0;
      uint32_t c_code;
      uint32_t r_code;
      uint32_t oui2;
      uint32_t oui1;
      uint32_t oui0;
      uint32_t version;
      uint32_t board2;
      uint32_t board1;
```

(下页继续)

(续上页)

```
    uint32_t board0;
    uint32_t revis3;
    uint32_t revis2;
    uint32_t revis1;
    uint32_t revis0;
    uint32_t reserved[12];
    uint32_t sernum1;
    uint32_t sernum0;
} tBOARDInfo;
```

Name	Type	Description
checksum	uint32_t	Checksum value of the Configuration ROM space.
checksum_length2	uint32_t	3-byte checksum length (i.e. the number of bytes in the Configuration ROM to checksum).
checksum_length1	uint32_t	
checksum_length0	uint32_t	
checksum_constant2	uint32_t	3-byte Configuration Rom constant.
checksum_constant1	uint32_t	
checksum_constant0	uint32_t	
c_code	uint32_t	ASCII C character code (identifies this as CR space).
r_code	uint32_t	ASCII R character code (iden fies this as CR space).
oui2	uint32_t	3-byte IEEE Organizationally Unique Identifier (OUI).
oui1	uint32_t	
oui0	uint32_t	
version	uint32_t	Board version information.
board2	uint32_t	3-byte board ID.
board1	uint32_t	
board0	uint32_t	
revis3	uint32_t	4-byte hardware revision.
revis2	uint32_t	
revis1	uint32_t	
revis0	uint32_t	
reserved[12]	uint32_t	*n.a.*
sernum1	uint32_t	Board Serial Number.
sernum0	uint32_t	

5.4.4 _tUSBDevice

This structure defines the USB device descriptor.

```
typedef struct _tUSBDevice
{
    uint32_t id;
    char SN[64];
    char DESC[64];
} tUSBDevice;
```

Name	Type	Description
id	uint32_t	Incremental number of the enumerated interface (starts from 0).
SN[64]	char	The string of the serial number of the device.
DESC[64]	char	USB device string descriptor: can be “DT5495” or “V2495”.

CHAPTER 6

SCI-COMPILER

SCI

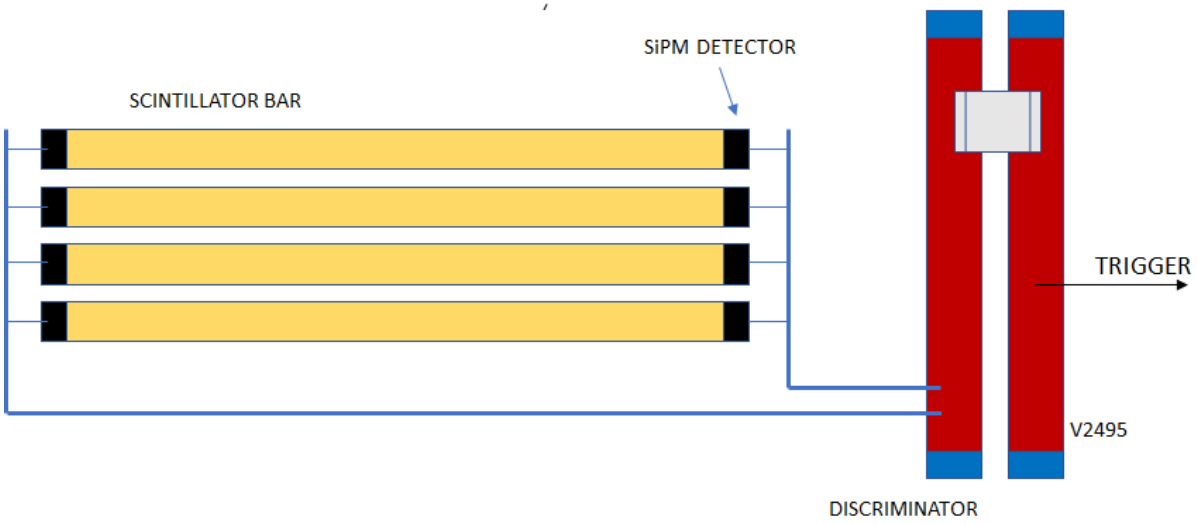
CHAPTER 7

COINCIDENCE TRIGGER

http://www.sci-compiler.com/example-sci-1.html

A discrete trigger circuit is a fundamental part of a large detector system used to reduce the readout data rate. The following trigger is based on scintillator bars coupled with SiPM detectors. The high energy particles will cross the scintillator bars releasing a fraction of their energy.

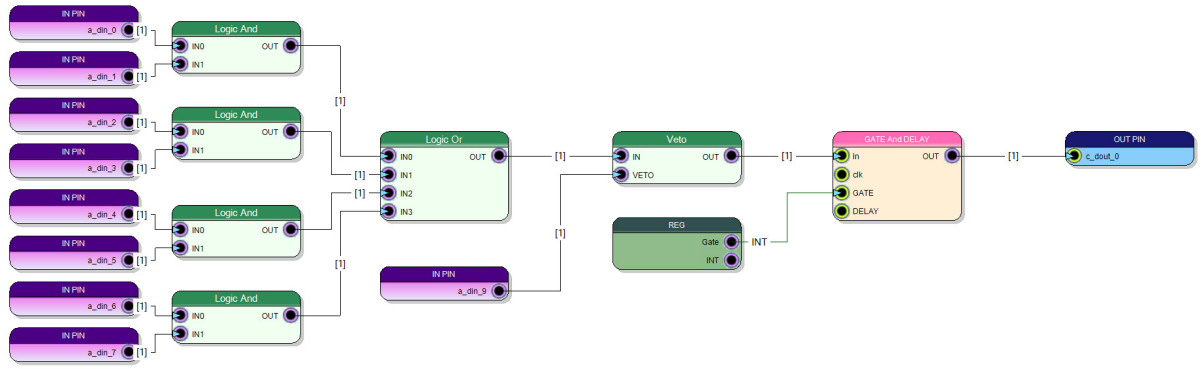
If a particle cross one of the scintillator bar, the scintillation light will be detected by SiPM detectors (i.e. Ispector detector). A discriminator generate a digital pulse converting the analog signals from detectors in digital pulses.



A Caen V2495 is used to make coincidence and trigger logic. In particular pulses generated by each side of the scintillator are in AND (coincidence) in order do distinguish real signal from noise/dark count. All scintillator bars are in OR in order to fire a trigger independently from the bar hit.

The gate and delay is used to configure the width of the pulse in output. The gate time is mapped on a register that can be configured with a PC using VME/USB bus

A veto logic is used to stop trigger generation.

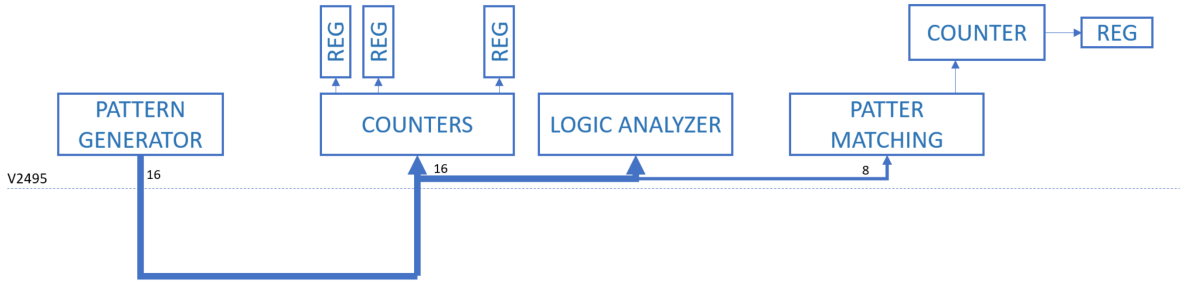

IN PIN
a_din_0
[1]
IN PIN
a_din_1
[1]
Logic And
IN0
IN1
OUT
IN PIN
a_din_2
[1]
IN PIN
a_din_3
[1]
Logic And
IN0
IN1
OUT
IN PIN
a_din_4
[1]
IN PIN
a_din_5
[1]
Logic And
IN0
IN1
OUT
IN PIN
a_din_6
[1]
IN PIN
a_din_7
[1]
Logic And
IN0
IN1
OUT
[1]
[1]
[1]
[1]
Logic Or
IN0
IN1
IN2
IN3
OUT
IN PIN
a_din_9
[1]
[1]
Veto
IN
VETO
OUT
REG
Gate
INT
INT
[1]
GATE And DELAY
In
clk
GATE
DELAY
OUT
[1]
OUT PIN
c_dout_0

CHAPTER 8

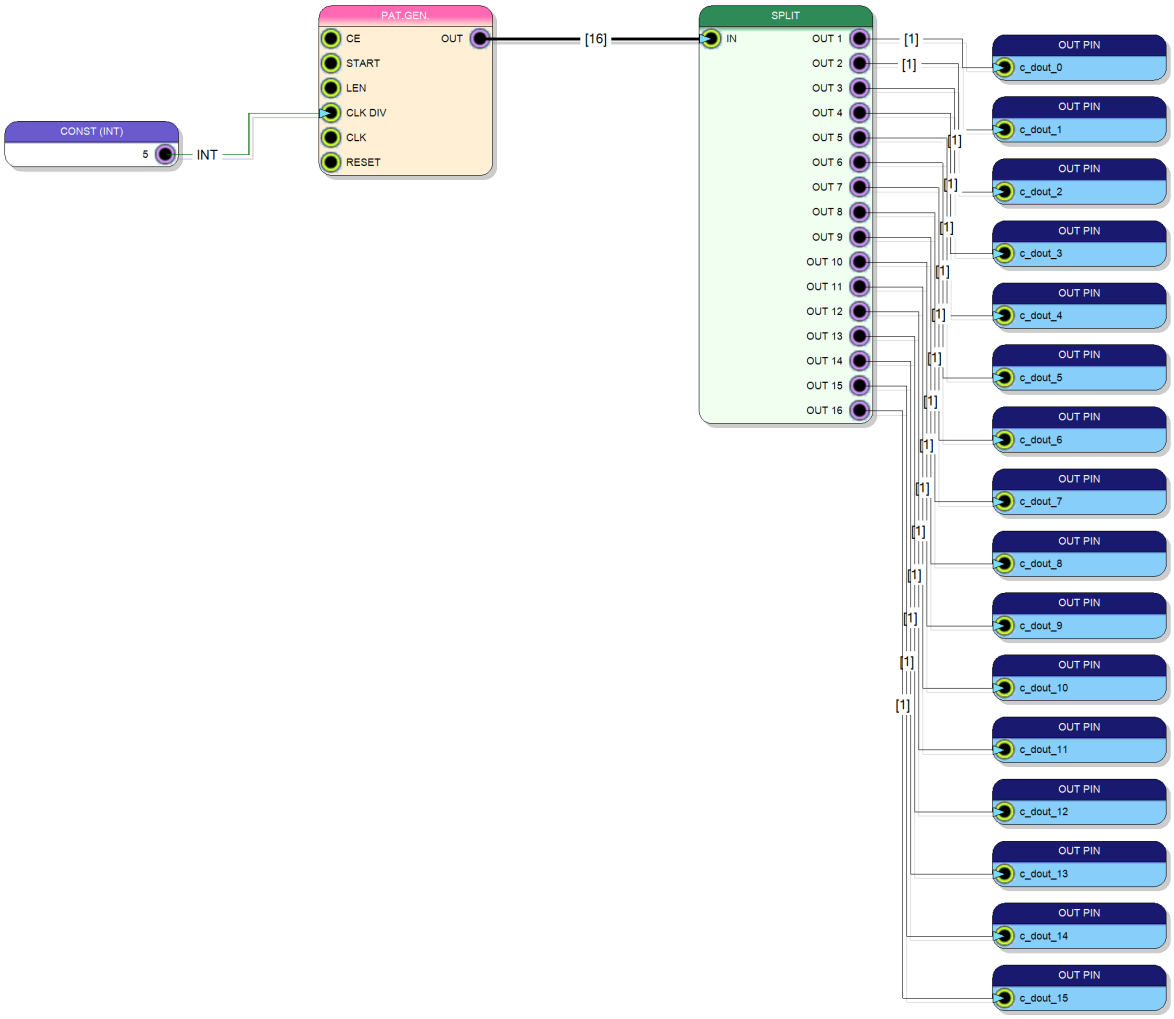
MULTICHANNEL SCALER AND PATTER MATCHING TRIGGER LOGIC

http://www.sci-compiler.com/example-sci-8.html

CAEN V2495 is a powerful programmable digital system designed to manage about 300 I/O signal in order to implement complex logic function, counters and triggers. In this example we will use the V2495 board to generate a pattern starting from the content of a file; the pattern generator is then used to stimulate a series of counters and a pattern matching state machine that can be programmed by USB, VME or Ethernet in order to recognize a two word pattern. All counters can be read out or resetted from USB, VME or Ethernet. When the pattern is recognized a Pattern Matching (PM) counter is increased. The PM counter can be read out by the comunication bus and heights of its bits (15:8) are shown on the 8 user leds. In order to readout and control counters or set the pattern in this example we will not use any custom software: SCICompiler has inside a tool called Resource Explorer. This tool enumerate all the pheripheral connected to the local bus of the V2495 device and list all available register and devices. In the firmware are inserted two logic analyzer: the first connected to all inputs (16 channels of A connector) and the second connected to the input and output of the patter matching state machine, in order to monitor the numeric value present on the bus.

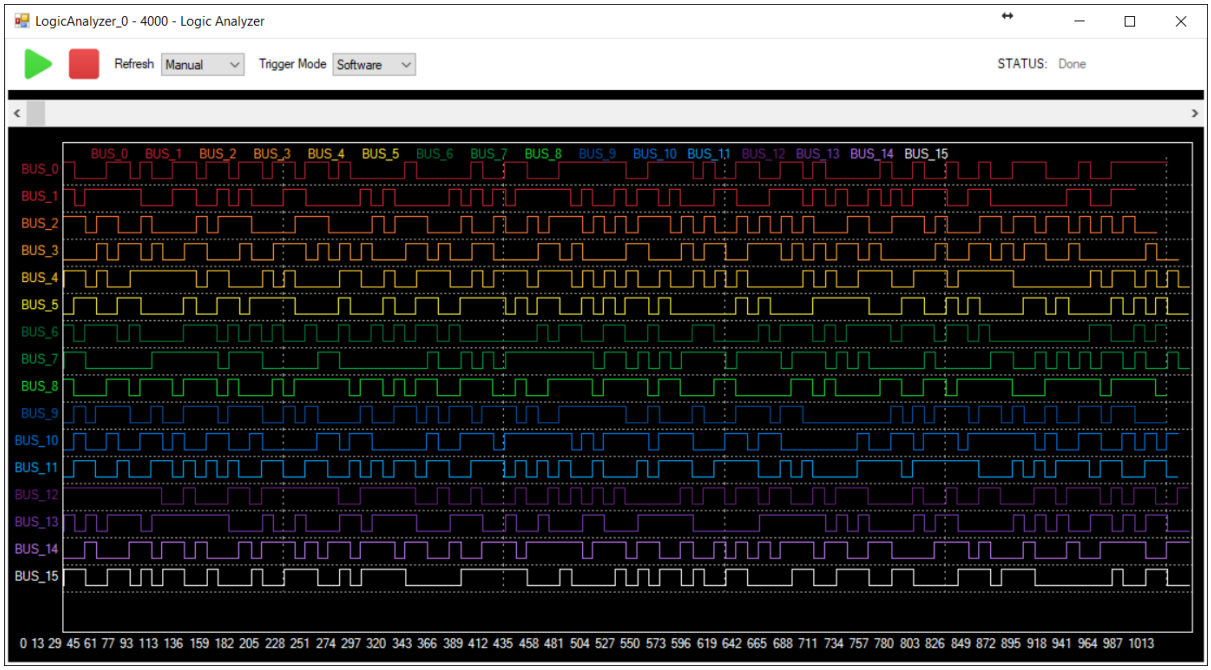


8.1 PATTERN GENERATOR



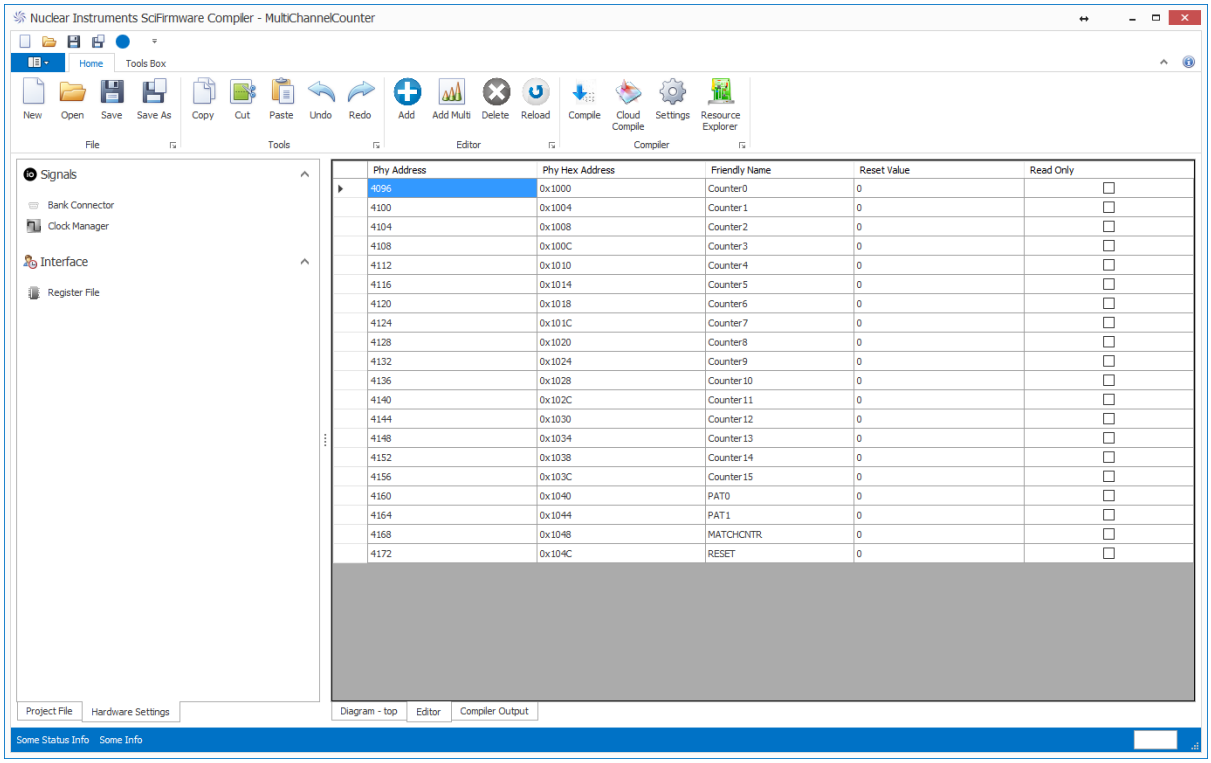
SCICompiler has different embedded signal generator. It is for example possible to generate exponential signals to test energy processing blocks, generate digital waveform or import file in binary, heaxadecimal or decimal, format in order to generate a periodic sequence. The Patter Generator can be configure in order to reproduce the sequence with a configurable speed. The Pattern reproduction can be started automatically or upon the transition of a start signal.

8.2 LOGIC ANALYZER



SCICompiler includes a powerful Logic Analyzer that can be used to remote monitor the input signal and the internal interconnection of the design. The Logic Analyzer can be used both douring the design phase to debug the processing core both during normal operation to control the status of I/O even from a remote desktop. If you are not sure to have connect all cables correctly now you can avoid to enter in the experimental area and you can just add a debug probe to a pin and se if it toggle. The Logic Analyzer can be triggered, internally (upon the transition of one of the input), externally (upon the trasition of a dedicated trigger input), or by software. Signal in the Logic Analyzer can be grouped into bus. More then one Logic Analyzer can be added to the same design. The resource exporer software is able to enumarate the variuos logic analyzers and implements a GUI to interact with them and show the waveforms. A DLL library (open source) is also available to integrate the Logic Analyzer GUI in a user application. Logic Analyzer data could also be dumped by library function using any programming language that support interaction with dynamic libraries (in Windows, Linux and MacOS) like C/CPP, Matlab, ROOT, Labview

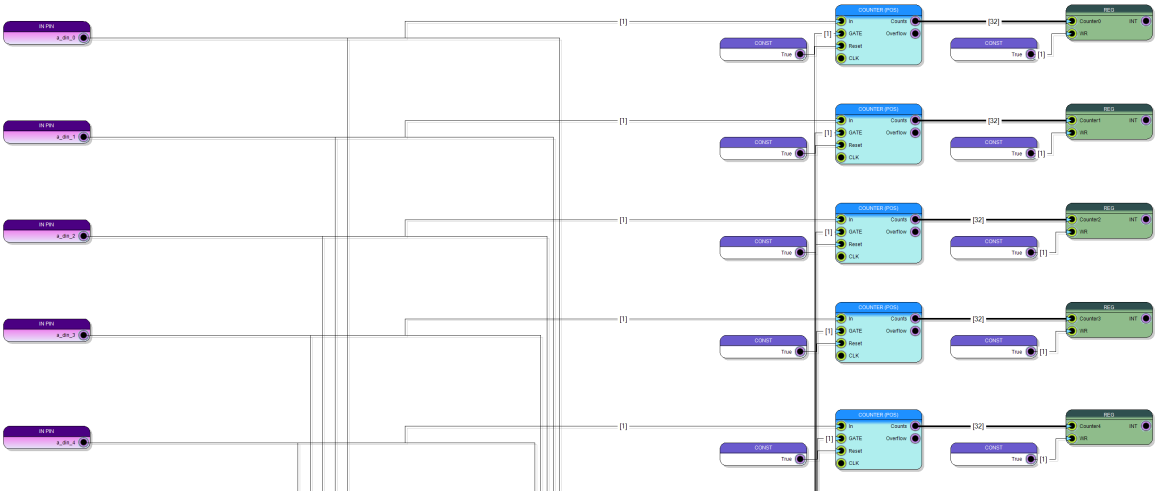
8.3 MEMORY MAPPED REGISTER



Phy Address	Phy Hex Address	Friendly Name	Reset Value	Read Only
4096	0x1000	Counter0	0	
4100	0x1004	Counter1	0	
4104	0x1008	Counter2	0	
4108	0x100C	Counter3	0	
4112	0x1010	Counter4	0	
4116	0x1014	Counter5	0	
4120	0x1018	Counter6	0	
4124	0x101C	Counter7	0	
4128	0x1020	Counter8	0	
4132	0x1024	Counter9	0	
4136	0x1028	Counter10	0	
4140	0x102C	Counter11	0	
4144	0x1030	Counter12	0	
4148	0x1034	Counter13	0	
4152	0x1038	Counter14	0	
4156	0x103C	Counter15	0	
4160	0x1040	PAT0	0	
4164	0x1044	PAT1	0	
4168	0x1048	MATCHCNTR	0	
4172	0x104C	RESET	0	

User Interaction by USB/VME/Ethernet bus is one of the most important feature of SCICompiler In the design of a processing system the implementation of a secure and fast communication protocol is one of the most time spending task. SCICompiler implements in its core all the necessary circuit to map Registers and Memory Buffer in a 16 bit for V2495 and 32 bit for DT5550 memory space direct addressable from a remote PC. All drivers and libraries for Windows and Linux are automatically generate by SCICompiler User tasks are limited to define the registers and connect them to internal signal of the design Register can be seen like an endpoint that can be written or read from the control PC. FIFO (List moduele or Image module) are more similar to a pipe. What is written inside disappear and is transported on a memory buffer on the PC. Spectrum memory can be seen as an external memory area that the computer can read/write specifying the address

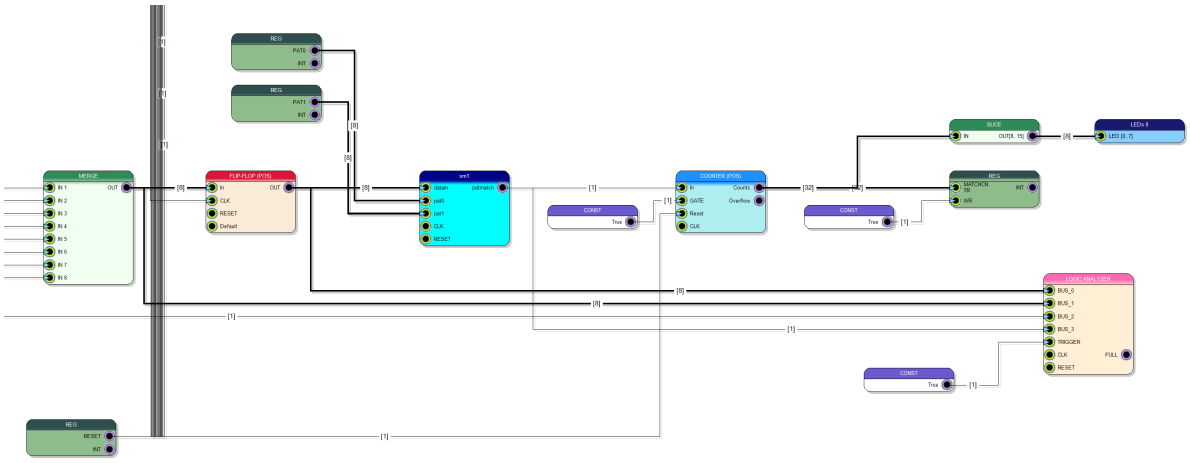
8.4 COUNTERS



Counter modules are direct connected to the digital input of the board. The increase by one every time they are

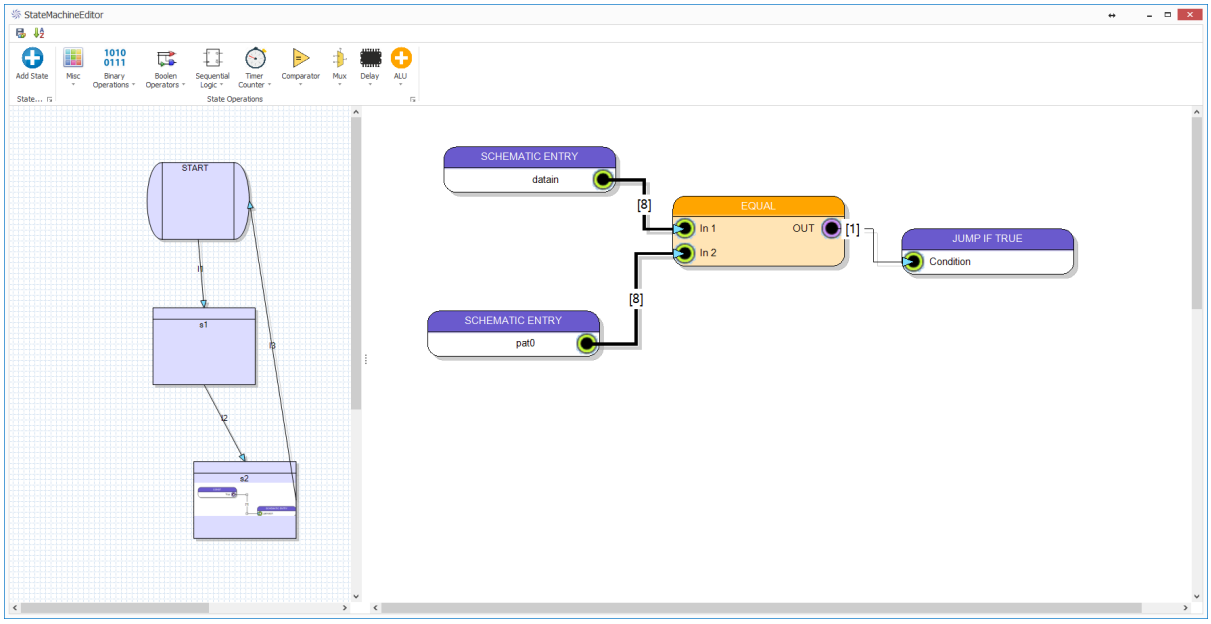
triggered by a transition (HL, LH or BOTH) of the input signals. The 32 bit counter outputs are direct connected to the Memory Mapped registers those are periodically read out by the controlling PC

8.5 PATTERN MATCHING



Heigh of the 16 input signal is merged in a bus in order to be processed by a pattern matching state machine. A validation signal (one of the 16 inputs) is used to capture by a rising edge flip flop the 8 bit sub-bus. The output of the FF is then passed as input of the pattern matching state machine. The state machine is also connected to two registers (PAT0, PAT1) programmed by the PC with the two stages pattern. Each time the pattern is recognized a pulse increment a counter. The counter can be read out by a PC or by the V2495 user leds.

8.6 STATE MACHINE GENERATOR



SCICompiler include a complex state machine generator. The state machine allows the execution of sequential operations in a fully parallel context like inside FPGAs and ASIC. The state machine editor windows is divided in two part. On the left the user design the state diagram with the connections between different states. On the right it is possible to edit:

- The actions perform by the different states

- The conditions to exit from a state

The state actions can be very simple (like set output state) or much more complex like deserialize incoming data. Even the link exit condition can be extremely siple, like wait the transition of an input or more complex like counts the transitions of one or more inputs respecting a maxiumum programmable timeout.

8.7 FIRMWARE REVIEW

The block diagram below shows the full firmware implemented inside the V2495.