GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES Ciemat Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

CALOCUBE T+ROC2 PROGRAMMER'S GUIDE

1. INTRODUCTION

This document contains a description of Calocube T+ROC2 registers and it is intended to be used as programmer guide.

2. HIDRA CHIP SEQUENCE GENERATION.

HIDRA ASIC sequence is stored in a RAM memory as shown in figure 1. Each row (position, address, state …) of this RAM represent the signals needed by the HIDRA ASIC in a given instant of time and each column represent therefore the sequence in time of a particular signal. The time distance of two consecutive rows is $16,\hat{6}\,ns$ corresponding to the time distance of two consecutive rising edges of a 60 MHz clock (distributed to the T+ROC2 from the T+ROC1). A given row contain also the control signal of a multiplexer governing the next row in time. By default (MUX = 0 in the figure) the next row will be consecutive to the present one. If the value of MUX control signals is not the default one, the next position will be selected by the multiplexer accordingly. An external signal (indicated by number 2 in the figure 1) has priority over all the rest of control signals and allow exit of programmed loops. Sequence repetition is also allowed by the appropriate control signal (MUX = 3 in the figure) and the content of register 0x17 containing the times (= TC in figure 1) a given sequence is repeated.

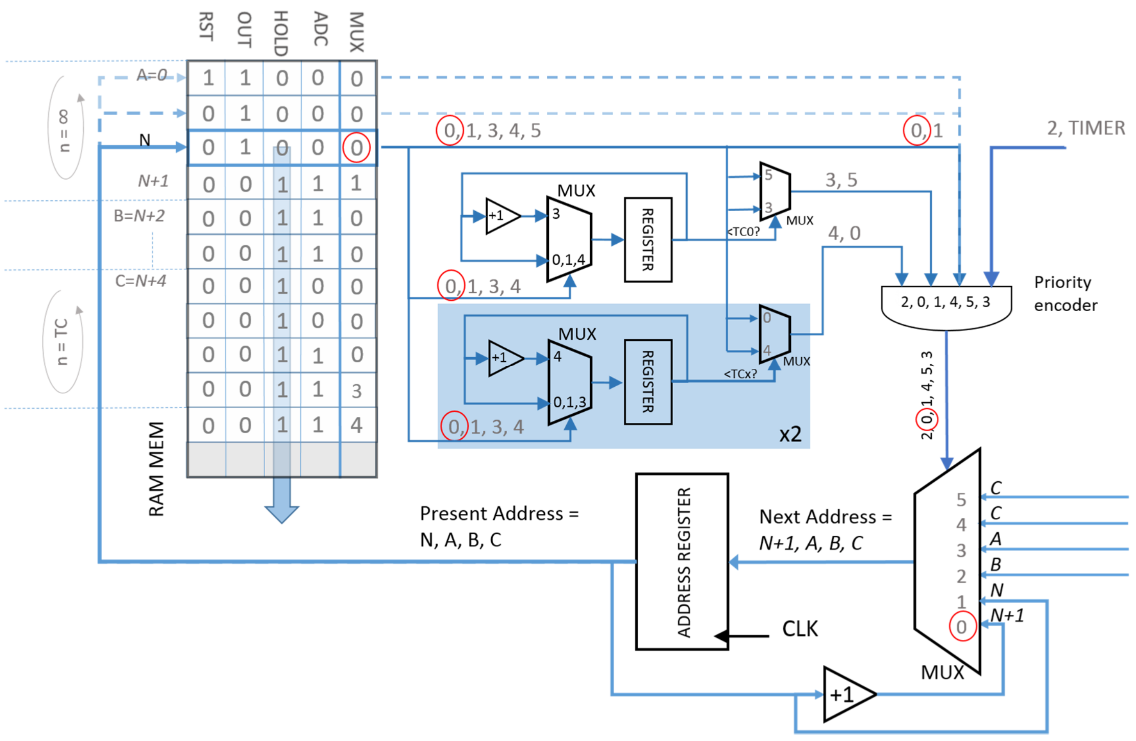


Figure 1 T+ROC2 HIDRA ASIC sequence generation block diagram

RAM memory is up to 4096 position depth and 16 bits wide. RAM address is stored in registers 0x04 (LSB) and register 0x05 (MSB), RAM data is stored at register 0x01 (LSB) and 0x02 (MSB). Correspondence between bits registers and signals is as follow:

REGISTER 0x01	BIT 0	RSTA
	BIT 1	RSTB
	BIT 2	HOLD
	BIT 3	HOLDG
	BIT 4	CAL_EN
	BIT 5	OUT_D
	BIT 6	OUT_RST
	BIT 7	OUT_CLK
REGISTER 0x02	BIT 0	OUT_EN
	BIT 1	SCLK
	BIT 2	SCS
	BIT 3	INC
	BIT 4	12
	BIT 5	13
	BIT 6	EN_TRG/BUSY
	BIT 7	RST_COUNT

Table 1. T+ROC2 RAM data registers.

Control signals are INC, 12, 13, RST_COUNT and the external signal is EXT_TRG (not shown in the list, as far as said previously cannot be set by the RAM bits). If INC is set a loop counter is incremented as many time as set in register 0x17 (TC) and the next address in RAM where the sequence will continue is stored at registers 0x0E (LSB) and 0x0F (MSB). When the number of repetitions at register 0x09 are reached, then the next address position is stored at register 0x0C (LSB) and 0x0D (MSB). If the external signal EXT_TRG is set then the next address is stored at registers 0x0A (LSB) and 0x0B (MSB). If RST_COUNT is set the next address will be 0x0000. If signal 12 (or 13) is set a loop counter is incremented as many time (+1 time) as set in register 0x18 (or 0x19) and the next address in RAM where the sequence will continue is stored at registers 0x10 (or 0x12) (LSB) and 0x11 (or 0x13) (MSB). When the number of repetitions at register 0x18 (or 0x19) are reached the next address position will be the consecutive one.

T+ROC2´s registers can be modified by two means. Through a T+ROC1 serial link access (7500 KBUAD, 8 bits data, 1 start bit , 1 stop bit, no parity) and through an RS232 link access (115 KBAUD, 8 bits data, 1 start bit, 1 stop bit, no parity) .
Any access to modify the content of a T+ROC2 registers is 2 bytes long. First byte represent the register address and second one the content of that register.

T+ROC2 register address are as follow:

Address	R/W	Type	Description
0x00	R/W	REG	Control Register: Bit 0: 0 sequence reset / 1 normal operation. Bit 1: GAIN2SEL. Bit 2: 0 RS232 reset active / 1 RS232 normal operation. Bit 3: 0 external trigger / 1 internal trigger (for test purposes) Bit 4: 0 normal operation / 1 internal trigger counter reset Bit 5: Used for test purposes. Should be 0 during normal operation Bit 6: Used for test purposes. Should be 0 during normal operation Bit 7: 1 configuration / 0 data taking Default register content 0x00.
0x01	R/W	REG	This register stores the **LSB data** to be write into the RAM. Bit 0: RSTA. Bit 1: RSTB. Bit 2: HOLD. Bit 3: HOLD_G. Bit 4: CAL EN.

			Bit 5: OUT_D. Bit 6: OUT_RST. Bit 7: OUT_CLK. Default register content 0x00.
0x02	R/W	REG	This register stores the **MSB data** to be write into the RAM. Bit 0: OUT_EN. Bit 1: SCLK. Bit 2: SCS. Bit 3: INC. Bit 4: 12. Bit 5: 13. Bit 6: EN_TRG/BUSY. Bit 7: RST_COUNT. Default register content 0x00.
0x04	R/W	REG	This register stores the **LSB RAM address** where to write and from where to read. Default register content 0x00.
0x05	R/W	REG	This register stores the **MSB RAM address** where to write and from where to read. Default register content 0x00.
0X06	R/W	REG	It is possible to read the content of most of T+ROC2 registers. This register store the address of the T+ROC2 register to be read. 0x00: Control Register. 0x01: LSB RAM data 0x02: MSB RAM data 0x03: 0x00 0x04: LSB RAM address 0x05: MSB RAM address 0x06: Read selection register 0x07: 0x00 0x08: 0x00 0x09: Number of times a given sequence is repeated (# INC). 0x0A: Jump Address LSB destination in case an Ext TRG is detected. 0x0B: Jump Address MSB destination in case an Ext TRG is detected. 0x0C: Jump Address LSB destination in case of TC. 0x0D: Jump Address MSB destination in case of TC. 0x0E: Jump Address LSB destination in case of INC. 0x0F: Jump Address MSB destination in case of INC. 0x10: Jump Address LSB destination in case of 12. 0x11: Jump Address MSB destination in case of 12. 0x12: Jump Address LSB destination in case of 13. 0x13: Jump Address MSB destination in case of 13 0x14: Jump Address LSB destination in case of 12 and 13. 0x15: Jump Address MSB destination in case of 12 and 13 0x17: Number of times a given sequence is repeated (# INC). 0x18: Number of times a given sequence is repeated (# 12). 0x19: Number of times a given sequence is repeated (# 13). 0x1A: CAL LOAD 0x1B: ST LOAD 0x1C: ST Threshold at HIDRA2_1 0x1D: ST Threshold at HIDRA2_2 0x1E: ST Threshold at HIDRA2_3 0x1F: ST Threshold at HIDRA2_4 0x20: RS232_Enable_TX only accessible through an RS232 access. Default register content 0xFF. 0x21: USB_RS232_Enable_TX only accessible through an RS232 access. Default register content 0x00.

			0x22: TRG_GEN LSB. 0x23: TRG_GEN LSB. 0x24: GAIN HOLD TIMER LSB 0x25: GAIN HOLD TIMER MSB 0x26: HOLD TIMER LSB 0x27: HOLD TIMER MSB 0x2B: Scaler selector. Default register content 0x00. 0x2C: Scaler LSB. Default register content 0x00. 0x2D: Scaler MSB. Default register content 0x00.
0x07	W	REG	Any access to this register followed by any byte content perform a RAM write at RAM address set by registers 0x05 and 0x04 of the data contained in registers 0x01 and 0x02
0x08	W	REG	0x00: When an access to this register is followed by byte 0x00, a RAM read is execute from the address set by registers 0x05 and 0x04. Depending of the content of register 0x06, LSB (if 0x01) or MSB (if 0x02) of RAM data is read. 0x01: When an access to this register is followed by byte 0x01, a register read is performed depending of the content of register 0x06.
0X09	R/W	REG	Number of times a given sequence is repeat. Default register content 0x3F.
0x0A	R/W	REG	If an external trigger is detected jump **destination address LSB.** Default register content 0xB0.
0x0B	R/W	REG	If an external trigger is detected jump **destination address MSB**. Default register content 0x04.
0x0C	R/W	REG	If control bit INC is set and has been detected as many times as set in register 0x09, jump **destination address LSB.** Default register content 0x00.
0x0D	R/W	REG	If control bit INC is set and has been detected as many times as set in register 0x09, jump **destination address MSB.** Default register content 0x00.
0x0E	R/W	REG	If control bit INC is set and has not been detected the number of times set in register 0x09, jump **destination address LSB.** Default register content 0xC0.
0X0F	R/W	REG	If control bit INC is set and has not been detected the number of times set in register 0x09, jump **destination address MSB.** Default register content 0x07.
0x10	R/W	REG	If control bit 12 is set jump **destination address LSB.** Default register content 0x00.
0x11	R/W	REG	If control bit 12 is set jump **destination address MSB.** Default register content 0x00.
0x12	R/W	REG	If control bit 13 is set jump **destination address LSB.** Default register content 0x00.
0x13	R/W	REG	If control bit 13 is set jump **destination address MSB.** Default register content 0x00.
0x14	R/W	REG	If control bit 12 and control bit 13 are set jump **destination address LSB.** Default register content 0x00.
0x15	R/W	REG	If control bit 12 and control bit 13 are set jump **destination address MSB.** Default register content 0x00.
0x17	R/W	*REG*	Number of times a given sequence is repeat. Controlled by signal INC Default register content 0x3F.
0X18	R/W	REG	Number of times a given sequence is repeat. Controlled by signal 12 Default register content 0x3F.
0X19	R/W	REG	Number of times a given sequence is repeat. Controlled by signal13

			Default register content 0x3F.
0x1A	W	REG	CAL MASK LOAD: Bit 0: CAL CLK. Bit 1: CAL RST. 0 reset / 1 normal operation. Bit 2: CAL D Bit 3: CAL EN (0) Bit 4: CAL EN (1) Bit 5: CAL EN (2) Bit 6: CAL EN (3) Bit 7: Not used. Default register content 0x00.
0x1B	W	REG	ST MASK LOAD: Bit 0: ST CLK Bit 1: ST RST 0 reset / 1 normal operation. Bit 2: ST EN Bit 3: ST D(0) Bit 4: ST D(1) Bit 5: ST D(2) Bit 6: ST D(3) Bit 7: Not used. Default register content 0x00.
0x1C	R/W	REG	ST Threshold at HIDRA2_1 the mapping of the bits with ASIC **TBD** Default register content 0x55.
0x1D	R/W	REG	ST Threshold at HIDRA2_2 the mapping of the bits with ASIC **TBD** Default register content 0x55.
0x1E	R/W	REG	ST Threshold at HIDRA2_3 the mapping of the bits with ASIC **TBD** Default register content 0x55.
0x1F	R/W	REG	ST Threshold at HIDRA2_4 the mapping of the bits with ASIC **TBD** Default register content 0x55.
0X20	R/W*	REG	RS232_Enable_TX only accessible through an RS232 access. Default register content 0xFF.
0x21	R/W*	REG	USB_RS232_Enable_TX only accessible through T+ROC1 configuration access. Default register content 0x00.
0X22	R/W	REG	*TRG_GEN LSB. Used to set the frequency of an internal random trigger generator. This generator will produce a trigger pulse every 200ns*(256*MSB + LSB)* RND_GEN. Where RND_GEN is a number randomly generated between 1 and 256 (Used for test purposes).* *Default register content 0xFF.*
0x23	R/W	REG	*TRG_GEN MSB. Used to set the frequency of an internal random trigger generator. This generator will produce a trigger pulse every 200ns*(256*MSB + LSB)* RND_GEN. Where RND_GEN is a number randomly generated between 1 and 256 (Used for test purposes).* *Default register content 0xFF.*
0x24	R/W	REG	GAIN HOLD TIMER LSB.
0x25	R/W	REG	GAIN HOLD TIMER MSB.
0x26	R/W	REG	HOLD TIMER LSB.
0x27	R/W	REG	HOLD TIMER MSB.
0x2B	R/W	REG	**Scaler selection** (0x00-0x7F) Default register content 0x00.
0x2C	R	REG	Selected **scaler LSB.** Default register content 0x00.
0x2D	R	REG	Selected **scaler MSB.** Default register content 0x00.

Table 2. T+ROC2 registers

* All the register are accessible through a T+ROC1 configuration access or a RS232 link (115000 BAUD, 8 bits data, 1 bit stop, no parity, no flow control) access but registers 0x20 and 0x21. Register 0x20 can only be write through a RS232 access and register 0x21 can only be write through a T+ROC1 configuration access. Both of them however can be read either through a T+ROC1 configuration access or RS232 access. Content of registers 0x20 and 0x21 should to be equal in order to get response from that particular T+ROC2 through the RS232 link.

__Examples__

1- Example of how to write (0x02 = MSB_Data, 0x01 = LSB_Data) at position (0x05 = MSB_Add, 0x04 = LSB_Add) in the RAM memory:

```
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
DWORD BytesWritten;
char TxBuffer[256]; // Contains data to write to device

TxBuffer[0] = 0x00;
TxBuffer[1] = 0x01; // is a write at T+ROC1 configuration T+ROC2 register.
TxBuffer[2] = 0x0A;
TxBuffer[3] = 0x00; // register at T+ROC1 # of Byte to be written to T+ROC1
TxBuffer[4] = 0x05;
TxBuffer[5] = 0xMSB_Add; // MSB of the RAM mem address to be written
TxBuffer[6] = 0x04;
TxBuffer[7] = 0xLSB_Add; // LSB of the RAM mem address to be written
TxBuffer[8] = 0x02;
TxBuffer[9] = 0xMSB_Data; // MSB of the data to be written
TxBuffer[10] = 0x01;
TxBuffer[11] = 0xLSB_Data; // LSB of the data to be written
TxBuffer[12] = 0x07;
TxBuffer[13] = 0x00;

// The content of registers 0x02,0x01 are written in the RAM address at registers 0x05,0x04.
// We choose to load the MSB registers before the LSB but the order can be any the write do  // not take
place until the 0x07 , 0x00 is executed at T+ROC2 .

ftStatus = FT_Open(0, &ftHandle);

if(ftStatus != FT_OK) {// FT_Open failed
        return;
}

ftStatus = FT_Write(ftHandle, TxBuffer, sizeof(TxBuffer), &BytesWritten);

// sizeof(TxBuffer) = 0x0E

if (ftStatus == FT_OK) {
        // FT_Write OK
}
else {
        // FT_Write Failed
}
FT_Close(ftHandle);
```

Comments: No enable of the configuration links, has been made as far as by default all the configuration links are enabled at T+ROC1.

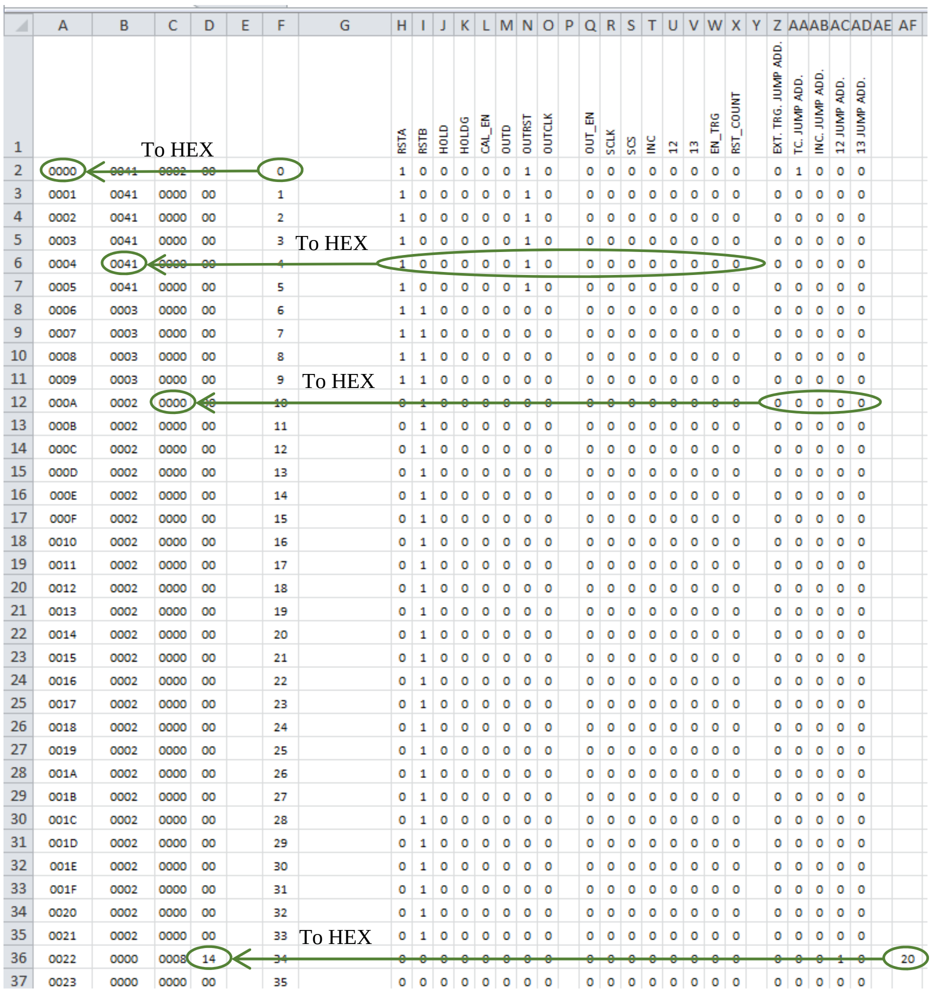
2- Example of how to read from position (0xMSB_Add ,0xLSB_Add) in the RAM memory:

```
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
DWORD EventDWord;
DWORD TxBytes;
DWORD RxBytes;
DWORD BytesReceived;
DWORD BytesWritten;
char TxBuffer[256];
char RxBuffer[256];

TxBuffer[0] = 0x00; // register at T+ROC1
TxBuffer[1] = 0x01; // register at T+ROC1
TxBuffer[2] = 0x0C; // register at T+ROC1 is a write of
TxBuffer[3] = 0x00; // register at T+ROC1 # of Bytes to write to T+ROC1
TxBuffer[4] = 0x05;
TxBuffer[5] = 0xMSB_Add; // most signifiant byte of the RAM address to write
TxBuffer[6] = 0x04;
TxBuffer[7] = 0xLSB_Add; // Less signifiant byte of the RAM address to write
TxBuffer[8] = 0x06;
TxBuffer[9] = 0x01;
TxBuffer[10] = 0x08;
TxBuffer[11] = 0x00;
TxBuffer[12] = 0x06;
TxBuffer[13] = 0x02;
TxBuffer[14] = 0x08;
TxBuffer[15] = 0x00;

ftStatus = FT_Open(0, &ftHandle);

if(ftStatus != FT_OK) { // FT_Open failed
return;
}

ftStatus = FT_Write(ftHandle, TxBuffer, sizeof(TxBuffer), &BytesWritten);
// sizeof(TxBuffer) = 0x10

if (ftStatus == FT_OK) {
// FT_Write OK
}
else {
// FT_Write Failed
FT_Close(ftHandle);
return;
}

TxBuffer[0] = 0x00; // register at T+ROC1
TxBuffer[1] = 0xC1; // register at T+ROC1 is a read (0x80) of two bytes through link 0 (0x41)
TxBuffer[2] = 0x02; // register at T+ROC1 is a read of TxBuffer[3] bytes
TxBuffer[3] = 0x00; // register at T+ROC1 # of Byte to be written to T+ROC1

ftStatus = FT_Write(ftHandle, TxBuffer, sizeof(TxBuffer), &BytesWritten);
// sizeof(TxBuffer) is now = 0x4

if (ftStatus == FT_OK) {
// FT_Write OK
}
else {
// FT_Write Failed
FT_Close(ftHandle);
return;
}

ftStatus = FT_Read(ftHandle,RxBuffer,RxBytes,&BytesReceived);
// where RxBytes is in this example is = 0x02

if (ftStatus == FT_OK) {
// FT_Read OK
}
else {
// FT_Read Failed
FT_Close(ftHandle);
Return;
}
// data stores at RAM address = (MSB_Add, LSB_Add)
// are now in RxBuffer[0]= LSB_Data and RxBuffer[1]= MSB_Data

FT_Close(ftHandle);
```

Comments: No enable of any configuration link has been made in this example, it is assume therefore that has been made previously. To perform a read on the RAM memory is preceded by two writes access.

3. LabView CONFIGURATION PROGRAM.

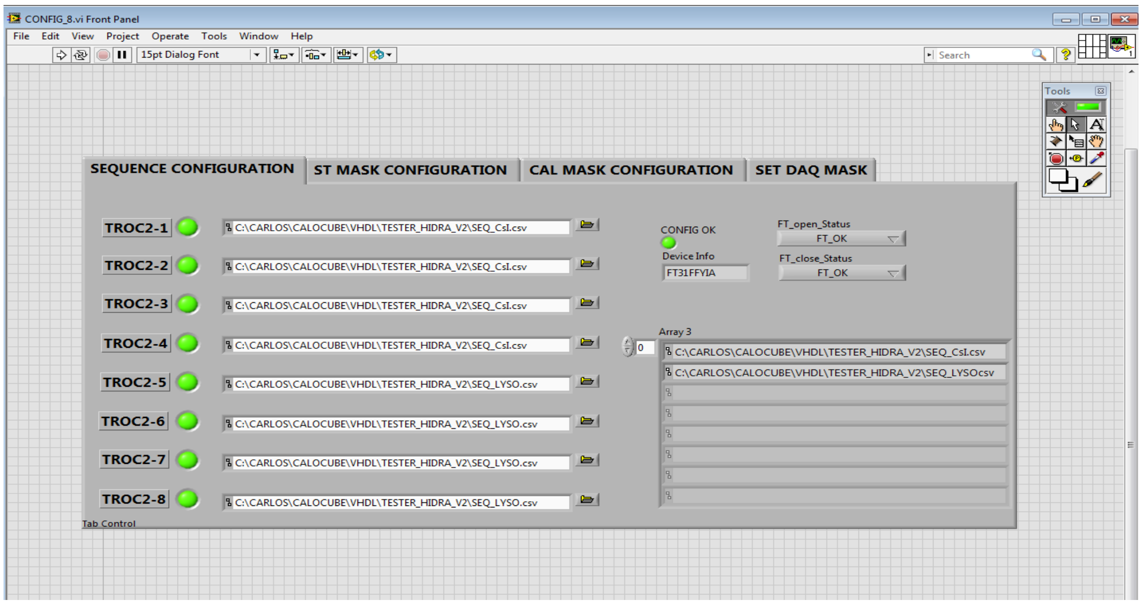
In order to easy the configuration process a LabView program has been developed allowing to load the sequence, the self- trigger mask and calibration mask from a .CSV file.
Selection of a .CSV format for sequence edition and generation allows the use of common tools like EXCEL or OPEN OFFICE. Tree column are then generated, the first one represent the RAM hexadecimal address (to be load on registers 0x04 and 0x05 of T+ROC2), second one represent the status of control signals according to Table 1 of this document and the corresponding columns in the file (to be load on registers 0x01 and 0x02 of T+ROC2) third one, if different of zero, represent the jump destination when a jump condition is reached and the fourth one represent the vaslue of register 0x09, 0x18 and 0x19 depweding on the value of the third column.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1								RSTA	RSTB	HOLD	HOLDG	CAL_EN	OUTD	OUTRST	OUTCLK		OUT_EN	SCLK	SCS	INC	I2	I3	EN_TRG	RST_COUNT		EXT. TRG. JUMP ADD.	TC. JUMP ADD.	INC. JUMP ADD.	I2 JUMP ADD.	I3 JUMP ADD.		
2	0000	~~0041~~	~~0002~~	~~00~~		0		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	1	0	0	0		
3	0001	0041	0000	00		1		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
4	0002	0041	0000	00		2		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
5	0003	0041	0000	00		3		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
6	0004	0041	~~0000~~	~~00~~		~~4~~		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
7	0005	0041	0000	00		5		1	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
8	0006	0003	0000	00		6		1	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
9	0007	0003	0000	00		7		1	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
10	0008	0003	0000	00		8		1	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
11	0009	0003	0000	00		9		1	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
12	000A	0002	0000	~~00~~		~~10~~		~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~		~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~		0	0	0	0	0		
13	000B	0002	0000	00		11		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
14	000C	0002	0000	00		12		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
15	000D	0002	0000	00		13		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
16	000E	0002	0000	00		14		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
17	000F	0002	0000	00		15		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
18	0010	0002	0000	00		16		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
19	0011	0002	0000	00		17		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
20	0012	0002	0000	00		18		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
21	0013	0002	0000	00		19		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
22	0014	0002	0000	00		20		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
23	0015	0002	0000	00		21		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
24	0016	0002	0000	00		22		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
25	0017	0002	0000	00		23		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
26	0018	0002	0000	00		24		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
27	0019	0002	0000	00		25		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
28	001A	0002	0000	00		26		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
29	001B	0002	0000	00		27		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
30	001C	0002	0000	00		28		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
31	001D	0002	0000	00		29		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
32	001E	0002	0000	00		30		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
33	001F	0002	0000	00		31		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
34	0020	0002	0000	00		32		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
35	0021	0002	0000	00		33		0	1	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		
36	0022	0000	0008	14		~~34~~		~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~		~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~		~~0~~	~~0~~	~~0~~	~~1~~	~~0~~		20
37	0023	0000	0000	00		35		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0		

Figure 2. Sequence .CSV file format.

Examples of this files can be found under the following link:
https://drive.google.com/drive/folders/10x0ShipqAlj7d-CmJNDGhZY3WfP3qGPa?usp=sharing
Program main tab can be seen at figure 3.



SEQUENCE CONFIGURATION | ST MASK CONFIGURATION | CAL MASK CONFIGURATION | SET DAQ MASK

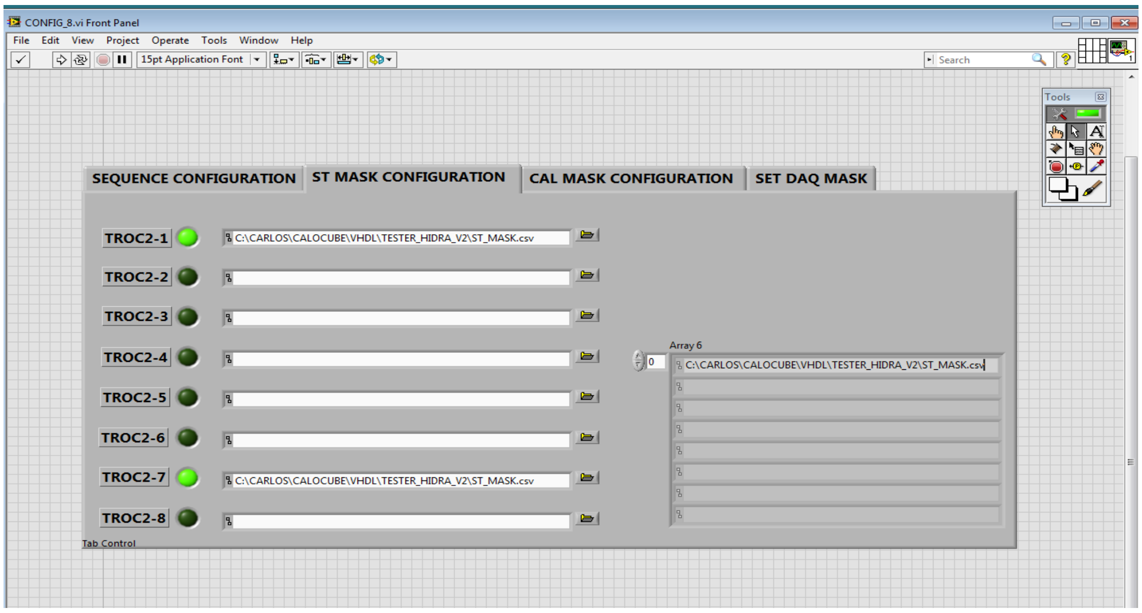
TROC2-1	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_CsI.csv
TROC2-2	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_CsI.csv
TROC2-3	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_CsI.csv
TROC2-4	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_CsI.csv
TROC2-5	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_LYSO.csv
TROC2-6	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_LYSO.csv
TROC2-7	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_LYSO.csv
TROC2-8	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_LYSO.csv

CONFIG OK
Device Info: FT31FFYIA
FT_open_Status: FT_OK
FT_close_Status: FT_OK

Array 3
0
C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_CsI.csv
C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\SEQ_LYSOcsv

Figure 3. LabView configuration programm main tab.

Up to eight T+ROC2 board can be connected to the same T+ROC1, either for configuration or for data taking. Up to eight T+ROC2 can be selected then ligthing the corresponding LED control. On a by T+ROC2 basis the sequence file it is selected.
Similar tab exist for self-trigger and calibration configuration.



SEQUENCE CONFIGURATION | ST MASK CONFIGURATION | CAL MASK CONFIGURATION | SET DAQ MASK

TROC2-1	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\ST_MASK.csv
TROC2-2	
TROC2-3	
TROC2-4	
TROC2-5	
TROC2-6	
TROC2-7	C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\ST_MASK.csv
TROC2-8	

Array 6
0
C:\CARLOS\CALOCUBE\VHDL\TESTER_HIDRA_V2\ST_MASK.csv

Figure 4. LabView configuration programm ST_MASK tab.

3.1 Configuration in Linux

The LabVIEW configuration program has been developed using the FTD2XX driver in synchronous FIFO mode. “FT calls” designed for the D2xx driver are running under Linux CENTOS7 or Windows7. The configuration program has been tested under these operating systems. Labview 2017 version has been downloaded directly from the NI webpage. FTDI has two types of drivers for all supported operating systems. These are the virtual COM port driver (VCP) and the D2XX API driver. For the installation of the D2XX linux driver, it is recommended to download the suitable Linux D2XX driver of x86 – for 32-bit IA-32 CPUs from the FTDI D2XX driver webpage. The driver files are contained in a tar gzip file: e.g. gunzip libftd2xx-i386-1.4.8.tar.gz and tar –xvf libftd2xx-i386-1.4.8.tar

Procedure:

1) All driver files need to be copied and symbolic links created using the Linux sudo command for root permissions:
sudo cp /releases/build/lib /usr/local/lib*

2) Make the following symbolic links and permission modifications in /usr/local/lib:
cd /usr/local/lib
sudo ln –s libftd2xx.so.x libftd2xx.so
sudo chmod 0755 libftd2xx.so.1.4.8

3) The symbolic link is used to select a default driver file. Any program can be linked against a specific version of the library by using a version numbered library file. Make sure the following dependences are correctly installed:
/etc/udev/rules.d/99-libftdi.rules
/usr/lib/libftdi.so.1
/usr/lib/ftdi.so.1.18.0

4) Add user to the user group
Usermod –G 984 nameuser
Adduser –G 5 nameuser less /etc/groups
Add user nameuser dialout tty

5) Plug in the FTDI based device. VCP driver need to be removed for its particular use:
sudo rmmod ftdi sio.

References

1. Xilinx,“ UG380 - Spartan-6 FPGA” Configuration User Guide (ver 2.10). DS162 - Spartan-6 FPGA Data Sheet: DC and Switching Characteristics (ver 3.1.1). DS160 - Spartan-6 Family Overview (ver 2.0).
2. Xilinx,“ UG470 - 7 Series FPGAs Configuration User Guide,” ver 1.13.1, (August 21th, 2018).
3. Gianluigi Zampa, “HiDRA2: High Dynamic Range Amplifier frontend ASIC,” HiDRA2 DATASHEET, (March, 2018).
4. FTDI,“FTDI drivers, Document Reference No.: FT 000723 Clearance No.: FTDI 302,” (July 6th, 2017).
5. Analog Devices, “16-Bit, 500 kSPS PulSAR ADC in MSOP,” Datasheet.
6. G.Martínez, et al. “TROC1 manual,” January, 2019. Firmware version x0C0B.