CMS DAQ

Zynq Ultrascale+

Software Reference Manual

INFN Padova

By Lorenzo Castellani

The interface consists of a web page that implements a series of forms for setting the operating parameters, access to all the registers of the various chips and a WebSocket Server (RFC 6455) which responds on port 4444.
LabView library to communicate with the server is available
https://www2.pd.infn.it/~caste/pub/WebSockets.zip
Also there are the following servers, one for RO data on port 3333, one for TRG data on port 3334, and one for MINICRATE communication on port 28888, primary port use CCB address 0, or if different from 0 use secondary port .
RO link led are on only if channel is enabled and locked, TRG link led are on only if locked, enable are not effect. If internal PLL lose lock all link are reset so if lose lock auto clear is enabled and lock is lost for 1 second the delay is automatically restored to the last value to try restore link lock.

Spare IO definition
IN_NIM1 is TTC input, must be selected by command
IN_NIM2 is L1A input, must be selected by command
IN_NIM3 is EXT1 TRIG input
IN_NIM4 is EXT2 TRIG input

OUT_NIM1 is CLK out
OUT_NIM2 is TTC out
OUT_NIM3 is L1A out
OUT_NIM4 is LHC CLK out

Preliminary commands implemented.

Binary commands accepted by server are:

Commands implemented with WebSocket opt code = binary frame.
The format of the frames: [cmd] [arg1] .... [argn].
The first byte represents the command followed by the arguments.
The following structures describe the arguments for each command, UINT32 represents an integer 32bit (4 bytes) in little-Endian format, FLOAT64 an 8byte floating point in little-Endian format.

Operation, code 0x01

Manual operation for data link phase adjust and enable and disable channels. Enable/Disable channel for trigger are :
TRG Channel 0 Global enable trigger;

TRG Channel 1 L1A enable.
TRG Channel 2-7 unused.

```
BYTE cmd;
BYTE interf;      //select interface 0=RO, 1=TRG
BYTE op;          //0=dec-delay, 1=inc-delay, 2=inc-pos, 3=clr-lost, 4=enable, 5 disable
BYTE ch;          //channel 0-7
```

Return : the status frame, see Status command

Auto Clear, code 0x04

Activate/disactivate the auto clear lost lock flag.

```
BYTE cmd;
BYTE interf;      //select interface 0=RO, 1=TRG
BYTE ch;          //channel 0-7
BYTE value;       //value 0-1
```

Return : the same data sent.

Reset lost lock counter, code 0x05

```
BYTE cmd;
BYTE interf;      //select interface 0=RO, 1=TRG
BYTE ch;          //channel 0-7
```

Return : the same data sent..

Set Val and Mask, code 0x03

```
BYTE cmd;
BYTE interf;      //select interface 0=RO, 1=TRG
INT32 val[4];     //format 0x0BBB0AAA, AAA= 10bit value CH0….. CH7
INT32 mask[4]; //format 0x0BBB0AAA, AAA= 10bit mask CH0….. CH7
```

Return :

```
BYTE cmd;         //0x03
BYTE interf;      //select interface 0=RO, 1=TRG
INT32 val[4];     //format 0x0BBB0AAA, AAA= 10bit value CH0….. CH7
```

INT32 mask[4]; //format 0x0BBB0AAA, AAA= 10bit mask CH0….. CH7

INT32 autoclear[8];	//0 or 1 auto clear state

INT32 enable; //channel enabled

Get Val and Mask, code 0x83

BYTE cmd;

BYTE interf;	//select interface 0=RO, 1=TRG

Return :

BYTE cmd;	//0x03

BYTE interf;	//select interface 0=RO, 1=TRG

INT32 val[4];	//format 0x0BBB0AAA, AAA= 10bit value CH0….. CH7

INT32 mask[4]; //format 0x0BBB0AAA, AAA= 10bit mask CH0….. CH7

INT32 autoclear[8];	//0 or 1 auto clear

INT32 enable; //channel enabled

Trigger Previous words, code 0x08

Set the number of word saved in the FIFO before the trigger.

BYTE cmd;

BYTE val;	//number of word saved in the FIFO before trigger

Return :

BYTE cmd;	//0x0E

UINT32 cfg;	//trigger config. Format: 0xFOSDDAAPP

//PP 8bit number of previous word

//AA 8bit number of after word

//DD 8bit delay

//S 3bit select input trigger

//O 4bit select output signal

//F msbit 1=fall edge, 0=rise edge

Trigger After words, code 0x09

Set the number of word saved in the FIFO after the trigger.

```
BYTE cmd;
BYTE val;       //number of word saved in the FIFO after the trigger
```

Return :

```
BYTE cmd;       //0x0E
UINT32 cfg;      //word trigger config. Format: 0xFOSDDAAPP
                //PP 8bit number of previous word
                //AA 8bit number of after word
                //DD 8bit delay
                //S 3bit select input trigger
                //O 4bit select output signal
                //F msbit 1=fall edge, 0=rise edge
```

Trigger Delay, code 0x0A

Set trigger delay to start save word on the FIFO.

```
BYTE cmd;
BYTE val;       //Delay in CLK unit (1/3 BX  = 120MHz)
```

Return :

```
BYTE cmd;       //0x0E
UINT32 cfg;      //word trigger config. Format: 0xFOSDDAAPP
                //PP 8bit number of previous word
                //AA 8bit number of after word
                //DD 8bit delay
                //S 3bit select input trigger
                //O 4bit select output signal
                //F msbit 1=fall edge, 0=rise edge
```

Select input Trigger, code 0x0B

Set input trigger.

BYTE cmd;

BYTE val; //0=TRGOUT, 1=CALup, 2=CCB_RDYup, 3=CALdw, 4=CCB_RDYdw, 5=trigger(0), 6=trigger(1), 7=random trigger generator

Return :

BYTE cmd; //0x0E

UINT32 cfg; //word trigger config. Format: 0xFOSDDAAPP

//PP 8bit number of previous word

//AA 8bit number of after word

//DD 8bit delay

//S 3bit select input trigger

//O 4bit select output signal

//F msbit 1=fall edge, 0=rise edge

Select output L1A, code 0x25

Select signal ouput on LEMO L1A.

BYTE cmd;

BYTE val; //0=L1A 1=TRGOUT, 2=CALup, 3=CCB_RDYup, 4=CALdw, 5=CCB_RDYdw, 6=trigger(0), 7=trigger(1), 8=random trigger generator

Return :

BYTE cmd; //0x0E

UINT32 cfg; //word trigger config. Format: 0xFOSDDAAPP

//PP 8bit number of previous word

//AA 8bit number of after word

//DD 8bit delay

//S 3bit select input trigger

//O 4bit select output signal

//F msbit 1=fall edge, 0=rise edge

Rate Random Trigger Generator, code 0x24

Set input trigger.

```
BYTE cmd;
BYTE val;          // rate = (2^val) * 25ns / 2
```

Return :

```
BYTE cmd;          //0x24
BYTE val;
```

Read Rate Random Trigger Generator, code 0xA4

Set input trigger.

```
BYTE cmd;
```

Return :

```
BYTE cmd;          //0x24
BYTE val;          // rate = (2^val) * 25ns / 2
```

Set edge Trigger, code 0x14

Set input trigger edge.

```
BYTE cmd;
BYTE val;          //1=fall edge, 0= ris edge
```

Return :

```
BYTE cmd;          //0x0E
UINT32 cfg;         //word trigger config. Format: 0xFOSDDAAPP
                   //PP 8bit number of previous word
                   //AA 8bit number of after word
                   //DD 8bit delay
                   //S 3bit select input trigger
                   //O 4bit select output signal
                   //F msbit 1=fall edge, 0=rise edge
```

TTC Orbit, code 0x0C

Set TTC orbit

```
BYTE cmd;
BYTE val;          //BX orbit = (val+1)*44; LHC = 3564 bx
```

Return :

```
BYTE cmd;          //0x0F
UINT32 cfg;         //word trigger config. Format: 0x0000EEOO
                   //OO 8bit number orbit/44
                   //EE 1bit enable
```

TTC Orbit enable, code 0x0D

Set TTC orbit enable

```
BYTE cmd;
BYTE val;          //ON/OFF
```

Return :

```
BYTE cmd;          //0x0F
UINT32 cfg;         //word trigger config. Format: 0x0000EEOO
                   // OO 8bit number orbit/44
                   //EE 1bit enable
```

CFG read, code 0x90

```
BYTE cmd;
BYTE interf;       //0=RO, 1=TRG
```

Return :

```
BYTE cmd;          //0x10
BYTE interf;       //0=RO, 1=TRG
UINT32 cfg;         //word interface config. Format: 0x0000EEOO for RO and 0xFOSDDAAPP
                   //for trigger
```

UINT32 cfg1;	//RO 0x00000MDD. DD L1A delay, M bit0=TTCinput Mbit2=L1Ainput

//TRG dead time

UINT32 cfg2;	// RO not used

// TRG num trigger

Dead time , code 0x15

BYTE cmd;

UINT32 dead;	//trigger dead time in clk of 120MHz

Return :

BYTE cmd;	//0x15

UINT32 dead;	//trigger dead time in clk of 120MHz

Set Number of Trigger , code 0x16

BYTE cmd;

UINT32 numtrg;	//0=trigger counter disabled, the number of trigger to acquire, use TRB Disable/Enable bit 0 to rearm counter

Return :

BYTE cmd;	//0x16

UINT32 numtrg;	//the number of trigger to acquire

Read Dead time , code 0x95

BYTE cmd;

Return :

BYTE cmd;	//0x15

UINT32 dead;	//trigger dead time in clk of 120MHz

Read Number of Trigger , code 0x96

BYTE cmd;

Return :

BYTE cmd;	//0x16

UINT32 numtrg;	//the number of trigger to acquire

Set L1A delay , code 0x19

```
BYTE cmd;
BYTE delay;              //num clock (40MHz) delay
```

Return :

```
BYTE cmd;       //0x19
BYTE delay;     //delay
```

Use TTC input , code 0x1A

```
BYTE cmd;
BYTE input;              //1= use NIM1 input as TTC optic
```

Return :

```
BYTE cmd;       //0x1A
BYTE input;     // input
```

Read Use TTC input , code 0x9A

```
BYTE cmd;
```

Return :

```
BYTE cmd;       //0x1A
BYTE input;     // TTC input
```

Use L1A input , code 0x1B

```
BYTE cmd;
BYTE input;              //1= use NIM2 input as L1A to TTC
```

Return :

```
BYTE cmd;       //0x1B
BYTE input;     // input
```

Read Use L1A input , code 0x9B

```
BYTE cmd;
```

Return :

```
BYTE cmd;       //0x1B
BYTE input;     // L1A input
```

TTC broadcast, code 0x11

Set TTC broadcast cmd

BYTE cmd;	
BYTE val;	//TTC broadcast value
Return :	
BYTE cmd;	//0x11
BYTE val;	//TTC broadcast value

Link scan, code 0x06

Find and set delay for the data link

BYTE cmd;	
BYTE interf;	//select interface 0=RO, 1=TRG
Return :	

```
0x07,<byte Interface>,<uint32 Totalsize>,
<uint32 Namesize>,<Cstring>,
<uint32 Datasize>,<Data>,
<uint32 Namesize>,<Cstring>,
<uint32 Datasize>,<Data>,
…
…
<uint32 Maxrecord>
```

Interface:	the interface number 0=RO 1=TRB
Totalsize:	is the size in byte of data transferred, tag 0x07 included, in little Endian format.
Namesize:	is the size in byte of the string name that identified the data, end string included (char=0), in little Endian format.
Cstring:	is the byte array contained the name of data.

Datasize:	is the data size in byte.
Data:	is the vector array. Vector_0 [x, y], Vector_1[x, y] …..
	x and y are in double float precision (8 byte) in little Endian format.
Maxrecord:	is the maximum number of vectors return in data.

Status, code 0x82

Get the links status, if scanning is running return also scan data frame

```
BYTE cmd;
BYTE interf;      // select interface 0=RO, 1=TRG
```

Return :

```
BYTE cmd;         //0x02
BYTE interf;      //0=RO, 1=TRG
INT32 spydata[8];       //format 0xLVVVPDDD, L bit 31=Lock, bit 30=LoseLock, VVV 12bit value, P
                        4bit position, DDD 12bit delay
INT32 ber[2];     //0xDDCCBBAA 0xHHGGFFEE,  AA=CH0, ….HH=CH7
INT32 nLose[8]; //the 8 channel counter
```

LinkAuto, code 0x12

Tune the Minicrate primary port link

```
BYTE cmd;
```

Return :

```
BYTE cmd;         //0x13
INT32 offset;     // offset
INT32 ampl;       // amplitude
```

Save, code 0x17

Save the link delay

```
BYTE cmd;
```

Return :

```
BYTE cmd;         //0x17
```

Rate, code 0x98

Read the trigger rate

```
BYTE cmd;
```

Return :

```
BYTE cmd;        //0x18
INT32 rate;      // rate
```

Extern Clock, code 0x1C

Set external clock input

```
BYTE cmd;
BYTE ext;        //1=use external clock input, 0=use internal clock
```

Return :

```
BYTE cmd;        //0x1C
BYTE pllstatus;  //0x00 = OK pll locked
```

Si5338 PLL status and Xilinx Pll, code 0x9D

Get Si5338 PLL status

```
BYTE cmd;
BYTE reset;      //reset nLoseLock;
```

Return :

```
BYTE cmd;        //0x1D
BYTE pllstatus;  //0x00 = OK pll locked
INT32 nLoseLock;          // Lose lock number
BYTE xpllstatus;  //Xilinx pll, bit0 = lose lock, bit1= locked. lose lock automatically clear
```

Temperature, code 0x9E

Get temperature sensors

```
BYTE cmd;
```

Return :

```
BYTE cmd;         //0x1E
FLOAT32 pl;       //Programmable Logic Temperature
FLOAT32 ps;       //Processor Syntem Temperature
FLOAT32 rem;      //Remote Temperature
FLOAT32 phy;      //Ethernet PHY Temperature
INT32   numsensor;      //number of external sensors


INT32   id_L;     // external sensor code ID lsb
INT32   id_H;     // external sensor code ID msb
FLOAT32 t;        // Temperature
.
.
INT32   id_L;     // external sensor code ID lsb
INT32   id_H;     // external sensor code ID msb
FLOAT32 t;        // Temperature
```

Find Temperature Sensor, code 0x9F

```
BYTE cmd;
```

Return :

```
BYTE cmd;         //0x1F
INT32   numsensor;      //number of external sensors


INT32   id_L;     // external sensor code ID lsb
INT32   id_H;     // external sensor code ID msb
.
.
INT32   id_L;     // external sensor code ID lsb
INT32   id_H;     // external sensor code ID msb
```

I2C, code 0x20

BYTE cmd;

BYTE data;

BYTE ctrl; bit0=start, bit1=stop, bit2=read, bit3=ackn, bit4=abort. Bit3 and 4 are ignored

.

.

BYTE data;

BYTE ctrl;

Return :

BYTE cmd; //0x20

BYTE data;

BYTE err;

.

.

BYTE data;

BYTE err;

Max in DDR Buffer, code 0xA1

BYTE cmd;

BYTE interf; // select interface 0=RO, 1=TRG

Return :

BYTE cmd; //0x21

BYTE interf; // select interface 0=RO, 1=TRG

INT32 size; // maximum size reached by the DDR buffer since the last reading

RO rate, code 0xA2

BYTE cmd;

Return :

BYTE cmd;   //0x22

INT32 rate;   // RO data rate in byte

TP Generator, code 0x23

BYTE cmd;

INT32 TPrate;   //TP rate in milliseconds

INT32 TPnum;   //numbers of Test Pulse

INT32 ADVnum;   //numbers of sequence Advance

Return :

BYTE cmd;   //0x23

INT32 TPrate;   //TP rate in milliseconds

INT32 TPnum;   //numbers of Test Pulse

INT32 ADVnum;   //numbers of sequence Advance

Return data on error , code 0xFF:

BYTE cmd;   //0xFF

INT32 errorcode; // Error code

Error code:

-1	Not Authorized
-22	Invalid Value
-5	I/O Error
-9	Unknow command (connection will be closed)
-2	no such file
-13	Permission denied
-16	Busy

See linux c/c++ error base for undefined number (errno-base.h)

TEXT commands accepted by server are:

Temperature?	Return the temperature
FindSensor?	Find external sensor and return the sensor id code
Rate?	Return the L1A rate
RORate?	Return the RO data rate in byte
Version?	Return the App version string
Save	Save link delay
Async	Set async data mode for scan waveform
notAsync	Disable async data mode (no data return at end of scan).
Inclkstep:<inc>	inc=0 decrement internal frequency 1ppm, inc=1 increment frequency 1ppm
Extclkstep:<inc>	inc=0 decrement external frequency 1ppm, inc=1 increment frequency 1ppm
Inclkreset	reset internal frequency to default value.
Extclkreset	reset extern frequency to default value.

Readout Data Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	TDC MSG			TDC ID				DATA																							
Zynq	TDC MSG			ROB			TDC ID		DATA																							

Zynq added WORDS message:

Parity Error

The word with same ROB ID that following the parity error message is corrupted, bit3 flag indicate word bits 31-24 and so on

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zynq	1	1	0	ROB			TDC ID		0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	parity error flags			

Lost Data

The FIFO has lost data (FIFO full)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zynq	1	1	0	ROB			TDC ID		0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Unlock ROB Link

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zynq	1	1	0	ROB			0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

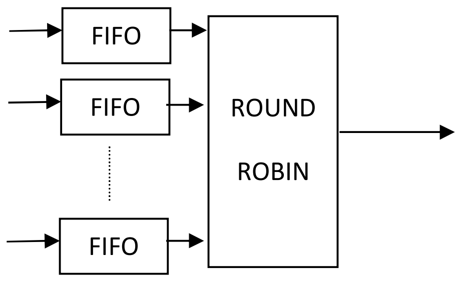
DUMMY (used for draining data from DMA cache)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zynq	1	1	0	000			0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

L1A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zynq	1	1	0	000			0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The output data from board are not ROB ordered, the data are extracted from links FIFO by a round robin logic to yield maximum performance.



Trigger Data Format

WORD 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRG(1:0)		CCB Ready	CAL	Parity	BC Reset	F/S	Q(2:0)			Bending up(9:0)										Radial up(11:0)											

WORD 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Q(1:0)		Bending down(9:0)										Radial down(11:0)												BC(1:0)		TRG(7:2)					

WORD 2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRG	LAST	0	0	0	0	0	0	Link Lock								TRG Out	BC Reset	HL(7:0)								CCB Ready	CAL	Parity	BC Reset	F/S	Q2

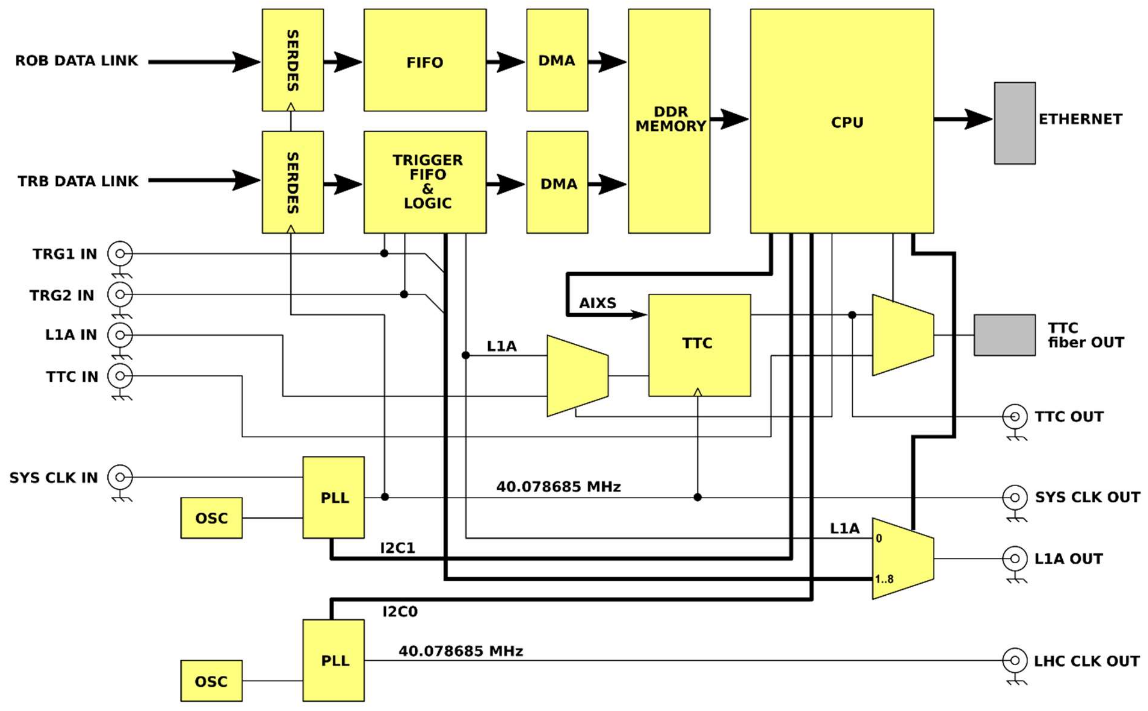
TRG is the ZYNQ selected trigger signal, the number of 96bit words before the trigger can vary by 1.

LAST indicate the last event word.

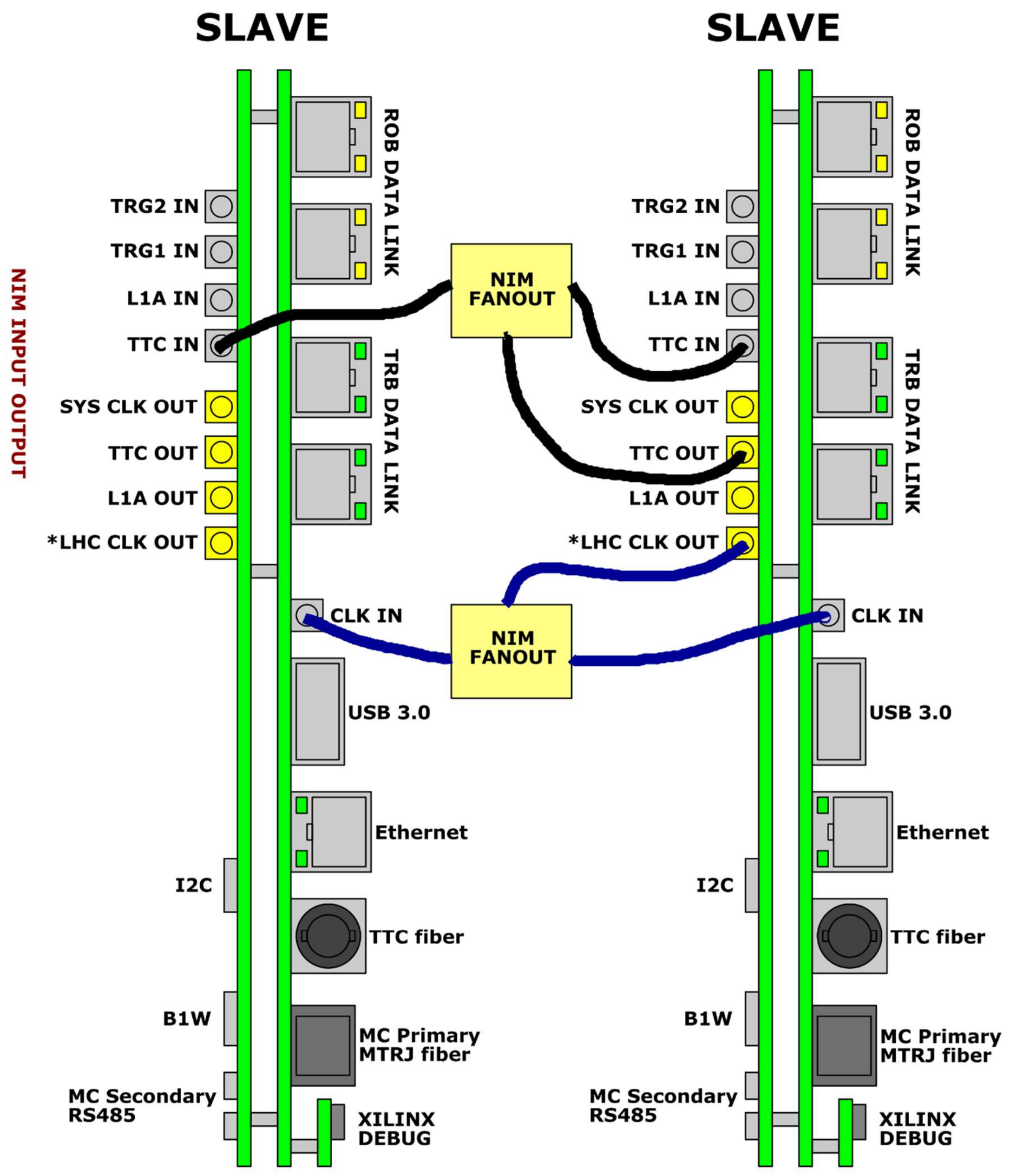
DUMMY word is 0xFFFFFFFF, it is used for draining data from DMA cache

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CMS DAQ Zynq Ultrascale+ BLOCK DIAGRAM



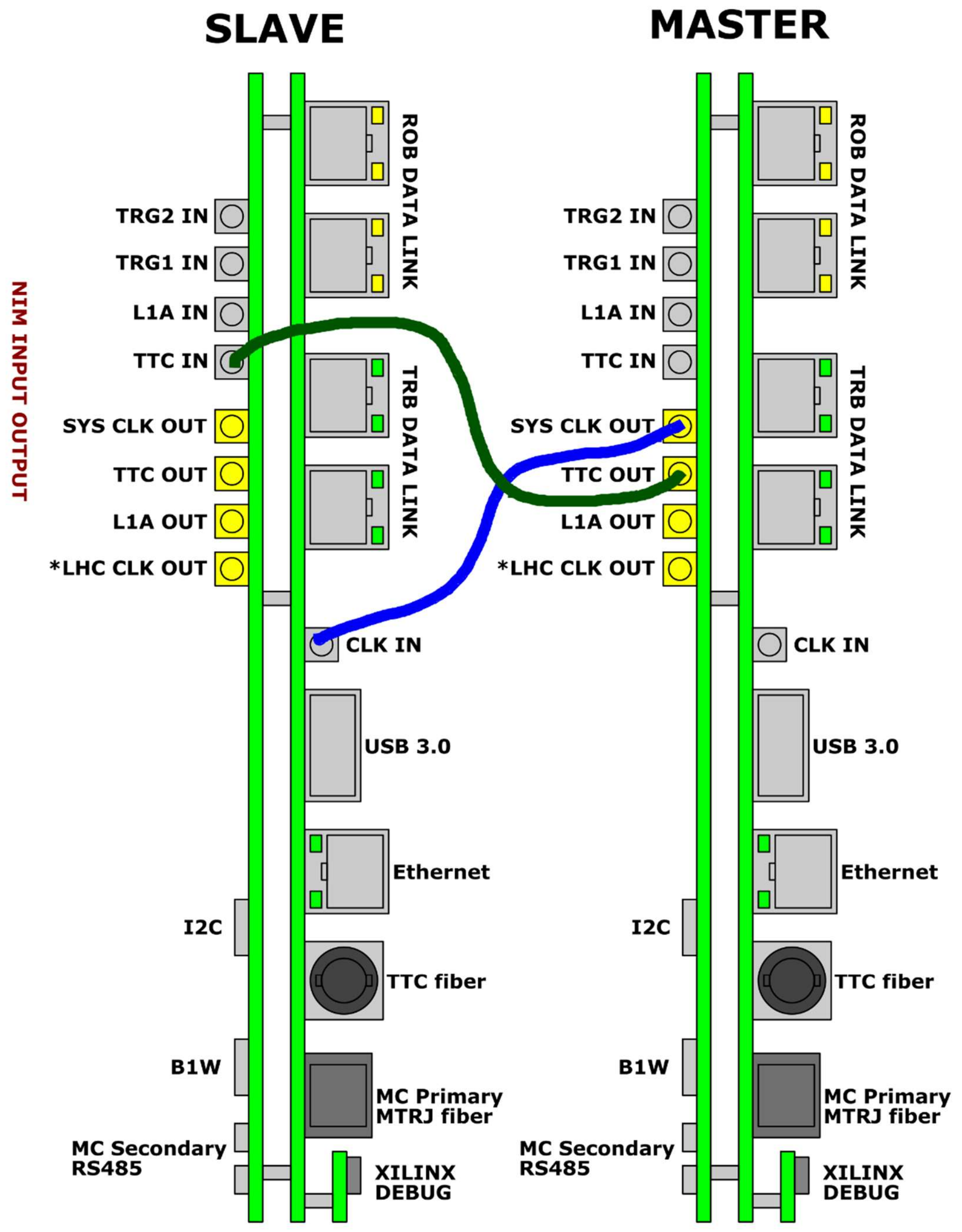
CMS DAQ Zynq Ultrascale+



Slave cmd:
0x1C 0x01 (use external CLK)
0x1A 0x01 (use NIM input for TTC fibert out)

*** CLK indipendente da usare con un fanout per poi collegarlo al CLK IN di piu schede**

CMS DAQ Zynq Ultrascale+



Slave cmd:
0x1C 0x01 (use external CLK)
0x1A 0x01 (use NIM input for TTC fibert out)

Master cmd:
0x1C 0x00 (use internal CLK)
0x1A 0x00 (use internal TTC for fibert out)

*** CLK indipendente da usare con un fanout per poi collegarlo al CLK IN di piu schede**