CARLOSrx v4 rel 1.2 reference manual

Samuele Antinori, Filippo Costa, Davide Falchieri,
Alessandro Gabrielli, Enzo Gandolfi,
Massimo Masetti, Samuele Zannoli

Department of Physics and INFN Bologna

February 2005

Outline

What's new in CARLOSrx v4 rel 1.2

Hardware upgrades:

- Same board as CARLOSrx rel 1.

Firmware upgrades:

- CARLOSrx v4 rel 1.2 main upgrade with respect to CARLOSrx rel 1 concerns the ways parameters are downloaded on the front-end chips from the PC. Since the DRORC does not support JTAG transmission over the DDL (even if the DAQ TDR says the contrary …), front-end configuration data have to be sent over the DDL using particular commands, stored on a RAM on CARLOSrx, that provides a JTAG protocol towards CARLOS, AMBRA and PASCAL. So far a 256 32-bit words RAM has been implemented on the CARLOSrx FPGA. The format of the file written into this RAM has been carefully studied in order to avoid ambiguities when CARLOSrx has to decode it and send the correct information towards the front-end chips.

Software upgrades:

- Two new C++ tools have been written:
    - *main_menu.c*: it allows to generate the file to be downloaded on the CARLOSrx RAM starting from the value of the parameters in the files:
        - *PASCAL.config*
        - *AMBRA.config*
        - *CARLOS.config*
        - *CARLOSrx.config*
    - *check_jtag.c:* it allows to decode the JTAG answers coming from the front-end chips, for example when reading back register values.

General description

This is the new sequence of actions occurring when performing parameters downloading with CARLOSrx release 1.2:

1. a active low reset is sent towards CARLOSrx;
2. the DDL sends towards CARLOSrx the front end command STBWR (Start of Block Write);
3. the DDL sends towards CARLOSrx the 32-bit words containing the front-end chips parameters with the format shown in Table 1. CARLOSrx stores these words into a 256 32-bit RAM.
4. after the last configuration word, the DDL sends the FESTW (Front End Status Word) meaning that the parameter downloading on CARLOSrx is over.
5. CARLOSrx begins reading out data from the RAM, interpreting them and sending commands and parameter values using the JTAG protocol towards CARLOS through the serial back-link.
6. The JTAG answers coming from the front-end chips are stored into the 80 KByte CARLOSrx FIFO (the same one used for storing data during data acquisition and processing).
7. the DDL sends towards CARLOSrx the front end command STBRD (Start of Block Read);
8. CARLOSrx takes possession of the bidirectional bus *fbd* and begins sending towards the DDL the JTAG words until the FIFO is completely empty.
9. CARLOSrx sends the FESTW.
10. the DDL sends the EOBTR command (End of Block Transfer).
11. CARLOSx sends to CARLOS the command "Enter in RUN mode" via the serial backlink .
12. the busy turns from 1 to 0 and trigger inputs can be accepted.
13. before a new data acquisition run can take place, the RDYRX (Ready to receive) command has to be sent to CARLOSrx through the DDL.

The process of sending the commands STBWR and STBRD can be easily run using a script containing the following instructions:

write_block 0 *configuration_file.txt* %08X
read_block 0 *jtag_answers.txt* %08X

where:

- 0 is the address of the front-end electronics;
- *configuration_file.txt* is the file containing configuration parameters with the format reported in Table 1.
- *jtag_answers.txt* is the name of a new file that will contain the JTAG answers from the front-end chips. This file has to be decoded in order to check that that JTAG programming has been successfully accomplished.

An example script file follows:

```
# Fe2C script file
#

# reset RORC
# reset DIU
# reset SIU

write_block 0 configuration_file.txt %08X
# write_block 0 jtag_8a_c.txt %08X
# write_block 0 jtag_baseline_50.txt %08X
# write_block 0 jtag_ambra_left.txt %08X
# write_block 0 jtag_ambra_right.txt %08X
# write_block 0 jtag_pascal_left.txt %08X
# write_block 0 jtag_pascal_right.txt %08X
# write_block 0 config.dat.carlos %08X

read_block 0 jtag_answers.txt %08X

# read_and_check_block 0 expected.txt %08X

# end Fe2c script file
```

The last instruction, *read_and_check_block*, can be used to compare the JTAG answers coming from the front-end chips with the expected values. It is useful to check if the JTAG answers are repetitive from time to time.

Configuration file format

	number of words to be read from the RAM			
CARLOSrx	*10xxxxxx*	FF	anode length	load trigger
CARLOS	*11xxxxxx*	*address*	*address*	*address*
	TH ch1	TH ch0	TL ch1	TL ch0
	stop if error	enable 2D	anode length ch1	anode length ch0
	FF	FF	FF	read back
AMBRA ch0 x4	*0000xx10*	*address*	*address*	*address*
	tx address	read counter	write counter	SOP delay
	FF	FF	baseline present	read back
	baseline 3	*baseline 2*	*baseline 1*	*baseline 0*
	...	...	...	...
	baseline 63	*baseline 62*	*baseline 61*	*baseline 60*
PASCAL ch0 x4	*0000xx00*	*address*	*address*	*address*
	calibration DAC		VREF control DAC	
	ADC full	AM full	gain control	sel cal channels
	FF	FF	FF	read back
AMBRA ch1 x4	*0010xx10*	*address*	*address*	*address*
	tx address	read counter	write counter	SOP delay
	FF	FF	baseline present	read back
	baseline 3	*baseline 2*	*baseline 1*	*baseline 0*
	...	...	...	...
	baseline 63	*baseline 62*	*baseline 61*	*baseline 60*
PASCAL ch1 x4	*0010xx00*	*address*	*address*	*address*
	calibration DAC		VREF control DAC	
	ADC full	AM full	gain control	sel cal channels
	FF	FF	FF	read back

Table 1: Configuration file format

Notes:

- For stopiferror and enable 2D the following encoding has been used:
  - 0 → 00000000
  - 1 → 01111111
- For ADC full and AM full the following encoding has been chosen:
  - 11111111 → full frequency
  - 00000000 → half frequency
- tx_addr:

  - tx_addr is only written during individual addressing
  - only the 2 LSBs are meaningful for tx_addr.
- The baseline values are 6-bit numbers. Only the 6 LSBs are taken into account.

How to get a *configuration_file.txt*

After manually editing the configuration files:

- PASCAL.config;
- AMBRA.config;
- CARLOS.config;
- CARLOSrx.config

with the desired parameter values, just run *./main_menu.out* and select which front-end chips to configure. The tool generates two output files:

- *jtag_new.txt*: it is the *configuration_file.txt* desired file (just rename it).
- *jtag_list.txt*: it is a list of the JTAG commands that CARLOSrx sends to the front-end chips with the current configuration file. This file is useful for JTAG answers decoding.

A sample *jtag_list.txt* file follows:

```
#                    AMBRA0__LEFT
A:WRITE_SOP_DELAY 16
0
A:WRITE_WCNT_STOP 16
0
A:WRITE_RCNT_STOP 16
0
A:WRITE_TX_ADDRESS 16
0
A:READ_SOP_DELAY 16
9
A:READ_WCNT_sTOP 16
9
A:READ_RCNT_STOP 16
9
A:READ_TX_ADDRESS 16
9
#                    PASCAL0__LEFT
P:LOAD_VREF_CTRL_DAC 16
0
P:LOAD_CALIBR_DAC 16
0
P:LOAD_SEL_CAL_CHANN 16
0
P:LOAD_GAIN_CONTROL 16
0
P:SET_AM_FULL 16
0
P:SET_ADC_FULL 16
0
P:READ_VREF_CTRL_DAC 16
9
P:READ_CALIBR_DAC 16
9
```

P:READ_SEL_CAL_CHANN 16
9
P:READ_GAIN_CONTROL 16
9
P:READ_AM_ADC_FREQ 16
9
CARLOS
C:LOAD_T1L_LEFT 5
9
C:LOAD_T1L_RIGHT 5
9
C:LOAD_T1H_LEFT 5
9
C:LOAD_T1H_RIGHT 5
9
C:LOAD_AL_LEFT 5
9
C:LOAD_AL_RIGHT 5
9
C:LOAD_ENABLE2D 5
9
C:LOAD_STOP_IF_ERROR 5
9
C:READ_T1L_LEFT 5
9
C:READ_T1L_RIGHT 5
9
C:READ_T1H_LEFT 5
9
C:READ_T1H_RIGHT 5
9
C:READ_AL_LEFT 5
9
C:READ_AL_RIGHT 5
9
C:READ_ENABLE_2D 5
9
C:READ_STOPIFERROR 5
9

Front-end chips configuration files

PASCAL.config:

READ_REG	Y
VALUE VREF CONTROL DAC	75
VALUE CALIBR DAC	511
VALUE SELECT CALIBRATION	21
VALUE GAIN CONTROL	3
SET AM FULL	Y
SET AM HALF	N
SET ADC FULL	Y
SET ADC HALF	N
READ AM ADC FREQ	Y

AMBRA.config:

READ_REG	Y
VALUE SOP DELAY	160
VALUE WCNT STOP	255
VALUE RCNT STOP	255
WRITE BASELINE	N

CARLOS.config:

AL RIGHT	255
AL LEFT	255
TL RIGHT	0
TL LEFT	0
TH RIGHT	0
TH LEFT	0
ENABLE 2D	Y
READ REG	Y
STOP IF ERROR	0

CARLOSRX.config:

VALUE AL CARLOSRX	255
VALUE TRIGGER	209

How to decode a *jtag_answers.txt*

Just run:
./check_jtag.out jtag_answers.txt jtag_list.txt

The tool produces a *check_jtag.txt* file with all the information on the commands sent via JTAG and on the value of the registers being read.

A sample *check_jtag.txt* file follows:

```
#                    AMBRA0__LEFT
A:WRITE_SOP_DELAY
A:WRITE_WCNT_STOP
A:WRITE_RCNT_STOP
A:WRITE_TX_ADDRESS
A:READ_SOP_DELAY
010100000 h00A0
A:READ_WCNT_sTOP
011111111 h00FF
A:READ_RCNT_STOP
011111111 h00FF
A:READ_TX_ADDRESS
000000000 h0000
#                    PASCAL0__LEFT
P:LOAD_VREF_CTRL_DAC
P:LOAD_CALIBR_DAC
P:LOAD_SEL_CAL_CHANN
P:LOAD_GAIN_CONTROL
P:SET_AM_FULL
P:SET_ADC_FULL
P:READ_VREF_CTRL_DAC
001001011 h004B
P:READ_CALIBR_DAC
000000000 h0000
P:READ_SEL_CAL_CHANN
000010101 h0015
P:READ_GAIN_CONTROL
000000011 h0003
P:READ_AM_ADC_FREQ
000000000 h0000
#                    CARLOS
C:LOAD_T1L_LEFT
000000000 h0000
C:LOAD_T1L_RIGHT
000000000 h0000
C:LOAD_T1H_LEFT
000000000 h0000
C:LOAD_T1H_RIGHT
000000000 h0000
C:LOAD_AL_LEFT
011111111 h00FF
C:LOAD_AL_RIGHT
```

011111111 h00FF
C:LOAD_ENABLE2D
011111111 h00FF
C:LOAD_STOP_IF_ERROR
000000001 h0001
C:READ_T1L_LEFT
000000000 h0000
C:READ_T1L_RIGHT
000000000 h0000
C:READ_T1H_LEFT
000000000 h0000
C:READ_T1H_RIGHT
000000000 h0000
C:READ_AL_LEFT
011111111 h00FF
C:READ_AL_RIGHT
011111111 h00FF
C:READ_ENABLE_2D
111111111 h01FF
C:READ_STOPIFERROR
000000000 h0000

How to install the software

1. gunzip carlosrx_rel1.2.tar.gz
2. tar –xvf carlosrx_rel1.2.tar
3. make gen_file
4. ./file_generator.out
5. make
6. gcc –o check_jtag.out check_jtag.c

Updated versions of the software are available at the site:
http://www.bo.infn.it/~falchier/alice.html

Using CARLOSrx rel1.2: step by step

- **STEP1**: send an active low reset to CARLOSrx;
- **STEP2**: run the command:

  /date/pRorc/Linux/FeC2_1 -F /home/datesdd/fe2c/Fe2C.scr –v
- **STEP3**: decode JTAG data (if you want to). Run the command:

  ./check_jtag.out jtag_answers.txt jtag_list.txt
- **STEP4**: begin a new run with DATE (Start processes → Start)
- **STEP5**: begin sending trigger signals and acquiring data.

CARLOSrx rel1.2 pinout

```
NET "ck_in" TNM_NET = "ck_in";
TIMESPEC "TS_ck_in" = PERIOD "ck_in" 90 MHz HIGH 50 %;
#PACE: Start of Constraints generated by PACE
#PACE: Start of PACE I/O Pin Assignments
NET "busy"  LOC = "j18" | SLEW = FAST ;
NET "carlos4_output<0>"  LOC = "h4"  ;
NET "carlos4_output<10>"  LOC = "k2"  ;
NET "carlos4_output<11>"  LOC = "k1"  ;
NET "carlos4_output<12>"  LOC = "k4"  ;
NET "carlos4_output<13>"  LOC = "k3"  ;
NET "carlos4_output<14>"  LOC = "h2"  ;
NET "carlos4_output<15>"  LOC = "h1"  ;
NET "carlos4_output<1>"  LOC = "h3"  ;
NET "carlos4_output<2>"  LOC = "h5"  ;
NET "carlos4_output<3>"  LOC = "j6"  ;
NET "carlos4_output<4>"  LOC = "g2"  ;
NET "carlos4_output<5>"  LOC = "g1"  ;
NET "carlos4_output<6>"  LOC = "e2"  ;
NET "carlos4_output<7>"  LOC = "e1"  ;
NET "carlos4_output<8>"  LOC = "l5"  ;
NET "carlos4_output<9>"  LOC = "l4"  ;
NET "cav"  LOC = "m6"  ;
NET "ck_in"  LOC = "e12"  ;
NET "ck_strip"  LOC = "aa15" | SLEW = FAST ;
NET "fbctrl_n"  LOC = "d13" | SLEW = FAST ;
NET "fbd<0>"  LOC = "l17" | SLEW = FAST ;
NET "fbd<10>"  LOC = "m17" | SLEW = FAST ;
NET "fbd<11>"  LOC = "n17" | SLEW = FAST ;
NET "fbd<12>"  LOC = "n22" | SLEW = FAST ;
NET "fbd<13>"  LOC = "n21" | SLEW = FAST ;
NET "fbd<14>"  LOC = "r22" | SLEW = FAST ;
NET "fbd<15>"  LOC = "r21" | SLEW = FAST ;
NET "fbd<16>"  LOC = "r20" | SLEW = FAST ;
NET "fbd<17>"  LOC = "r19" | SLEW = FAST ;
NET "fbd<18>"  LOC = "r18" | SLEW = FAST ;
NET "fbd<19>"  LOC = "p17" | SLEW = FAST ;
NET "fbd<1>"  LOC = "l18" | SLEW = FAST ;
NET "fbd<20>"  LOC = "t22" | SLEW = FAST ;
NET "fbd<21>"  LOC = "t21" | SLEW = FAST ;
NET "fbd<22>"  LOC = "v22" | SLEW = FAST ;
NET "fbd<23>"  LOC = "v21" | SLEW = FAST ;
NET "fbd<24>"  LOC = "v20" | SLEW = FAST ;
NET "fbd<25>"  LOC = "v19" | SLEW = FAST ;
NET "fbd<26>"  LOC = "w22" | SLEW = FAST ;
NET "fbd<27>"  LOC = "w21" | SLEW = FAST ;
NET "fbd<28>"  LOC = "y22" | SLEW = FAST ;
NET "fbd<29>"  LOC = "y21" | SLEW = FAST ;
NET "fbd<2>"  LOC = "l19" | SLEW = FAST ;
NET "fbd<30>"  LOC = "w20" | SLEW = FAST ;
NET "fbd<31>"  LOC = "aa20" | SLEW = FAST ;
NET "fbd<3>"  LOC = "l20" | SLEW = FAST ;
NET "fbd<4>"  LOC = "l21" | SLEW = FAST ;
NET "fbd<5>"  LOC = "l22" | SLEW = FAST ;
NET "fbd<6>"  LOC = "m21" | SLEW = FAST ;
```

```
NET "fbd<7>"  LOC = "m20" | SLEW = FAST ;
NET "fbd<8>"  LOC = "m19" | SLEW = FAST ;
NET "fbd<9>"  LOC = "m18" | SLEW = FAST ;
NET "fbd_strip<0>"  LOC = "aa14" | SLEW = FAST ;
NET "fbd_strip<1>"  LOC = "ab15" | SLEW = FAST ;
NET "fbten_n"  LOC = "c13" | SLEW = FAST ;
NET "fiben_n"  LOC = "e11"  ;
NET "fidir"  LOC = "d12"  ;
NET "filf_n"  LOC = "c12"  ;
NET "fobsy_n"  LOC = "b13" | SLEW = FAST ;
NET "foclk"  LOC = "a13" | SLEW = FAST ;
NET "foclk_strip"  LOC = "ab14" | SLEW = FAST ;
NET "gol_ready"  LOC = "e6" | SLEW = FAST ;
NET "reset_carlos"  LOC = "c1" | SLEW = FAST ;
NET "reset_n"  LOC = "e22"  ;
NET "serial_backlink"  LOC = "c2" | SLEW = FAST ;
NET "tdc"  LOC = "k22" | SLEW = FAST ;
NET "testpulse"  LOC = "e20"  ;
NET "trigger"  LOC = "g22"  ;
NET "tx_en"  LOC = "m3"  ;
```

References

- ALICE Technical Design Report: CERN-LHCC-2003-062, ALICE TDR 010, 7 January 2004
- FEE control and configuration via the DDL, Ervin Denes, ALICE DAQ meeting, Mach 10, 2003 (available at www.cern.ch/ddl)
- Hardware Guide for the front-end Designers Vers 2.1, Csaba Soos (available at www.cern.ch/ddl)
- CARLOS and CARLOSrx datasheets
  (available at www.bo.infn.it/~falchier/alice.html)