Mini-Crate Software Reference Manual

CCB Commands Description

L. Castellani

I.N.F.N. sez. PADOVA

5 May 2011

Document Version 6.9

Firmware Version 1.34

Introduction

The commands that the CCB accept through the serial port are transmitted with the format

[0x55][n+2][data 0][data 1]…..[data n-1][CRC h][CRC l]

in which [data 0] represents the code of the command, while the rest of the data represents the arguments of the command. The CRC polynomial X16+X12+X5+1 is calculated for all bytes of the frame starting from the most significant bit until the less significant bit.
The format of the arguments is BIG_ENDIAN, the type float is in DSP format [16bit mantissa][16bit exponent] value= $m * 2^e$ (1>m>=0 no hidden bit), normalizes the mantissa between 0 and 1.
[2^0][2^-1] [2^-2] [2^-3] [2^-4] [2^-5] [2^-6] [2^-7] [2^-8] [2^-9] [2^-10] [2^-11] [2^-12] [2^-13] [2^-14] [2^-15].
The sign is encoded into the mantissa by taking the 2.s complement for negative numbers and adding a 1 bit in the front. For positive numbers a 0 bit is added at the front.
Negative exponents is in 2.s complement (see appendix A for conversion from/to IEEE32), the type char is 8bits, int is 16bits, short is 16bits, long is 32bits.
The argument name followed by **:** and a number, indicate a bit field (see bit field in C/C++ language) example: char AnPwr:1; .
The bit field order is LESS SIGNIFICANT BIT FIRST, MOST SIGNIFICANT BYTE FIRST.

To communicate by primary port (Optical RS232) the serial port must be configured with 8bits data , 1 stop bit and no parity.
To communicate by secondary port (half duplex RS485) or by debug port (RS232), the serial port must be configured with 8bits data , 1 stop bit, space parity for data transmission and mark parity for address transmission. The secondary port and debug port use the same UART channel but with different electrical standard. Before to communicate with CCB by this ports, the CCB must be activated by an address frame. The address frame format [0x55][ADDR h][ADDR l][CRC h][CRC l]. On debug port can be used broadcast address (0xFFFF). If broadcast address is used on secondary port the commands are executed on all CCB connected on the bus but any return data will not be transmitted on the bus.
The baud rate for all ports is 38400.

The CCB at power on, or after a reset, executes the BOOT program. In this phase it transmits on the primary serial port some PING data (code 0xE8) that must be retransmitted to the CCB: these data are used for the threshold setup of the optical link discriminator. At the end of this phase the CCB transmits on primary and debug port some results of the tests with the code 0xFA to indicate that the CCB is powered on or a reset. The structure of the data is identical to the one returned from the command **Status** (code 0xEA).
At this point, after 30 second if the CCB doesn't receive any BOOT command except Status command, it runs MINI-CRATE program stored in FLASH memory if it exist. With the command **Script** (code 0xEE and argument 0x08 0x02 0x00) is possible to run the MINI-CRATE program without waiting for the 30 seconds timeout.
The BOOT commands, detailed below, are reserved for firmware update.
When mini-crate executes BOOT program the access by primary and secondary port can't be simultaneous, If simultaneous command arrive on both port the data on one port will be lose.
At the beginning the MINI-CRATE program perform a self test of the duration of about 2 minute if it don't find saved configuration of the mini crate in the flash memory, otherwise it restore the saved configuration, but from firmware version 1.30 not power TRBs and ROBs, in this phase the CCB can receive commands but it will respond with a **BUSY code 0x3F** and the command will not be executed.
If on the mini-crate will be replace one board the stored configuration can't be loaded and at power on will be execute a mini-crate self-test.
If CCB receive **unknown command** the return data is **codes 0xFC 0x00**.
The MINI-CRATE software accept on Script command this arguments:
- "mctest" Force mini-crate test and ignore configuration stored on flash;
- "sb" skip all startup sequence and power on only SB/CCB.
- "null" skip all startup sequence.

The argument is a C-style null-terminated string.
Invalid argument are signaled by InvalidArg flag in the mini-crate status data structure.

Automatic Optical Link Discriminator Threshold setup procedure:

From firmware version 1.19 is implemented an automatic procedure to set up optical link discriminator threshold :

- after 10 minutes, or if the program is in idle state after 30 seconds, the program send on primary port a request with code 0xAB
- if after the request the program receive the same code it execute the procedure with ping code to setup threshold and send the code 0xAC to signal the end.
- if after 1second timeout from the request the program not receive the code 0xAB it repeat the request for maximum 3 times, and at the last time it execute the procedure with ping code to setup threshold and send the code 0xAC to signal the end.

The Mini-Crate-Server program must implement the following actions:

- when receive the code 0xAB the server can complete already transmitted command and after answer with 0xAB code. The server must send only one answer also if it receive more than one request, for example because server can't answer before the 1second timeout request.
- retransmit the ping command received from Mini-Crate.
- when receive the code 0xAC the server can return in normal functionality.
- if after 5 seconds it not receive 0xAC command the server can return in normal functionality.

TTC Commands :

The Mini-Crate accept from TTCrx the following commands:
System Broadcast BRCST[5..2] = 0000(binary) ignored;
System Broadcast BRCST[5..2] = 0001(binary) generate the Mini-Crate RESET;
System Broadcast BRCST[5..2] = 0010(binary) generate Test Pulse Sequence Reset;
System Broadcast BRCST[5..2] =from 0011 to 1111(binary) ignored;
User Broadcast BRCST[7..6] = 00(binary) ignored;
User Broadcast BRCST[7..6] = 01(binary) generate Test Pulse Sequence Advance;
User Broadcast BRCST[7..6] = 10(binary) generate the start of the Test Pulse.
User Broadcast BRCST[7..6] = 11(binary) ignored;

Mini-Crate power on sequence :

Power on supply for 3.3V and after power on supply for 5V, to power off mini crate , power off supply for 5V and after power off supply for 3.3V.

Summary of firmware changes

Firmware Version 1.6

- ➔ New Command code 0x8E, TTC fine delay.
- ➔ New Command code 0x8F, Read Mini crate sensors ID.
- ➔ Added new argument on mini crate structure: EnTrgPhi, EnTrgThe, EnTrgH, DisTrbCk, DisSbCk, DisOsc, L1A_Delay, EnAutoTrg, SelL1AVeto, ForceTp, CCBReady, unused.

Firmware Version 1.7

- ➔ New Command code 0x91, CCBRdy Pulse.
- ➔ Replaced on mini crate status structure the 32bit counter "LoseLockCount" with three 8bit counter and 8bit unused. The counter are LoseLockCountTTC, LoseLockCountQPLL1, LoseLockCountQPLL2 .
- ➔ Added new argument on mini crate status structure: RunInProgress.
- ➔ Resolved a error on Test Front-End command.
- ➔ Resolved a error on TRB power-on command.

Firmware Version 1.8

- ➔ Changed max temperature threshold at 50.0°C.
- ➔ Added new command code 0x92, PIRW.
- ➔ Added check of the parallel interface RW and /PROG net on self test, the result is written on bit 11 and 12 of the TrbPiTest result.
- ➔ New command code 0x93: Disable TRB CK.
- ➔ New command code 0x94: Disable SB CK.
- ➔ New command code 0x95: Disable Oscillator CK.
- ➔ New command code 0x96: Restore CFG.
- ➔ New command code 0x97: Write custom data.
- ➔ New command code 0x98: Read custom data.

Firmware Version 1.9

- ➔ Changed Front-Ends reset function for dummy chamber.
- ➔ Corrected an error on command code 0x78 : Relative Test-Pulse settings.
- ➔ Added arguments on command code 0x8B: Set Test-Pulse offset.
- ➔ Added arguments on command code 0x8C: Read Test-Pulse offset.
- ➔ Resolved an error on temperature monitor function.

Firmware version 1.10

- ➔ Corrected on this document an error on description of the return arguments order from command Status on boot : Tp1H, Tp1L,Tp2H, Tp2L.
- ➔ New command code 0x9A: Front-ends super layer enable.
- ➔ New command code 0x9B: Front-ends super layer enable Max Temperature.
- ➔ Corrected description return type argument on code 0x10, TrbTestJtag[8] is int type.
- ➔ Modified return data from command code 0x33 for correct improper read ROB error signal.

Firmware version 1.11

- ➔ Optimized command code 0x52.
- ➔ Added new command 0x9C: Compare BTI register with memory.
- ➔ Added new script argument “null”.
- ➔ Added new argument on mini crate status structure: min an max value read for Vccin, Vddin, Sb_Vcc, and Sb_Vdd.
- ➔ Added new command 0x9E: Clear min max voltage monitor.
- ➔ Added new argument on self test result structure: Abort_Vcc and Abort_Vdd.

Firmware version 1.12

- ➔ Corrected an error on command code 0x8F, Read Mini crate sensors ID.

Firmware version 1.13

- ➔ Added new command code 0x9F: Select L1A / Veto Mode.

Firmware version 1.14

- ➔ Added new command code 0xA0: TRB PI Test.
- ➔ Added new command code 0xA1: Clear TRB Seu counters.
- ➔ Added the TRB Seu monitor process.
- ➔ Added new argument on command code 0x5F: Test PI.
- ➔ Added new command code 0xA1: TestIR.
- ➔ Resolved an error on write TRACO configuration after board power on.
- ➔ Resolved an error on write TRACO LUT after mini-crate power on.

Firmware version 1.15

- ➔ Updated this document with all BOOT commands.
- ➔ Changed max temperature limit to 45°C.
- ➔ Modified ROBs power on/off function and RobOnTime unit on test result data structure.
- ➔ Corrected an error on ROB fault power monitor.
- ➔ Corrected an error on JTAG test commands for the ROBs.

Firmware version 1.16

- ➔ Added new flag on mini-crate status data structure (CfgNotChanged).
- ➔ Added new command code 0xA2, GetConfigCRC.

Firmware version 1.17

- ➔ Modified Save and Load configuration functions and start up procedure.
- ➔ Added new flags on mini-crate status data structure (InvalidArg and CfgLoaded).
- ➔ Modified return data from char to int on command Load CFG code 0x61.
- ➔ Modified return data from char to int on command Save CFG code 0x40.
- ➔ Removed Restore CFG command code 0x96 (included in Load CFG code 0x61).
- ➔ Corrected an error on Front-ends super layer enable command, code 0x9A.
- ➔ Added new command Find Sensor, code 0xA6.
- ➔ Added new arguments on self-test return data structure.

Firmware version 1.18

- ➔ Removed command Write TRACO LUTs, code 0x3E.
- ➔ Added new command Write Mini-Crate LUTs, code A8.
- ➔ Removed Read TRACO LUT parameters, code 0x88.
- ➔ Added new command Read Mini-Crate LUT parameter, code 0xA9.

Firmware version 1.19

- ➔ Added automatic optical link discriminator threshold setup procedure (default is disabled, because MC-Server not yet support it, use command 0x77 to enable it).
- ➔ Resolved an error on Relative Test Pulse settings, code 0x78.
- ➔ Modified BTI Emulation function to reduce delay from CCB_Ready pulse to TRGOUT signal.
- ➔ Resolved an error on ROBs configuration during power up.

Firmware version 1.20

- ➔ This firmware have a bug on front end temperature read out.

Firmware version 1.21

- ➔ Added trigger histogram , see command Clear Histogram, code 0xAD and Read Histogram, code 0xB0.
- ➔ Added new command Trigger Frequency, code 0xB2.
- ➔ Added new command code 0xAF for parallel interface tests.
- ➔ Added new command Chamber map code 0xB4
- ➔ Added new argument on status data structure.
- ➔ Modified command Front-end Test code 0x4B.
- ➔ Added new command Retransmit code 0xB5.
- ➔ Added new special code 0xB6 ad 0xB7, to split return data.

Firmware version 1.22

- ➔ Added new command 0xB8 for Sector-Collector synchronization.
- ➔ Reduced maximum temperature threshold to 40°C.
- ➔ Added new argument TempTestDisabled on status data structure.

Firmware version 1.23

- ➔ Corrected an error on LUT computation.

Firmware version 1.24

- ➔ Modified Front-Ends read temperature function.
- ➔ The disable flag for Link Monitor is saved on flash.
- ➔ Added new command Set Timeouts Link Monitor, code 0x96. The values are saved on flash.
- ➔ Added new command Write TDC setup code 0x21

Firmware version 1.25

- ➔ Corrected an error on TRB's SEU monitor when active RunInProgress.
- ➔ Added new command SEU Mask code 0x2E.
- ➔ Added new command Read SEU Mask, code 0xB9.
- ➔ Added new command Check SEU LUT , code 0x3E.
- ➔ Modified the load configuration error code result.
- ➔ Modified automatic load configuration procedure.
- ➔ Added CfgLoadResult on status data structure.
- ➔ Generate ROB-RESET after load configuration
- ➔ Added write flash and protect flash command with the same code as BOOT program to facilitate firmware update.
- ➔ Modified command Trigger Frequency, code 0xB2.

Firmware version 1.26

- ➔ Modified command Mini-Crate Temperature code 0x3C.
- ➔ Corrected an error when use the Bridge special code.
- ➔ Added new command Mask Temperature code 0xBB.
- ➔ Added new command Check TDCs Status, code 0xBC.
- ➔ Added new argument TDCsStatusFlags on status data structure.
- ➔ Modified return data of the command Get Configuration CRC, code 0xA3.
- ➔ Corrected an error on command Read TRACO LUT, code 0x86.

Firmware version 1.27

- ➔ Added a patch to prevent the fail result of the UARTs self-test executed during program start-up. The self-test fail when there is a signal in the UART input. The path repeat UARTs self-test four time before check one not working. Added not documented commands (used for debugging) to force UART self-test and check result.
- ➔ Corrected an improper notification error of the load configuration, when using Script command with no argument.
- ➔ Added not documented command for capturing power-on failure, this command are used only for debugging.
- ➔ Modified the function that monitor the FAULT line of the TRB board. if the FAULT line is active the line is read three time before signaling a fault.
- ➔ Modified send frame address function on secondary port, used when use bridge special code.

Firmware version 1.28

- ➔ Changed the formula to calculate the index for angle LUT, see Appendix D.
- ➔ Eliminate the CfgLoaded flag clear when power on/off LED alignment and RPC

Firmware version 1.29

- ➔ Added new command Power Mask, code 0x30.
- ➔ Added new argument PowerMask on status data structure.
- ➔ From this firmware load configuration command can read previous flash configuration format version if it is changed and then with save configuration is stored new version format.

Firmware version 1.30

- ➔ Modified Power on/off command 0x2F.
- ➔ Modified startup sequence, the TRBs and ROBs are not automatically powered.

Firmware version 1.31

- ➔ Automatic restore TTCrx configuration after a TTCrx watchdog auto reset.
- ➔ Corrected TDC check status on inexistent TDC on mini-crate MB1.
- ➔ Corrected a possible mismatch from TRBs power indicator and real state of TRBs.
- ➔ Modified data returned by command Read SEU Mask, return also witch chip have generate the SEU .
- ➔ Corrected CfgNotChanged indicator flags.
- ➔ Changed name and modified command Compare BTI Register to Compare Chip Register.
- ➔ Modified JTAG TDI read input to prevent false read generated by crosstalk with Sequence Advance signal.
- ➔

Firmware version 1.32

- ➔ Corrected an error on command Get Configuration CRC, code 0xA3.

Firmware version 1.33

- ➔ Moved the TTCrx configuration on trigger CRC.

Firmware version 1.34

- ➔ Added the clear SEU rams counters on Clear TRB SEU counters command, code 0xA1.

The BOOT commands:

Script, code 0xEE:
This **boot** command is used to run program.
The 24bit address is where the CCB find necessary information to run program. The script data is always present in the firmware, normally at the address 0x80200 if run firmware stored in FLASH or at the address 0x200 if you load the RAM firmware version directly in RAM. The RAM firmware differ from FLASH version only in the script data.
Arguments:

char page;	8 bits address high.
int addr;	16 bits address low.
char arg[n];	Optional argument, n from 0 to 32. The argument is C-style null-terminated string.

Return data:

char 0xEE;	
char size;	if size is not zero the script is valid and it will be executed.

The data pointed by the command address is a sequence of command necessary to execute before running the program. the sequence must terminate with command END.
The script commands and format are:

---

command WRITEBITS:	Set bits at the address specified.
`int cmd;`	`cmd=1.`
`long addr;`	`address.`
`int type;`	`variable type. 1=char, 2=int, 4=long.`
`long mask;`	`mask of bits to write.`
`long value;`	`value of bits`

---

command WRITE:	Write data at address specified.
`int cmd;`	`cmd=2.`
`long addr;`	`address to write.`
`long size;`	`size of data.`
`char data[size];`	`data.`

---

command COPY:	Copy data from saddr to daddr. This command is used to transfer the executable from FLASH to RAM
`int cmd;`	`cmd=3.`
`long saddr;`	`source address.`
`long daddr;`	`destination address.`
`long size;`	`size of the data to copy.`

---

command RUN:	Execute the function pointed by funcptr, prototype is **void func(void *arg)**. The function is the startup code that setup stack pointer, initialize C++ class etc and then call the main function.
`int cmd;`	`cmd=4.`
`long funcptr;`	`function pointer.`

---

command DELAY:	Wait for the time specified in delay.
`int cmd;`	`cmd=5.`
`long delay;`	`unit is 12.8us but the minimal delay generated is about 3ms necessary to execute the function.`

---

command END:	This command must be place at the end of the script sequence.
`int cmd;`	`cmd=0.`
`int id;`	`id=0xA55A.`

Watchdog Reset, code 0xF8
This **boot** command force CPU to enter in a infinite loop to force a watchdog reset. CPU will be reset after sent return data.
This command is present also in mini-crate program.

Return data:
char 0xFC;
char 0xF8;

HD Watchdog Reset, code 0xF7
This **boot** command force CPU to enter in a infinite loop to force an hardware watchdog reset. CPU will be reset after sent return data.
CCB have two different watchdog , one is the internal microcontroller watchdog (intervention time is about 0.4s), the second one is an external watchdog (intervention time is about 1s) , the command force the intervention of the second one.
This command is present also in mini-crate program.

Return data:
char 0xFC;
char 0xF7;

BOOT Status, code 0xEA:
From this command is possible to return two different data structure, it depend if CCB is running BOOT program or MINI-CRATE program.
Return data:

char 0xE9	Identification code of the data structure.
char rsr;	Type of reset, 0x20 corresponds to internal watch-dog reset, 0x80 corresponds to power on, external watch-dog reset or TTCrx reset command.
unsigned int Ccb_ID;	CCB Identified code.
short HVersion;	High version of the BOOT program, actual is 3.
short LVersion;	Low version of the BOOT program, actual is 9.
char EUartIRQ:1;	Test result of the external UART IRQ line, 1=OK.
char TTCi2c:1;	Test result of the TTCrx I2C bus, 1=OK.
char CpldTtc:1;	Ccb board preset.
char QpllARdy:1;	QPLL_A 1=locked.
char QpllBRdy:1;	QPLL_B 1=locked.
char QpllAChng:1;	If 1 QPLL_A lose lock. This bit is cleared after executed this command.
char QpllBChng:1;	If 1 QPLL_B lose lock. This bit is cleared after executed this command.
char TtcCk:1;	Test result of the TTCrx CK, 1=OK.
char IUart:1;	Test result of the internal UART, OK=1.
char EUartA:1;	Test result of the external UART channel A, OK=1.

char EUartB:1;	Test result of the external UART channel B, OK=1.
char ExtRam:1;	Fast RAM test result, OK=1.
char Flash:1;	Test result of the power on FLASH, OK=1.
char AutoRun:1;	Auto run cancelled if zero.
char TTCrxRdy:1;	TTCrx PLL LOOK.
char LinkAutoset:1;	Auto set link threshold procedure successful.
int PwrRam;	Power on RAM result. pwr_on=1, pwr_off=0, pwr_already_off=2, pwr_already_on=3, pwr_on_fail=-1, pwr_fail=-2, invalid_fault=-3 (problem on line fault).
int ErrRam;	RAM diagnostic, must be zero.
int vErrRam;	RAM diagnostic, must be zero.
int PwrFlash;	Power on FLASH result. pwr_on=1, pwr_off=0, pwr_already_off=2, pwr_already_on=3, pwr_on_fail=-1, pwr_fail=-2, invalid_fault=-3 (problem on line fault).
int FlashID;	FLASH memory Code identifier.
int LinkOfs;	Optical link output offset measured in DAC count(Vref=4.53V, DAC bits=10)
int LinkHyst;	Optical link discriminator hysteresis measured in DAC count (Vref=4.53V, DAC bits=10).
int LinkAmp;	Optical link signal amplitude measured in DAC count (Vref=4.53V, DAC bits=10) valid only if LinkAutoset=1.
int LinkThr;	Optical link discriminator threshold set in DAC (Vref=4.53V, DAC bits=10), optimal value if LinkAutoset=1 otherwise it is a default value.
long SEUCounter;	Counter of the SEU event in RAM, if -1 CCB is clearing power on errors.
long RefreshAddr;	Actual refresh RAM address.
long PwrCrashFlags;	Power on crash diagnostic. Any bit identified a crash during power on of one device. The device list from lsb is: RAM, FLASH, ANALOG, CK, RPC, LED, TSMS, TSMDU, TSMDD, SBTH, SBCK, BUFFTRB, VCCTRB, TRB0 to TRB7, ROB0 to ROB6. (see appendix E). Bit 29 is used for CPUCK delay test, bit 30 is used during TRB self test, bit 31 during ROB self test. Send a Watchdog Reset to clear PwrCrashFlags and CRC.
int PwrCrashCrc;	CRC Power on crash diagnostic. It must be used to validate previous argument. Send a Watchdog Reset to clear PwrCrashFlags and CRC.
int CrashFlags;	Invalid interrupt Crash diagnostics. The bit list from lsb is: `BREAKPOINT, BUSERROR, SWINTERRUPT, ILLEGALINSTRUCTION, DIVIDEBYZERO, UNINITINTERRUPT, SPURIOUSINTERRUPT, NOINTERRUPT.` Send a Watchdog Reset to clear CrashFlags and CRC.

int CrashCrc;	CRC Invalid interrupt Crash diagnostics. It must be use to validate previous argument. At power on without problems it is always invalid. Send a Watchdog Reset to clear CrashFlags and CRC.
int PwrAnalog;	Power on ANALOG result. pwr_on=1, pwr_off=0, pwr_already_off=2, pwr_already_on=3, pwr_on_fail=-1, pwr_fail=-2, invalid_fault=-3 (problem on line fault).
float Vccin;	Vcc BusBar voltage
float Vddin;	Vdd BusBar voltage
float Vcc;	CCB Vcc voltage
float Vdd;	CCB Vdd voltage
float Tp1H;	Test pulse 1 reference H voltage. Difference from TP1H and TP1L must be 1.23V (with about 1% tolerance).
float Tp1L;	Test pulse 1 reference L voltage. Difference from TP2H and TP2L must be 1.23V (with about 1% tolerance).
float Tp2H;	Test pulse 2 reference H voltage. Difference from TP1H and TP1L must be 1.23V (with about 1% tolerance).
float Tp2L;	Test pulse 2 reference L voltage. Difference from TP2H and TP2L must be 1.23V (with about 1% tolerance).

Read Com Error, code 0xF0:
Read and clear the first error on communications ports.
This command is present also in mini-crate programs with some more bits.

Return data:

char 0xF0;	
char port;	primary port=1. secondary port=2
char Parity:1;	
char Framing:1;	
char Break:1;	
char Noise:1;	
char Overrun:1;	
char Sync:1;	
char Crc:1;	
char LoseData:1;	
char Size:1;	
char TimeOut:1;	
char Unexpected:1;	
char unused:4;	

Auto LINK set, code 0xE7:
This **boot** command execute the procedure for setup the threshold of the optical link discriminator.

Arguments:

int levL;	set this value to the LinkOfs value returned by status command.
int levH;	set this value to 1023.

Return data:

char 0xFC;	
char 0xE7;	
char result;	non zero if success.

LINK Test, code 0xE4:
This **boot** command work only on secondary port, it forward the received data on the primary port.

Arguments:
char data[size]; data to be forwarded on primary port, size from 1 to 245.
Return data:
no data returned from this command.

Reset SEU, code 0xE6:
Reset the RAM SEU counter;

Return data:
char 0xFC;
char 0xE6;

Link DAC, code 0xE5:
This **boot** command set the optical discriminator threshold to the specific DAC value.

Arguments:
int dacvalue;
Return data:
char 0xFC;
char 0xE5;

TTC write, code 0xE3:
Write TTCrx register by I2C bus.

Arguments:
char addr; register address.
char data; register value.
Return data:
char 0xFC;
char 0xE3;
char error; return 0 if success, no_ack=1, SDA error=-1, SCL error=-2.

TTC read, code 0xE2:
Read TTCrx register by I2C bus.

Arguments:
char addr; register address.
Return data:
char 0xE1;
char data; register value.
char error; return 0 if success, no_ack=1, SDA error=-1, SCL error=-2.

TTC find, code 0xE0:
Find the I2C address of the TTCrx chip.

Return data:
char 0xFC;
chat 0xE0;
char error; return 0 if success, not found=1, SDA error=-1, SCL error=-2.

The following commands are reserved for firmware update

Read, code 0xF5:
Read from CPU address space. See appendix C for address mapping.
This command is present also in mini-crate program. To read FLASH memory you must before power on it, because normally it is off.
Arguments:

char page;	8 bits address high.
int addr;	16 bits address low.
char sizeh;	8 bits size high, must be always 0
int size;	16 bits size low. From 1 to 245.

Return data:

char 0xF4;	
char page;	8 bits address high.
int addr;	16 bits address low.
char data[size];	requested data.

Write, code 0xFE:
Write to CPU address space. See appendix C for address mapping.
Not use this command to write FLASH memory.
This command is present also in mini-crate program.
Arguments:

char page;	8 bits address high.
int addr;	16 bits address low.
char data[size];	data to be write, size from 1 to 245.

Return data:

char 0xFC;
char 0xFE;

Write FLASH, code 0xF3:
Write using the device AT29C0XX sector program cycle.
The command at the end verify the sector and return the result.
Not use this command out of the FLASH memory address space. See appendix C for FLASH memory address mapping.
WARNING: in the mini-crate is installed an AT29C020 (256K x 8) flash memory chip with 256 bytes sector size, so it is necessary to send two write flash command to write a single FLASH sector, the first one address must be aligned to the fist 128 bytes of the sector and the second one with the last 128 bytes of the sector. The data is wrote on the sector with the second command .
It is ***very important*** that mini-crate between first and second command not receive another command (for example any command on the other communication port) because otherwise the fist 128 bytes data are lost and in the sector will be wrote corrupted data.
The mini-crate firmware file generated by C++ compiler (Motorola S format) is not a linear address sequence, so before you write the firmware in flash you must read the file and generate a memory copy of the contents, therefore you can write the sectors contents. If you must write partial sector you must before read the contents part of the sector that you do not want modify, merge it with new contents and than write full sector. You must use this procedure because in flash is present also the mini-crate configuration data, so you prevent possible data corruption.

Arguments:

char page;	8 bits address high.
int addr;	16 bits address low. The address must be aligned to 128bytes size.
char data[128];	If data size differ from 128 byte the missing bytes assume casual value on the FLASH memory.

Return data:

char 0xF2;	
char error;	return 0 if write success;
char sector;	return 0xFF for 256 byte flash sector(AT29C020), or 0x00 for 128 byte flash sector (AT29C010).

Write BITs, code 0xF1:
Write only some bits to CPU address space. See appendix C for address mapping.
Not use this command to write FLASH memory.
Arguments:

char page;	8 bits address high.
int addr;	16 bits address low.
char type;	variable type, 1 for char, 2 for int, 4 for long.
long mask;	mask of the bits to be write.
long value;	value to be write.

Return data:
char 0xFC;
char 0xF1;

Protect/Unprotect FLASH, code 0xED:
This command active or deactivate the write protection on the FLASH memory.

Arguments:

char protect;	if 1 Enable write protection.

Return data:
char 0xFC;
char 0xED;

Erase FLASH, code 0xEB:
This command erase the contents of the FLASH memory.

Return data:
char 0xFC;
char 0xEB;

The MINI-CRATE commands:

Watchdog Reset, code 0xF8:
This command force CPU to enter in a infinite loop to force a watchdog reset. CPU will be reset after sent return data.
This command work also if the mini crate is in BUSY state.
This command is present also in BOOT program.
Arguments:

Return data:
char 0xFC;
char 0xF8;

HD Watchdog Reset, code 0xF7:
This command force CPU to enter in a infinite loop to force an hardware watchdog reset. CPU will be reset after sent return data.
CCB have two different watchdog , one is the internal microcontroller watchdog (intervention time is about 0.4s), the second one is an external watchdog (intervention time is about 1s) , the command force the intervention of the second one.
This command work also if the mini crate is in BUSY state.
This command is present also in BOOT program.

Return data:
char 0xFC;
char 0xF7;

Read Self-Test Result, code 0x10:
Read result of self-tests after power on.

Return data:

char 0x11;	Identification code of the data structure.
char TTCFpga:1;	Test result of the FPGA registers, OK=1.
char AnPwr:1;	Power on analogical block, OK=1.
char CKPwr:1;	Power on CK distribution block, OK=1.
char LedPwr:1;	Power on LED alignment, OK=1.
char RpcPwr:1;	Power on RPC, OK=1.
char BufTrbPwr:1;	Power on TRB buffers, OK=1.
char VccTrbPwr:1;	Power on TRB Vcc, OK=1.
char B1w:1;	Test result bus one wire, OK=1.
char SOPwr:1;	Power on TSM sorter, OK=1.
char DUPwr:1;	Power on TSM data up, OK=1.
char DDPwr:1;	Power on TSM data down, OK=1.
char SBCKPwr:1;	Power on TSM CK distribution , OK=1.
char THPwr:1;	Power on theta fast OR and link , OK=1.
char CPUdelay:1;	CPU CK Fine delay Test result, OK=1.
char TPFineDelay1:1;	Test Pulse1 Fine delay Test result, OK=1.
char TPFineDelay2:1;	Test Pulse2 Fine delay Test result, OK=1.
int OnTime[11];	Measure of the power on time (ms).
int AdcNoise[32];	Measure of the ADC noise (adc count). Element list is: [0]Front-Ends MAD temperature connector 1, [1] Front-ends average temperature connector 1, [2]Front-Ends Vcc connector 1, [3]Front-ends Vdd connector 1, [4]Front-Ends MAD temperature connector 2, [5]Front-ends average temperature connector 2, [6]Front-Ends Vcc connector 2, [7]Front-ends Vdd connector 2, [8]Front-Ends MAD temperature connector 3, [9]Front-ends average temperature connector 3, [10]Front-Ends Vcc connector 3, [11]Front-ends Vdd connector 3, [12]Bias 1, [13]Threshold 1, [14]Bias 2, [15]Threshold 2, [16]Bias 3, [17]Threshold 3, [18]Width, [19]TRBs Fault, [20]Vdd in, [21]Vcc in, [22]Vdd, [23]Vcc, [24]Vdd splitter board, [25]Vcc splitter board, [26]TP1L, [27]TP2L, [28]TP1H, [29]TP1, [30]TP2H, [31]TP2.
int Dac;	Test of the front-end threshold DAC, bit0-1 fe1, bit2-3 fe2, bit4-5 fe3,bit6 width, bit8-9 Test pulse. OK=1.
int SbTestJtag;	Test result of the SB jtag chain, Bit15 indicate that the chain is interrupted, bit(0..7) indicate the number of chips in the chain, bit(8..14) indicate the size of the instruction register.
int SbTestPi;	Test result of the SB parallel interface, OK=0. Any bit to 1of the bit(7..0) indicate a problems on PI bus bit, bit8 indicate problems

	between microcontroller and TSMSorter, bit9 indicate problems between microcontroller and TSMDataDown, bit10 indicate problems between microcontroller and TSMDataUp.
char TrbPwr;	Power on TRBs, bit0-5 for TRB_phi, bit6-7 for TRB_theta, OK=1
char TrbBadJtagAddr;	Invalid Jtag address assigment, bit0-5 for TRB_phi, bit6-7 for TRB_theta, Bad=1.
int TrbPresMsk[8];	Result of the jtag scan address (0-15) with one TRB on, for individuate problem on power on/off line, the bit at one indicate the address where CCB found the board. The element order refer to the hardware TRB map.
int TrbTestJtag[8];	Test result of the TRBs jtag chain. Bit15 indicate that the chain is interrupted, bit(0..7) indicate the number of chips in the chain, bit(8..14) indicate the size of the instruction register. The element order refer to the hardware TRB map.
char TrbFindSensor[8];	Result of the research of the temperature sensors, OK=1, FAIL=0, greater than 1 ambiguity between more sensor. The element order refer to the hardware TRB map.
Int TrbOnTime[8];	Measure of the power on time (ms) of the TRBs. The element order refer to the hardware TRB map.
int TrbPiTest[6];	Test result of the TRBs phi parallel interface, OK=0. Any bit to 1of the bit(7..0) indicate a problems on PI bus, bit8 indicate problems between TSMSorter and TSS, bit9 indicate problems between TSS and TRACO, bit10 indicate problems between TRACO and BTI, bit 11 indicate that PI RW net is not connected, bit 12 indicate that PI /PROG net is not connected. The element order refer to the hardware TRB map.
char RobPwr;	Power on ROBs, one bit for board, OK=1.
char RobBadJtagAddr;	Bad jtag address assignment, one bit for board, BAD=1;
char RobOverlapAddr;	Overlap address assignment.
int RobPresMsk[7];	Result of the jtag scan address (0-15) with one ROB on, to individuate problem on power on/off line, the bit at one indicate the address where CCB found board.
int RobTestJtag[7];	Test result of the ROBs jtag chain. Bit15 indicate that the chain is interrupted, bit(0..7) indicate the number of chips in the chain, bit(8..14) indicate the size of the instruction register.
char RobFindSensor[7];	Result of the research of the temperature sensors, OK=1, FAIL=0, greater than 1 ambiguity between more sensor.
int RobOnTime[7];	Measure of the power on time (0.1ms) of the ROBs.
int McType	MINI CRATE type, MB1=1, MB2=2, MB3=3, MB4=4, MB4_9=5, MB4_4=6, MB4_10=7, invalid=0.
char Abort_Vccin;	The self test was aborted because Vccin is too low.
char Abort_Vddin;	The self test was aborted because Vddin is too low.

char Vofftest:1;	This bit indicate correct voltage value read on Vfault input line. Vfault line is used to measure TRBs voltage by open collector power fault signal.
char unused:5;	
char nTrbBrd;	Number of TRBs connected. This value is calculate by Vfault input line. Voltage on Vfault change by 0.037V for every board connected

float Vfoff;	Vfault voltage with all TRBs off. correct value must be:
	`Vfoff = VP*KP-vdd*0.1-(8-nboard)*VNC*0.1,`
	`where VP= 4.53*10/22, KP=1+10/(100/9), vdd=3.3,`
	`VNC= VP*22/122, nboard= number of TRB connected (included TRB theta).`

float Vtrbmin;	The min value read from all TRB power fault line. Correct value is about 3.0V. This value is measured by Vfault line. The Vfault change by 0.3V for every TRB on
float Vtrbmax;	The max value read from all TRB power fault line. Correct value is about 3.0V. This value is measured by Vfault line. The Vfault change by 0.3V for every TRB on

Run Self-Test, code 0x12:
Execute the self tests of the mini crate.
The self test duration is about 2 minute.

Return data:
Same data as Read Self-Test Result command code 0x10.

Status, code 0xEA:
From this command is possible to return two different data structure, it depend if CCB is running BOOT program or MINI CRATE program.

Return data from **MINI CRATE program**:

char 0x13	Identification code of the data structure.
int Ccb_ID;	CCB Identified code.
int HVersion;	High version of the MINI CRATE program.
int LVersion;	Low version of the MINI CRATE program.
int McType;	MINI CRATE type, MB1=1, MB2=2, MB3=3, MB4=4, MB4_9=5, MB4_4=6, MB4_10=7, invalid=0.
char PwrAn:1;	Power state analogical block.
char PwrCK:1;	Power state CK distribution block.
char PwrLed:1;	Power state LED alignment.
char PwrRpc:1;	Power state RPC.
char PwrTrbBuf:1;	Power state TRB buffers.
char PwrTrbVcc:1;	Power state TRB Vcc.
char PwrSO:1;	Power state TSM sorter.

char PwrDU:1;	Power state TSM data up.
char PwrDD:1;	Power state TSM data down.
char PwrSBCK:1;	Power state TSM CK distribution .
char PwrTH:1;	Power state theta fast OR and link .
char TTCrdy:1;	TTCrx PLL lock state.
char PwrFlash:1;	Power state Flash (normal is off)
char EnTtcCkMux:1;	State of the TTC CK MUX, if zero force the CPU to run with internal CK.
char QpllARdy:1;	QPLL_A 1=locked.
char QpllBRdy:1;	QPLL_B 1=locked.
char PwrTrb;	Power state TRBs, bit0-5 for TRB_phi, bit6-7 for TRB_theta, OK=1
char PwrRob;	Power state ROBs, one bit for board, OK=1.
char AlrmPwrAn:1;	The CCB have detected power fault on analogue block. Clear alarm with power off command.
char AlrmPwrCK:1;	The CCB have detected power fault on CK block. Clear alarm with power off command.
char AlrmPwrLed:1;	The CCB have detected power fault on led alignment block. Clear alarm with power off command.
char AlrmPwrRpc:1;	The CCB have detected power fault on RPC block. Clear alarm with power off command.
char AlrmPwrTrbBuf:1;	The CCB have detected power fault on TRBBUF block. Clear alarm with power off command.
char AlrmPwrTrbVcc:1;	The CCB have detected power fault on TRBVCC block. Clear alarm with power off command.
char AlrmPwrSO:1;	The CCB have detected power fault on SORTER block. Clear alarm with power off command.
char AlrmPwrDU:1;	The CCB have detected power fault on DATAUP block. Clear alarm with power off command.
char AlrmPwrDD:1;	The CCB have detected power fault on DATADOWN block. Clear alarm with power off command.
char AlrmPwrSBCK:1;	The CCB have detected power fault on SBCK block. Clear alarm with power off command.
char AlrmPwrTH:1;	The CCB have detected power fault on THETA FAST OR block. Clear alarm with power off command.
char AlrmTTC:1;	The CCB have detected TTCrx PLL lose lock.
char AlrmPwrFlash:1;	The CCB have detected power fault on FLASH block. Clear alarm with power off command.

char AlrmB1w:1;	
char AlrmQpllAChng:1;	If 1 QPLL_A lose lock. This bit is cleared after executed this command.
char AlrmQpllBChng:1;	If 1 QPLL_B lose lock. This bit is cleared after executed this command.
char AlrmPwrTrb;	The CCB have detected power fault TRB (one bit for board). Clear alarm with power off command.
char AlrmPwrRob;	The CCB have detected power fault ROB (one bit for board). Clear alarm with power off command.
char AlrmTempTrb;	The CCB have detected over temperature on TRB (one bit for board).
char AlrmTempRob;	The CCB have detected over temperature on ROB (one bit for board).
char LoseLockCountTTC;	TTCrx PLL Lose lock counter.
char LoseLockCountQPLL1;	QPLL1 Lose lock counter.
char LoseLockCountQPLL2;	QPLL2 Lose lock counter.
char Unused;	
long RamAddr;	Actual RAM address refresh.
long SeuRam;	Counter of the SEU in the RAM, if -1 CCB clearing power on errors.
int IntRamAddr;	Actual Internal RAM address refresh.
long SeuIntRam;	Counter of the SEU in the internal RAM (1k byte).
long SeuBTI;	Counter of the SEU in the BTI
long SeuTRACO;	Counter of the SEU in the TRACO configuration
long SeuLUT;	Counter of the SEU in the TRACO LUT
long SeuTSS;	Counter of the SEU in the TSS
float Vccin;	Vcc bus bar voltage.
float Vddin;	Vdd bus bar voltage
float Vcc;	CCB Vcc.
float Vdd;	CCB Vdd.
float Tp1L;	Test pulse 1 reference L voltage.
float Tp1H;	Test pulse 1 reference H voltage
float Tp2L;	Test pulse 2 reference L voltage
float Tp2H;	Test pulse 2 reference H voltage
float Fe_Vcc[3];	Front-end VCC, element 0=phi_inner, 1=theta, 2=phi_outer.

float Fe_Vdd[3];	Front-end VDD, element 0=phi_inner, 1=theta, 2=phi_outer.
float Sp_Vcc;	Splitter board VCC.
float Sp_Vdd;	Splitter board VDD.
float Fe_Bias[3];	Front-end Bias, element 0=phi_inner, 1=theta, 2=phi_outer.
float Fe_Thr[3];	Front-end Thresholds, element 0=phi_inner, 1=theta, 2=phi_outer.
float Fe_Width;	Front-end Width
int in_Tmax;	Max Temperature inner super layer (unit 0.1°C).
int in_Tmed;	Average Temperature inner super layer (unit 0.1°C).
int th_Tmax;	Max Temperature theta super layer (unit 0.1°C).
int th_Tmed;	Average Temperature theta super layer (unit 0.1°C).
int out_Tmax;	Max Temperature outer super layer (unit 0.1°C).
int out_Tmed;	Average Temperature outer super layer (unit 0.1°C).
float BrdMaxTemp;	Actual more high board temperature .
char CpuCkDelay;	Cpu ck delay (unit 0.15ns).
char SelQPLL1:1;	If 1 selected QPLL CK for ROB and TRB.
char SelQPLL2:1;	If 1 selected QPLL CK for SB.
char EnTrgPhi:1;	If 1 enabled phi trigger output.
char EnTrgThe:1;	If 1 enabled theta trigger output.
char EnTrgH:1;	If 1 enabled height quality trigger output, if 0 enabled low quality trigger out.
char DisTrbCk:1;	If 1 the TRB ck is disabled.
char DisSbCk:1;	If 1 the SB ck is disabled.
char DisOsc:1;	if 1 internal oscillator is disabled. The internal oscillator is used to generate CK for CPU when TTCrx CK is absent.
char L1A_Delay:7;	Auto trigger delay (7 bit).
char EnAutoTrg:1;	If 1 enabled auto trigger. The trigger out is sent with the L1A_Delay register delay to ROBs. The TTCrx L1A net is ignored.
char SelL1AVeto:1;	If 1 VETO input is used to generate L1A for ROBs, and the TTCrx L1A net is ignored.
char ForceTp:1;	If 1 the test pulse outputs are forced to level 1 (Test pulse sequence can't work in this state). Normally it is used only to read and calibrate signals amplitude during mini crate self test.

char CCBReady:1; State of line CCB Ready. This bit is active by Run in progress command and cleared if some alarm will be active .
The bit is cleared by: TTCrx lose look, QPLL lose look, Analog power fail, CK power fail, Vdd buffer fail, Vcc TRB fail, SB TSMD power fail, SB TSMS power fail, SB CK power fail, SB theta power fail, TRBs power fails, ROBs power fail, CCB power fail.

char RunInProgress:1; This bit is active by Run in progress command .When 1 CCB use only JTAG bus to read and write chips registers, and active also SEU test on trigger chips .

char CfgNotChanged:1; This bit is active by Run in progress command and cleared if mini-crate receive any command that change the configuration.

char CfgLoaded:1; This bit indicate that the flash configuration was loaded on all powered board, for the load/unload result see CfgLoadResult.
This two argument describe the following situation:

CfgLoaded	CfgLoadResult	Note
1	0	Flash configuration is loaded
1	0x80XX	Flash configuration is loaded but there are write chip error
0	From -2 to -10	Flash configuration not loaded
0	0	Mini-Crate have received configuration command

char InvalidArg:1; This bit indicate that at startup have received an invalid argument.

char TempTestDisabled:1; This bit indicate if temperature monitor is disabled (see command 0x83 Disable Temperature Test)

float Vccin_min; The min value read on Vccin.

float Vccin_max; The max value read on Vccin.

float Vddin_min; The min value read on Vddin.

float Vddin_max; The max value read on Vddin.

float Sb_Vcc_min; The min value read on Sb_Vcc.

float Sb_Vcc_max; The max value read on Sb_Vcc.

float Sb_Vdd_min; The min value read on Sb_Vdd.

float Sb_Vdd_max; The max value read on Sb_Vdd.

int ChamberMap; bit(15..12) always zero, bit(11..8) wheel (-2, -1, 0, +1, +2), bit(7..4) sector (1, 2,…), bit(3..0) station (1, 2, 3, 4). Sector 13 is used to identified MB4_4 near the sector 5 and Sector 14 is used to identifier MB4_10 near the sector 11.

Int CfgLoadResult; The result of loaded configuration

0	Success
-2	Power FLASH fail
-3	Invalid file size. File not found
-4	CRC error, data corrupted
-5	Invalid mode. Software error.
-6	Read error. Software error.
-7	Invalid update mode. Software error
-8	Invalid file version. Data saved with old version format
-9	Board changed. Invalid sensor code
-10	Mini-Crate was forced do not load configuration
0x80XX	The configuration is loaded but the seven less significant bits indicate an write chip error: bit0 for TTCrx; bit1 for SB (TSM chips); bit2 for TRACO LUT; bit3 for TRB (BTI, TRACO, TSS chips); bit4 for BTI on TRB theta; bit5 for TDC chips; bit6 for Front-Ends;

long TDCsStatusFlags; Content the accumulated result for TDC status register check, one bit every TDC. Active when RunInPorgess is 1.
Use code 0xA1 to clear.

long PoweMask; The mask of the boards that must not be powered at mini-crate power on. If mask differ from 0 the CfgLoadResult can't report 0x80XX errors. Bits identifier, 2=Analog, 3=CCB_CK, 4=RPC, 5=LED, 6=TSMS, 7=TSMDU, 8=TSMDD, 9=SB_THETA, 10=SB_CK, 11=BUFF_TRB, 12=VCC_TRB, 13=TRB0 to 20=TRB7, 21=ROB0 to 27=ROB6.

Exit, code 0xA5:
Exit and return to BOOT program. This command don't modify mini crate state

Return data:
char 0xFC;
char 0xA5;

Write BTI 8bit, code 0x14:
Arguments:
char brd; Board number form 0 to 7 (6 and 7 are trb_theta)
char chip; BTI chip form 0 to 31, bit[0..1] bti number, bit[2] inner/outer 0/1, bit[3..4] traco number.
char conf[11]; CONF register (8bit format).
char testin[13]; TESTIN register (8bit format).
Return data:
char 0xFC;
char 0x14;
char result; Return 1 if successful, return zero on verify error, return -1 on parameter error.

Read BTI 8bit, code 0x19:

Arguments:

char brd;	Board number form 0 to 7 (board 6 and 7 are trb_theta)
char chip;	BTI chip from 0 to 31, bit 0,1 bti number, bit 2 inner/outer 0/1, bit3,4 traco number.

Return data:

char 0x1A;	
char conf[11];	CONF register (8bit format).
char testin[13];	TESTIN register (8bit format).
char testout[13];	SNAP register (8bit format).

Return data on error:

char 0xFC;	
char 0x19;	
char narg;	return the negative number of the wrong argument.

Write BTI 6bit, code 0x54:

Arguments:

char brd;	Board number form 0 to 7 (board 6 and 7 are trb_theta)
char chip;	BTI chip from 0 to 31, bit[0..1] bti number, bit[2] inner/outer 0/1, bit[3..4] traco number.
char conf[14];	6bit format CONF register.
char testin[17];	6bit format TESTIN register.

Return data:

char 0xFC;	
char 0x54;	
char result;	Return 1 if successful, return zero on verify error, return -1 on parameter error.

Read BTI 6bit, code 0x59:

Arguments:

char brd;	Board number form 0 to 7 (6 and 7 are trb_theta)
char chip;	BTI chip from 0 to 31, bit(0..1) bti number, bit(2) inner/outer 0/1, bit(3..4) traco number.

Return data:

char 0x5A;	
char conf[14];	6bit format CONF register.
char testin[17];	6bit format TESTIN register.
char testout[17];	6bit format SNAP register.

Return data on error:

char 0xFC;	
char 0x59;	
char narg;	return the negative number of the wrong argument.

Enable BTI, code 0x51:

Enable BTI inputs.

This command modify the configuration of BTI

Arguments:

int brd;	Board from 0 to 7.
int inner_msk;	Mask enable of the inner BTI.
int outer_msk;	Mask enable of the outer BTI.

Return data:

char 0xFC;	
char 0x51;	
char result;	Return 1 if successful, return zero on BTI verify error, return -1 on parameter error.

Load BTI emulation trace, code 0x52:
Load emulation wire timing on the BTI.
This command modify the configuration of BTI
Arguments:

char brd;	TRB from 0 to 7.
char bti;	BTI from 0 to 31.
int trace[9];	Wire timing (step 12.5ns).

Return data:

char 0xFC;	
char 0x52;	
char result;	Return 1 if successful, return zero on BTI verify error, return -1 on parameter error.

Write TRACO, code 0x15:
Arguments:

char brd;	Board number form 0 to 5
char chip;	TRACO chip.
char conf[10];	CONF register
char testin[28];	TESTIN register

Return data:

char 0xFC;	
char 0x15;	
char result;	Return 1 if successful, return zero on verify error, return -1 on parameter error.

Read TRACO, code 0x1B:
Arguments:

char brd;	Board number form 0 to 5
char chip;	TRACO chip.

Return data:

char 0x1C;	
char conf[10];	CONF register
char testin[28];	TESTIN register
char testout[22];	SNAP register

Return data on error:

char 0xFC;	
char 0x1B;	
char narg;	return the negative number of the wrong argument.

Write Mini-Crate LUTs, code 0xA8:
Calculate all TRACO LUTs for the Mini-Crate and write it.
This command need about 45 seconds to terminate. See Appendix D for details.
Arguments:

int ST;	TRACO BTIC parameter.
float d;	Distance vertex – normal, unit cm.
float Xcn;	Distance first correlator – normal, unit cm.
int wheel;	Wheel sign (-1 or +1).

Return data:

char 0xFC;	
char 0xA8;	
long error;	return 0 on success, return -2 if DAQ run is in progress and the LUTs are not wrote, return error > 0 on write error, any bit to 1 from bit 0 to bit 24 identified the TRACO write fault (6 board * 4 TRACO=24).

Read Mini-Crate LUT parameters, code 0xA9:
Read the parameters wrote with command code 0xA8.
See Appendix D for details.

Return data:

char 0xAA;	
int ST;	TRACO BTIC parameter.
float d;	Distance vertex – normal, unit cm.
float Xcn;	Distance first correlator – normal, unit cm.
int wheel;	Wheel sign (-1 or +1).

Preload TRACO LUT, code 0x4D:
This command can be used to directly write the TRACO LUT, that must before preloaded it in to RAM. Use command code 0x4E to transfer LUT from RAM to TRACO.
Arguments:

char brd;	TRB number from 0 to 5
char traco;	TRACO number from 0 to 3
char ang;	0=pos LUT, 1=ang LUT
int addr;	LUT location address (from 0 to 1024-size for angle LUT and from 0 to 1536-size for position LUT).
int data[size];	LUT data array (array size from 1 to 64)

Return data:

char 0xFC;	
char 0x4D;	
char result;	return 1 if successful.

Load TRACO LUT, code 0x4E:
Transfer preload LUT in to TRACO.
Arguments:

char brd;	TRB number from 0 to 5
char traco;	TRACO number from 0 to 3

Return data:

char 0xFC;	
char 0x4E;	
char result;	Return 1 if successful, return zero on verify error, return -1 on parameter error.

Read TRACO LUT, code 0x86:
Read the LUT written on the TRACO chip.
Arguments:

char board;	The TRB board number (from 0 to 5).
char chip;	The chip number (from 0 to 3)
char ang;	If zero return POS LUT, if 1 return ANG LUT.
int addr;	LUT Address, form 0 to 1024-size for angle LUT and from 0 to 1536-size for position LUT.
char size;	The number of word (10 bits for angle LUT, 12 bits for position LUT) to read (from 1 to 64).

Return data:

char 0x87;	
int data[size];	

Return data on error:

char 0xFC;	
char 0x86;	
char error;	return -1.Indicate an argument error.

Write TSS, code 0x16:
Arguments:

char brd;	Board number form 0 to 5
char conf[7];	CONF register
char testin[20];	TESTIN register

Return data:

char 0xFC;	
char 0x16;	
char result;	Return 1 if successful, return zero on verify error, return -1 on parameter error.

Read TSS, code 0x1D:
Arguments:

char brd;	Board number form 0 to 5

Return data:

char 0x1E;	
char conf[7];	CONF register
char testin[20];	TESTIN register
char testout[32];	SNAP register

Return data on error:

char 0xFC;	
char 0x1D;	
char narg;	return the negative number of the wrong argument.

Write TSM, code 0x17:
Arguments:

char conf_s[2];	CONF register sorter
char conf_u[2];	CONF register Data-Up
char conf_d[2];	CONF register Data-Down

Return data:

char 0xFC;	
char 0x17;	
char result;	Nonzero if the function is successful

Read TSM, code 0x1F:
Return data:

char 0x20;	
char conf_s[2];	CONF register Sorter
char conf_u[2];	CONF register Data-Up
char conf_d[2];	CONF register Data-Down

Write TDC Control, code 0x18:
Arguments:

char brd;	Board number from 0 to 6
char chip;	TDC number from 0 to 3
char control[5];	Control register

Return data:

char 0xFC;	
char 0x18;	
char result;	Return 1 if successful, return zero on verify error, return -1 on argument error.

Write TDC, code 0x44:
Write TDC configuration. The control sequence is automatically execute by CCB.
Arguments:

char brd;	Board number from 0 to 6
char chip;	TDC number from 0 to 3
char setup[81];	Setup register
char control[5];	Control register

Return data:

char 0x45;	
char status[8];	Status register

Return data on error:

char 0xFC;	
char 0x44;	
char narg;	return the negative number of the wrong argument.

Read TDC, code 0x43:
Arguments:

char brd;	Board number from 0 to 6
char chip;	TDC number from 0 to 3

Return data:

char 0x22;	
char setup[81];	Setup register, this is the value stored in the CCB memory
char control[5];	Control register, this is the value read from the chip.
char status[8];	Status register, this is the value read from the chip.
char idcode[4];	Id code register, this is the value read from the chip.

Return data on error:

char 0xFC;	
char 0x43;	
char narg;	return the negative number of the wrong argument.

TDC RunBist , code 0x41:
Execute the BIST test for all TDC.
Arguments:

char brd;	board number from 0 to 6.

Return data:

char 0x42	
char bist[4][2];	BIST test result.

Return data on error:

char 0xFC;	
char 0x41;	
char error;	return -1 to indicate an argument error.

Check TDCs Status, code 0xBC:
Check the TDCs status register.

Return data:

char 0xBD;	
long TDCsStatusFlags;	The check result of all TDC status register, one bit for TDC.

Rob Reset, code 0x32:
Reset ROBs.

Return data:

char 0xFC;	
char 0x32;	

Read ROBs error, code 0x33:
Read state of the ROBs.

Return data:
char 0x34;
char state[7]; The 7 ROB state, Error if zero. The ROB error a single line three state signal for all ROBs and it is used by the single ROB addressed by JTAG address line, On CCB there is a pull-up so if ROB is off or not present the error line is always to high level.

Read ROB Power, code 0x5B:
Read ROBs voltage and current. The value returned are the result of a conversion executed by an independent process with lower priority level, it is executed when CCB is in idle state, the conversion time is about 3 second if CCB not receive command.

Return data:
char 0x5C;
float Vcc[7]; Volt.
float Vdd[7]; Volt.
float current[7]; Ampere.

ROB Read Out, code 0x6C:
Read the TDC data by JTAG bus (see TDC manual).
Arguments:
char brd; Board number form 0 to 6.
Return data:
char 0x6D;
char more_data; if 1 there are more data;
long readout[size]; The readout data (size from 0 to 60).
Return data on error:
char 0xFC;
char 0x6C;
char error; return -1 to indicate an argument error.

Write TTCrx, code 0x27 or code 0xE3:
Arguments:
char addr; I2C register address
char data; data
Return data:
char 0xFC;
char 0x27 or 0xE3;
char result; OK=0, no_ack=1, SDA error=-1, SCL error=-2

Read TTCrx, code 0x28 or code 0xE2:
Arguments:
char addr; I2C register address
Return data:
char 0x29;
char data; Data read, valid only if result=0.
char result; OK=0, no_ack=1, SDA error=-1, SCL error=-2

TTC Skew, code 0x48:
This command must be used to change the phase between beam and BTI CK. This command respect the phase between CKDES1 and CKDES2 of the TTCrx used for internal mini crate timing.
Arguments:
char delay; Delay unit=1ns;
Return data:
char 0xFC;
char 0x48;
char error; return 0 if successful.

TTC fine delay, code 0x8E:
This command calculate the correct value to write in the TTCrx register 0 or 1 to modify the two TTCrx fine delay. This command must be used only to modify internal mini crate timing (ch0=2ns ch1=0ns for correct timing). Use command code 0x48 to modify phase between beam and BTI CK.
Arguments:

char ch;	0= TTCrx fine delay1, 1=TTCrx fine delay2
float delay;	Delay (ns) from 0 to 24.9;

Return data:

char 0xFC;	
char 0x8E;	
char error;	return 0 if successful.

Select TTC CK, code 0x2A:
Select CK type.
Warning: There is a small difference signals phase between QPLL CK and TTCrx CK of some nano seconds. If QPLL1==QPLL2 only BTI to beam phase change. If QPLL1!=QPLL2 internal mini crate timing must be corrected by tuning TTCrx fine delay1 and fine delay2.
Arguments:

char QPLL1:1;	If 1 this bit select QPLL CK for ROB and TRB, otherwise use TTCrx CK.
char QPLL2:1;	If 1 this bit select QPLL CK for SB, otherwise use TTCrx CK.
char EnTtcCkMux:1;	This bit if zero force the CPU to work with internal 40MHz CK but ROB, TRB and SB use always TTCrx or QPLL CK .

Return data:

char 0xFC;	
char 0x2A;	

Emulation TTC, code 0x50:
Emulate TTC commands.
Arguments:

char cmd;	Command: 0=sequence advance, 1=sequence test (test-pulse), 2=RESET CCB, 3=sequence reset.

Return data:

char 0xFC;	
char 0x50;	

Clear TTCrx Lose lock counters, code 0x8A:
This command clear all PLL lose lock counters.
Return data:

char 0xFC;	
char 0x8A;	

Front-end Test, code 0x4B:
Find the front-ends board.

Arguments:

char override	Optional argument, if 1 override the Chamber Map info (see command code 0xB4), this info are used by many front-end commands.

Return data:

char 0x4C;	
int Fe_TestI2C[3];	Front-end I2C bus diagnostic 0=OK, -1=SDA short circuit, -2=SCL short circuit, The elements order refer to physical hardware port.
int SlPresMsk[3];	Diagnostic on the three Front-end I2C bus (search chip PCF8574 on the three I2C bus), The elements order refer to physical hardware port.
long SlBoardPresMsk[3];	Presence of the board on the three superlayer, the elements order is the super layer order (0=inner, 1=theta, 2=outer).
long SlExBoardMsk[3];	Presence of the 20ch board on three superlayer, the elements order is the super layer order (0=inner, 1=theta, 2=outer).

Set Front-end threshold, code 0x35:

Arguments:

int sl;	super layer
float bias;	from 0.06 to 3.9 resolution 4.42mV (4.53/1024)
float threshold;	from 0.0 to 0.2 resolution 0.474mV (4.53/1024 * 1200/11200)

Return data:

char 0x25;	
float bias;	Bias set.
float threshold;	Threshold set
float adc_bias;	Bias read from ADC, resolution 4.42mV (4.53/1024).
float adc_threshold;	Threshold read from ADC, resolution 4.42mV (4.53/1024).

Return data on error:

char 0xFC;	
char 0x35;	
char error;	return 0 if successful, or the negative number of the wrong argument.

Set Front-end Width, code 0x47:

Arguments:

float width;	from 0.06 to 3.9 resolution 4.42mV (4.53/1024)

Return data:

char 0x24;	
float width;	Voltage set.
float adc_with	Voltage read from ADC.

Read Front-end Temperature, code 0x36:

Arguments:

char sl;	Superlayer 0=phi_inner, 1=theta, 2=phi_outer.
char opt;	-1= all temperature, 0= read only tmax and tmed.

Return data:

char 0x37;	
int temp[98];	unit=0.1°C, first element is Tmax, second element is Tmed. Real resolution is about 0.59°C (4.53/1024/0.0075)

Read Front-end Mask, code 0x38:

Return data:

char 0x39;	
char MadMsk[3][24][3];	State of the all masked channels. MadMask[sl][brd][bit msk]. The three byte mask order is: Bit0 of the byte0 is the fist FEB cannel, bit7 of the byte1 is the last FEB channel for 16 channels board and bit3 of the byte2 is the last FEB channel for 20 channels board.

Mask Front-end, code 0x3A:

Arguments:

int sl;	Superlayer 0=phi_inner, 1=theta, 2=phi_outer.
int brd;	Board from 0 to 24;
char mask[3];	channels mask. Bit0 of the byte0 is the fist FEB cannel, bit7 of the byte1 is the last FEB channel for 16 channels board and bit3 of the byte2 is the last FEB channel for 20 channels board.

Return data:

char 0x39;	
char MadMsk[3][24][3];	State of the all masked channels. MadMask[sl][brd][bit msk]. The three byte mask order is: Bit0 of the byte0 is the fist FEB cannel, bit7 of the byte1 is the last FEB channel for 16 channels board and bit3 of the byte2 is the last FEB channel for 20 channels board.

Mask Front-end channel, code 0x3B:

Arguments:

int sl;	Superlayer 0=phi_inner, 1=theta, 2=phi_outer.
int channel;	Channel
int stato;	0 or 1;

Return data:

char 0x39;	
char MadMsk[3][24][3];	State of the all masked channels. MadMask[sl][brd][bit msk]. The three byte mask order is: Bit0 of the byte0 is the fist FEB cannel, bit7 of the byte1 is the last FEB channel for 16 channels board and bit3 of the byte2 is the last FEB channel for 20 channels board.

Front-End Enable Max Temperature Chip, code 0x62:

Arguments:

int sl;	Super layer 0=phi_inner, 1=theta, 2=phi_outer.
int chip;	Chip number.
int stato;	0 or 1. If 1 the chip contribute to the Max temperature measure.

Return data:

char 0x63;	
char msk[3][24];	The actual mask for all super layer. msk[sl][brd].

Read Front-End Max Temperature Mask, code 0x71:

Return data:

char 0x63;	
char msk[3][24];	The actual mask for all super layer. msk[sl][brd].

Front-ends super layer enable, code 0x9A:

Enable Front-Ends channels.

Arguments:

char sl;	Super layer 0=phi_inner, 1=theta, 2=phi_outer.
char enable;	if 1 enable all channels.

Return data:

char 0xFC;	
char 0x9A;	
char error;	return 0 if success, sl not connected (no_ack) =1, SDA error=-1, SCL error=-2.

Front-ends super layer enable Max Temperature, code 0x9B:

Enable Front-Ends MaxTemperature.

Arguments:

char sl;	Super layer 0=phi_inner, 1=theta, 2=phi_outer.
char enable;	if 1 enable all sensor chip.

Return data:

char 0xFC;	
char 0x9B;	
char error;	return 0 if success, sl not connected(no_ack)=1, SDA error=-1, SCL error=-2.

Calibration DAC, code 0x6B:

Arguments:

float bias_gain[3];
float thr_gain[3];
float width_gain;
float bias_offset[3];
float thr_offset[3];
float width_offset;

Return data:

char 0xFC;
char 0x6B;

Read Calibration DAC, code 0x6E:

Return data:

char 0x6F;
float bias_gain[3];
float thr_gain[3];
float width_gain;
float bias_offset[3];
float thr_offset[3];
float width_offset;

Relative Test Pulse settings, code 0x78:

Arguments:

float delay;	Delay in nanoseconds between test-pulse output 1 and output 2, from -374.9 to 374.9. Resolution 0.150ns
float Amplitude1;	from 0.3 to 0.610
float Amplitude2;	from 0.3 to 0.610
char EndSqDly;	from 0 to 63, unit=25ns, from the end of greatest test-pulse coarse delay. This value must be bigger than 22 because the track sequence is disabled before go out the trigger output. work if it is less.
char FEDisDly;	from 0 to 15.

Return data:

char 0xFC;
char 0x78;

Test Pulse settings, code 0x4A:

This command is implemented only to have direct access to the test-pulse register, use command 0x78 to configure test-pulse.

Arguments:

char coarsedelay1;	from 0 to 15, unit=25ns (zero disable test-pulse).
char finedelay1;	from 0 to 255, unit=0.150ns.
char coarsedelay2;	from 0 to 15,unit=25ns (zero disable test-pulse).
char finedelay2;	from 0 to 255, unit=0.150ns
char endSqDly;	from 0 to 63, unit=25ns, from the end of greatest test-pulse coarse delay.
float Amplitude1;	from 0.3 to 0.610
float Amplitude2;	from 0.3 to 0.610
char FEDisDly;	from 0 to 15.
char fineofs[2];	Optional argument : Fine offsets from 0 to 40 (unit=0.150ns) . This values are used to calibrate fine delay when use code 0x78.

Return data:

char 0xFC;
char 0x4A;

Read Test Pulse settings, code 0x72:

Return data:

char 0x73;	
char coarsedelay1;	from 0 to 15, unit=25ns
char finedelay1;	from 0 to 255, unit=0.150ns (value stored on delay chip).
char coarsedelay2;	from 0 to 15,unit=25ns.
char finedelay2;	from 0 to 255, unit=0.150ns (value stored on delay chip).
char endSqDly;	from 0 to 63, unit=25ns, from the end of greatest test-pulse coarse delay.
float Amplitude1;	from 0.3 to 0.610 Volt
float Amplitude2;	from 0.3 to 0.610 Volt
char FEDisDly;	from 0 to 15, unit=25ns
char fineofs1;	fine offset, unit 0.150ns (addend to delay when use 0x78 code).
char fineofs2;	fine offset, unit 0.150ns (addend to delay when use 0x78 code).

Set Test Pulse Offset, code 0x8B:

Arguments:

char TpOffset[8];	Delay position (from min to max value defined below) used in relative test pulse delay command, code 0x78. One element for mini-crate types. This values are used to regulate the phase between test pulse and BTI ck.
char fineofs[2];	Fine offsets from 0 to 20 (unit=0.150ns) . This values are added to calibrate difference between the two fine delay output when use command code 0x78.
char min;	Fine delay min value (default 20), max-min must be bigger than 167.
char max;	Fine delay max value (default 235), max-min must be bigger than 167.

Return data:

char 0xFC;
char 0x8B;

Read Test Pulse Offset, code 0x8C:

Return data:

char 0x8D;	
char TpOffset[8];	Delay position used in relative test pulse delay command, code 0x78. One element for mini-crate types.
char fineofs[2];	Fine offsets from 0 to 40 (unit=0.150ns) . This values are added to calibrate difference between fine delay when use command code 0x78.
int min;	Fine delay min value.
int max;	Fine delay max value.

Power On/Off, code 0x2F:

Power on board and initialize it with the memory copy of the board registers(only ROBs and TRBs), or power on SB/CCB blocks.

Arguments:

char pwr_id;	Power identifier, -1=All, 1=FLASH, 2=Analog, 3=CCB_CK, 4=RPC, 5=LED, 6=TSMS, 7=TSMDU, 8=TSMDD, 9=SB_THETA, 10=SB_CK, 11=BUFF_TRB, 12=VCC_TRB, 13=TRB0 to 20=TRB7, 21=ROB0 to 27=ROB6. See appendix E.
char on;	0=off, 1=on.

Return data:

char 0xFC;	
char 0x2F;	
char result;	pwr_on=1, pwr_off=0, pwr_already_off=2, pwr_already_on=3, , pwr_on_fail=-1, pwr_fail=-2, invalid_fault=-3 (problem on line fault), invalid_id=-4. For ROBs if the result is -3 there is one board in fail state and CCB can't verify power on result, because fault flag is the OR of all boards.
char updateresult;	This argument is valid only if pwr_id is ROBx and TRBx and if the result=pwr_on. Indicate the configuration transfer result from RAM to board chips. OK=1 FAIL=0.

Power Mask, code 0x30:

This command is used to mark a broken board. You must save the configuration on flash to take effect. The mask can be read in the status data structure.

Arguments:

char pwr_id;	Power identifier, 2=Analog, 3=CCB_CK, 4=RPC, 5=LED, 6=TSMS, 7=TSMDU, 8=TSMDD, 9=SB_THETA, 10=SB_CK, 11=BUFF_TRB, 12=VCC_TRB, 13=TRB0 to 20=TRB7, 21=ROB0 to 27=ROB6. See appendix E.
char value;	1=board not powered at mini-crate power on.

Return data:

char 0xFC;
char 0x30;

CPU CK Delay, code 0x26:
This command is used to regulate the phase between JTAG CK an ROB CK.
Default delay value is 0. WARNING: this command have a small probability to generate a glitch on CPU ck with the consequent CPU crash.
Arguments:

char delay;	Delay unit 150ps

Return data:

```
char 0xFC;
char 0x26;
```

SYNC, code 0x2B:
Activate for 0.1second the Sync line of the SB links

Return data:

```
char 0xFC;
char 0x2B;
```

Snap Reset, code 0x2C:
Activate for about 0.002 second the SnapReset line.

Return data:

```
char 0xFC;
char 0x2C;
```

Soft Reset, code 0x2D:
Activate for about 0.002 second the SoftReset line. This line reset all CONFIG registers.

Return data:

```
char 0xFC;
char 0x2D;
```

Mini-Crate Temperature, code 0x3C:
Read MINI CRATE temperature. The return value is the result of the conversion executed by a independent process with lower priority level, it is executed when CCB is in idle state, the conversion time is about 3 second if CCB not receive commands.

Return data:

char 0x3D;	
float temp[20];	Elements from 0 to 6 are ROB temperature, elements from 7 to 14 are TRB temperature, elements from 15 to 19 are external sensors.
char code[20][8];	Sensor Identifier code, elements from 0 to 6 are ROB temperature sensors, elements from 7 to 14 are TRB temperature sensors, elements from 15 to 19 are external sensor. If Mini crate type is invalid value, the TRB order following the hardware map connection on SB.

Mask Mini-Crate Temperature, code 0xBB:
Mask TRBs ROBs temperature sensor in the max temperature monitoring.
Arguments:

int mask;	bit from 0 to 6 mask ROB sensor, bit from 7 to 14 mask TRB sensor. Use mask<0 if you want only read the actual mask.

Return data:

char 0xFC;	
char 0xBB;	
int mask;	The actual Mask;

Save Mini-crate configuration, code 0x40:
This command save in flash memory the configuration of the mini crate.
This configuration is automatically read at startup (no argument in the script command).
WARNING: the use of this command is limited by the flash endurance >10000 write cycles.
For TRACO are saved only the parameter to calculate the LUT (see command 0x3E).

Return data:
char 0xFC;
char 0x40;
int error; return 0 if successful.
Errors code:

0	Success
-1	Error writing data on FLASH
-2	Power FLASH fail
-5	Invalid mode. Software error.
-7	Invalid update mode. Software error

Load Mini-Crate configuration, code 0x61:
Load from FLASH memory the MINI CRATE configuration.
The command need about 1 minute to restore the configuration.
During the execution if mini-crate receive commands it respond with a busy.
If a board is power off the data is write only in the memory copy of the board registers.

Return data:
char 0xFC;
char 0x61;
int error; Return 0 if successful.
Errors code:

0	Success
-2	Power FLASH fail
-3	Invalid file size. File not found
-4	CRC error, data corrupted
-5	Invalid mode. Software error.
-6	Read error. Software error.
-7	Invalid update mode. Software error
-8	Invalid file version. Data saved with old version format
-9	Board changed. Invalid sensor code
0x80XX	The configuration is loaded but the seven less significant bits indicate an write chip error: bit0 for TTCrx; bit1 for SB (TSM chips); bit2 for TRACO LUT; bit3 for TRB (BTI, TRACO, TSS chips); bit4 for BTI on TRB theta; bit5 for TDC chips; bit6 for Front-Ends;

Get Configuration CRC, code 0xA3:
This command calculate the CRC of the current mini-crate configuration.

Return data:
0xA4;
int crc; CRC of the current mini-crate configuration.
int crcTRG; CRC of the Trigger configuration.
int crcRO; CRC of the Read-Out configuration.
Int crcFE; CRC of the Front-Ends configuration
char error; return 0 if success.

Run in Progress , code 0x46:
Communicate to mini crate that DAQ is running or not. When the argument “on” is 1 the CCB active CCB_READY line on the trigger bus data and use only JTAG bus to read and write chips registers, and active also CfgNotChanged flag on status data structure to signal if configuration was changed after this command.
This command active also SEU test on TRB chips.
The CCB_READY line is cleared by CCB if:
TTCrx lose look, QPLL lose look, Analog power fail, CK power fail, Vdd buffer fail, Vcc TRB fail, SB TSMD power fail, SB TSMS power fail, SB CK power fail, SB theta power fail, TRBs power fails, ROBs power fail, CCB power fail.
Arguments:

char on;	DAQ status.

Return data:
char 0xFC;
char 0x46;

Trigger Out Select, code 0x49:
This command select the trigger type place on line TRGOUT on the trigger data and on mini-crate output connector
Arguments:

char EnTrg_phi;	Enable trigger phi.
char EnTrg_theta;	Enable trigger theta.
char HTrg;	Enable High quality trigger.

Return data:
char 0xFC;
char 0x49;

RPC I2C Commands, code 0x64:
Execute I2C commands sequence on RPC bus .
Arguments:

int ctrl_data[size];	The bit8 is the start flag, bit9 is stop flags, bit10 is read flags, bit11 is ackn flag, bit12 is abort on error, the bits from 0 to 7 are the write data and will be ignored if read flag is 1. Size can be from 1 to 100. If start flag is 1 will be executed I2C start sequence before the read or write data on I2C bus. If stop flag is 1 will be executed I2C stop sequence after the read or write data on I2C bus. If read flag is 0 ackn flag is ignored and will be executed a write command with the data bit(7..0). If read flag is 1 will be executed a I2C read command with I2C ACKN bit set to the state of ackn flags.

Return data:
char 0x65;

char err_data[size] ;	The bits from 0 to 7 are the read data if read flag was 1, or wrote data if read flags was 0. The bits from 8 to 15 are the I2C bus diagnostic 0=OK, 1=WRITE error (the I2C ACKN bit was 1) , -1 =SDA short circuit, -2=SCL short circuit. The size is the same of the sent data if you have not enabled abort on error.

LED Alignment I2C Commands, code 0x66:
Execute I2C commands sequence on Led alignment bus .
Arguments:

int ctrl_data[size];	The bit8 is the start flag, bit9 is stop flags, bit10 is read flags, bit11 is ackn flag, bit12 is abort on error , the bits from 0 to 7 are the write data and will be ignored if read flag is 1. Size can be from 1 to 100. If start flag is 1 will be executed I2C start sequence before the read or write data on I2C bus. If stop flag is 1 will be executed I2C stop sequence after the read or write data on I2C bus. If read flag is 0 ackn flag is ignored and will be executed a write command with the data bit(7..0). If read flag is 1 will be executed a I2C read command with I2C ACKN bit set to the state of ackn flags.

Return data:

char 0x67;	
int err_data[size] ;	The bits from 0 to 7 are the read data if read flag was 1, or write data if read flags was 0. The bits from 8 to 15 are the I2C bus diagnostic 0=OK, 1=WRITE error (The I2C ACKN bit was 1) , -1=SDA short circuit, -2=SCL short circuit. The size is the same of the sent data if you have not enabled abort on error.

Read Com. Error, code 0xF0:
Read and clear the first error on communications ports.

Return data:

char 0xF0;	
char port;	primary port=1. secondary port=2
char Parity:1;	
char Framing:1;	
char Break:1;	
char Noise:1;	
char Overrun:1;	UART overrun.
char Sync:1;	Invalid synchronization char.
char Crc:1;	CRC error.
char LoseData:1;	Software overrun .
char Size:1;	Invalid frame size.
char TimeOut:1;	Receive frame timeout.
char Unexpected:1;	This bit indicate a software error in the firmware.
char BuffOverflow:1;	
char BufferEmpty:1;	
char BufferTooSmall:1;	
char unused:2;	

Auto Trigger, code 0x70:
Enable internal mini-crate auto trigger. The trigger output signal is sent to the LV1A of the mini-crate and TTCrx LV1A is ignored.
Arguments:

char en;	Enable Auto trigger (0=disabled 1=enabled)
char L1A_delay:7;	L1A delay (7bit), unit 25ns.

Return data:

char 0xFC;
char 0x70;

Auto Set Link, code 0x74:
This command set the threshold of the optical link discriminator.
It is better send the command by secondary port.

Return data:

char 0x75;	
int Offset;	Optical link output offset measured in DAC count(Vref=4.53V, DAC bits=10).
int Hyst;	Optical link discriminator hysteresis measured in DAC count (Vref=4.53V, DAC bits=10).
int Apl;	Optical link signal amplitude measured in DAC count (Vref=4.53V, DAC bits=10) .
int Thr;	Optical link discriminator threshold set in DAC count (Vref=4.53V, DAC bits=10).

Read Link data, code 0x76:
Read the optical link discriminator settings.

Return data:

char 0x75;	
int Offset;	Optical link output offset measured in DAC count(Vref=4.53V, DAC bits=10).
int Hyst;	Optical link discriminator hysteresis measured in DAC count (Vref=4.53V, DAC bits=10).
int Apl;	Optical link signal amplitude measured in DAC count (Vref=4.53V, DAC bits=10) .
int Thr;	Optical link discriminator threshold set in DAC (Vref=4.53V, DAC bits=10).

Disable Link monitor, code 0x77:
Arguments:

char disable;	if 1 disable the automatic procedure to set up optical link discriminator threshold. See special code 0xAB and 0xAC.

Return data:
char 0xFC;
char 0x77;

Set Timeouts Link monitor, code 0x96:
Arguments:

float noidle_to;	Not Idle Timeout, in seconds to run auto link setup procedure from 120 to 7200 (default 600sec)
float idle_to;.	Idle Timeout, in second to run auto link setup procedure from 30 to 600 (default 30sec) .

Return data:

char 0xFC;	
char 0x96;	
char error;	return -1 to indicate an argument error.

Disable Temperature Test, code 0x83:
This command disable temperature test of the TRB/ROB boards for 1hour.
Arguments:

char disable;	If 1 disable temperature test, if 0 enable temperature test.

Return data:
char 0xFC;
char 0x83;

Read Mini crate sensors ID, code 0x8F:
Read MINI CRATE temperature sensors identified code . Identified code have 64 bit format, see DS2438 and DS18S20 data sheets.

Return data:
char 0x90;
char code[21][8]; Elements from 0 to 6 ore ROB temperature sensors identified code, elements from 7 to 14 are TRB temperature sensors identified code, elements from 15 to 20 are external sensors identified code. If Mini crate type is invalid value the TRB order following the hardware map connection on SB.

Disable TRB CK, code 0x93:
Disable the TRB CK
Arguments:
char state; 1=disable.
Return data:
char 0xFC;
char 0x93;

Disable SB CK, code 0x94:
Disable the SB CK
Arguments:
char state; 1=disable.
Return data:
char 0xFC;
char 0x94;

Disable Oscillator CK, code 0x95:
Disable the free running oscillator CK
Arguments:
char state; 1=disable.
Return data:
char 0xFC;
char 0x95;

Write custom data, code 0x97:
The data are stored in flash memory with Save command.
Arguments:
char data[size]; custom data, size must be <=240
Return data:
char 0xFC;
char 0x97;

Read custom data, code 0x98:
Read custom data.

Return data:
char 0x99;
char data[size]; size <=240;

Clear min max voltage monitor, code 0x9E:
This command clear the min and max value of the mini crate and front end input voltage.

Return data:
0xFC;
0x9E;

Select L1A / Veto Mode, code 0x9F:
Select the use of veto input
Arguments:
char mode; 1 for L1A mode, 0 for VETO mode.
Return data:
0xFC;
0x9F;

Clear TRB SEU Counters, code 0xA1:
This command clear the TRB Seu counters and RAMs SEU.

Return data:
0xFC;
0xA1;

Clear Trigger Histogram, code 0xAD:
This command clear the histogram of the mini-crate trigger.

Return data:
0xFC;
0xAD;

Read Trigger Histogram, code 0xB0:
Read the trigger histogram data. The histogram not include hit generated by test-pulse trigger.
The histogram is constitute by 4096 16 bits counters , one for every bunch crossing number.
The histogram is generate by software spying the mini-crate trigger output.
Arguments:
int offset; offset must be from 0 to 3996.
Return data:
0xB1;
int data[100]; the data read from offset .
Return data on error:
char 0xFC;
char 0xB0;
char error; return -1 to indicate an argument error.

Trigger Frequency, code 0xB2:
This command measure the trigger out frequency (test-pulse excluded). If *fastmode* argument is 1 or no argument is send, the command measure the trigger out frequency for a period of 0.5second .
The frequency counter is software implemented so too near trigger are lost (max frequency is about 42KHz).
WARNING: during the measure period in fast mode all interrupts are disabled so the mini-crate can't receive command (the commands can be lost).
If *fastmode* argument is 0 the measure is executed in low priority mode, for a period specified on timeout argument. To read the measure result send the command with timeout argument equal to 0.0 so return a long value that if it is negative indicate that the measure is not terminated.
The test pulse triggers are masked by CAL signal that mark the test pulse sequence, but setting the EndSqDly parameter less than 21 (see command 0x78) is possible to see also this triggers because CAL signal is cleared before the trigger go out, but this change the trigger and read-out data output by mini-crate because is not active the test-pulse sequence filter .
Arguments:
char fastmode; If fast mode is 1 return immediately the measure (see WARING in the description command).
float timeout; Measure time, Ignored if fast mode is 1. If timeout is 0.0 return the current measure value. If timeout is negative return the number of trigger accumulated in the trigger histogram.
Return data with fastmode=1:
0xB3;
unsigned int frequency; The trigger frequency measured.
Return data with fastmode=0:
0xB3;
long ntrigger; The number of trigger accumulated. If negative or zero the measure is not terminated.

Chamber Map, code 0xB4:
This command is used to map the mini-crate with the Chamber position.
Mini-crate use this info to know the number of the FEBs , number of ROBs etc..
If chamber map is not used or saved on configuration, mini-crate use the info returned from command Front-end Test code 0x4B.

Arguments:

int id	bit(15..12) reserved must be zero, bit(11..8) wheel (-2, -1, 0, +1, +2), bit(7..4) sector (1, 2,…), bit(3..0) station (1, 2, 3, 4). Sector 13 is used to identified MB4_4 near the sector 5 and Sector 14 is used to identifier MB4_10 near the sector 11.

Return data:

char 0xFC;	
char 0xB4;	
int error;	1=invalid sector, 2=invalid station, 4=invalid wheel, 8=station conflict with auto-detect mini-crate type, 16=id not found on info table.

Retransmit , code 0xB5:
This command can be used to request the last transmission data. Primary an secondary port, copy the transmitted data on a independent buffer, when mini-crate receive this command it transmit the content of this buffer. On the secondary port can be transmitted a broadcast command and the result of the command of the single mini-crate can be read by this command. In the BOOT program this command in not implemented. The special code Split is ignored for this command.

SEU Mask , code 0x2E:
This command can be used to mask the damaged chips during SEU monitoring.
Arguments:

char board;	The board number, from 0 to 5 for TRB_PHI and from 6 to 7 for theta board.
long bitmask;	The BTI mask, one bit for chips.
char tracomask;	The TRACO mask, one bit for chips. Ignored for theta board.
char tssmask;	The TSS mask. Ignored for theta board.

Return data:

char 0xFC;	
char 0x2E;	
char error;	return -1 to indicate an argument error.

Read SEU Mask , code 0xB9:
This command can be used read back the masked chips during SEU monitoring and to see on which chip the SEU is found.
Arguments:

char board;	The board number, from 0 to 5 for TRB_PHI and from 6 to 7 for theta board.

Return data:

char 0xBA;	
long btimask;	The BTI mask, one bit for chips.
char tracomask;	The TRACO mask, one bit for chips. Always 0 for theta board.
char tssmask;	The TSS mask. Always 0 for theta board.
long btierr;	The BTI that have generated the SEU.
char tracoerr;	The TRACO that have generated the SEU. Always 0 for theta board.
char tsserr;	The TSS that have generated the SEU. Always 0 for theta board

Return data on error:

char 0xFC;	
char 0xB9;	
char error;	return -1 to indicate an argument error.

Check SEU LUT , code 0x3E:
This command can be used to verify SEU on TRACO LUT. The number of SEU is reported on status data structure. The LUT are not corrected, you must rewrite it.

Return data:
char 0xFC;
char 0x3E;
char result; return 1 if there isn't SEU;

Protect/Unprotect FLASH, code 0xED:
This command is the same command on the BOOT program, and must be used only for firmware update. This command active or deactivate the write protection on the FLASH memory, and if necessary power on the flash before.

Arguments:
char protect; if 1 Enable write protection.
Return data:
char 0xFC;
char 0xED;

Write FLASH, code 0xF3:
This command is the same command on the BOOT program, but there are not the same limitations.
The command must be used only for firmware update.
To write a sector (256 byte) on the flash you must send two consecutive write command with consecutive address, the second one write physically the flash sector.
The command is executed only if the data sent is contents in a single flash sector.

Arguments:
char page; 8 bits address high.
int addr; 16 bits address low.
char data[size]; The data to write, size from 0 to 200.
Return data:
char 0xF2;
char error; return 0 if write success; -1 write flash error, -2 non consecutive data, -3 bad address (not flash address), -4 data overlap between two sectors.
char sector; return 0xFF for 256 byte flash sector(AT29C020), or 0x00 for 128 byte flash sector (AT29C010).

Test commands:

Find Sensors, code 0xA6:
Find all sensor connected to 1wire bus and return Identified code (64 bit format), see DS2438 and DS18S20 data sheets.
This command is used for diagnostic but can be used to find external sensor connected after power on mini-crate.

Return data:
char 0xA7;
char code[n][8]; Identified code of the sensor, n is the number of sensors found.

Start emulation, code 0x53:
This command modify the configuration of the BTI.
This command generate a pulse on CCB Ready line.
Arguments:
char brd; TRB Board from 0 to 7.
char bti; BTI from 0 to 32.
Return data:
char 0xFC;
char 0x53;
char result; Return 1 if successful, return 0 on parameter error.

New Start emulation , code 0x79:
This new command can generate multiple trace on the same TRB.
This command modify the configuration of the BTI.
This command generate a pulse on CCB Ready line.
Arguments:

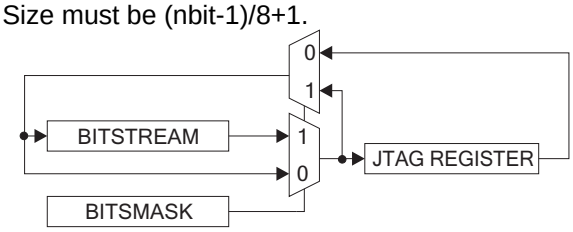
char brd;	TRB Board from 0 to 7.
long btimsk;	BTI mask.

Return data:

char 0xFC;	
char 0x79;	
char result;	Return 1 if successful, return 0 on parameter error.

JTAG Preset Data, code 0x55:
Preset data for JTAG sequence.
For the SB, execute SB_SetUpJtagChain command before of the JTAG_PresetData, because the JTAG chain change if the chips on the board are powered on or off.
Warning: This command are reserved for expert users. An improper user of the boundary scan can produce short circuit on bidirectional bus. This command is in conflict with all read or write chips configuration commands.
Arguments:

char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.
char ireg;	JTAG instruction code.
char index;	Index of the chip in the JTAG chain. (TDI-->[chip 0]-->[chip1]---….--->TDO).
int nbit;	Number of bits of the register identified by ireg.
char bitstream[size];	Data to be shift in the JTAG register. For TRB and SB the byte order is identical to the parallel interface address (JTAG register inversion on some chips are software corrected), for BSR the bit7 of the first element is the most significant bit of the shift register and so away (see appendix B), instead for ROB the bit0 of the first element is the less significant bit of the shift register and so away. Size must be (nbit-1)/8+1. TDI-->[msb….JTAG-register…..lsb]--->TDO.
char bitmask[size];	Any bit to 1 indicate that the corresponding data bit will be shifted in the JTAG shift register, and corresponding data bit shift out is ignored. This is a equivalent write command on a normal register. Any bit to 0 indicate that the corresponding data bit shifted out from JTAG shift register will be captured and shifted in. This is a equivalent read command on a normal register. Size must be (nbit-1)/8+1.

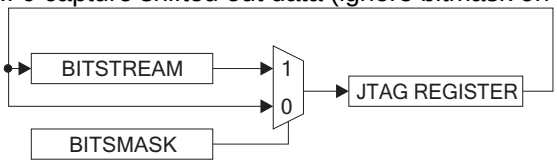


Return data:

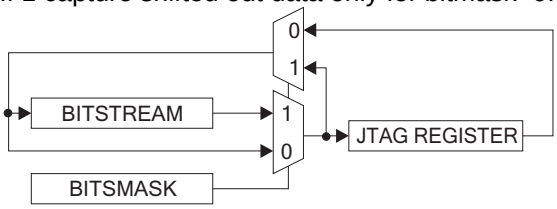
char 0xFC;	
char 0x55;	
char result;	return 1 if successful, 0 indicate invalid argument.

JTAG Execute, code 0x56:
Execute the JTAG sequence defined by JTAG_PresetData command.
Warning: This command are reserved for expert users. An improper user of the boundary scan can produce short circuit on bidirectional bus. This command is in conflict with all read or write chips configuration commands.
Arguments:

char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.
char reset;	If 1 place TAP in reset state at the and of sequence, if 0 place TAP in idle state.
char update;	If 0 capture shifted out data (ignore bitmask on return data).



If 1 capture shifted out data only for bitmask=0.



char test;	reserved must be 0.

Return data:

char 0xFC;	
char 0x56;	
char result;	return 1 if successful, 0 indicate invalid argument.

JTAG Shift IR, code 0x80:
Shift the Instruction code on the chips. This command in conjunction with JTAG_Shift_DR command is useful for EXTEST sequence on multiple boards. For example first Shift IR (code EXTEST) on all board, so the chips place preloaded data on outputs pin and after execute JTAG_Shift_DR for all board to capture input pins. The two command JTAG_Shift_IR and JTAG_Shift_DR are equivalente to the single command JTAG_Execute.
Warning: This command are reserved for expert users. An improper user of the boundary scan can produce short circuit on bidirectional bus. This command is in conflict with all read or write chips configuration commands.
Arguments:

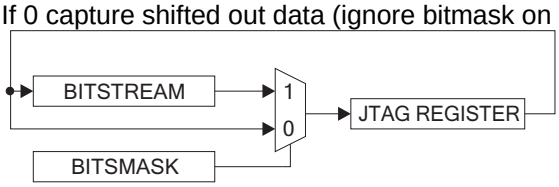
char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.

Return data:

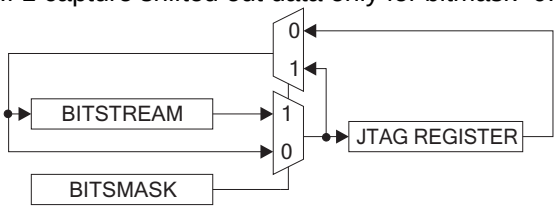
char 0xFC;	
char 0x80;	
char result;	return 1 if successful, 0 indicate invalid argument.

JTAG Shift DR, code 0x81:
Shift the data register chips. This command in conjunction with JTAG_Shift_IR command is useful for EXTEST sequence on multiple boards. For example first Shift IR (code EXTEST) on all board, so the chips place preloaded data on outputs pin and after execute JTAG_Shift_DR for all board to capture input pins. The two command JTAG_Shift_IR and JTAG_Shift_DR are equivalente to the single command JTAG_Execute.
Warning: This command are reserved for expert users. An improper user of the boundary scan can produce short circuit on bidirectional bus. This command is in conflict with all read or write chips configuration commands.
Arguments:

char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.
char reset;	If 1 place TAP in reset state at the and of sequence, if 0 place TAP in idle state.
char update;	If 0 capture shifted out data (ignore bitmask on return data).



If 1 capture shifted out data only for bitmask=0.



Return data:

char 0xFC;	
char 0x81;	
char result;	return 1 if successful, 0 indicate invalid argument.

JTAG Read Data, code 0x57:
Read return data after JTAG_Execute commad.
For the SB execute SB_SetupJtagChain command before of the JTAG_PresetData, because the JTAG chain change if the chips on the board are powered on or off.
Warning: This command are reserved for expert users. An improper user of the boundary scan can produce short circuit on bidirectional bus. This command is in conflict with all read or write chips configuration commands.
Arguments:

char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.
char ireg;	JTAG instruction code.
char index;	Index of the chip in the JTAG chain (TDI-->[chip 0]-->[chip1]---….--->TDO).

Return data:

char 0x58;	
char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB.
char nbrd;	Board number, ignored if brdid=0.
char ireg;	JTAG instruction code identifier the register.
char index;	Index of the chip in the JTAG chain (TDI-->[chip 0]-->[chip1]---….--->TDO).
char bitstream[size];	Return data.. Size is (nbit-1)/8+1, nbit is the number of bits of the shift register.

SB Setup JTAG Chain, code 0x4F:
Setup JTAG chain according to the SB chips power state.

Return data:
char 0xFC;
char 0x4E;

JTAG Board Test, code 0x84:
Verify integrity of the JTAG daisy-chain.
Arguments:

char brdid;	Board identifier 0=SB, 1=TRB, 2=ROB, 3=TTCrx.
char nbrd;	Board number, ignored if brdid=0 or 3.

Return data:

char 0x85;	
int result;	The result of the test. Bit15 indicate that the chain is interrupted, bit(0..7) indicate the number of chips in the chain, bit(8..14) indicate the size of the instruction register, 4bit for TRB chips and 5bit for ROBs chip.

JTAG Test, code 0x5D:
Verify integrity of the JTAG daisy-chain.
Arguments:

char jtag_addr;	Jtag address from 0 to 15, or -1 for test all address.

Return data:

char 0x5E;	
int result[16];	The result of the test. Bit15 indicate that the chain is interrupted, bit(0..7) indicate the number of chips in the chain, bit(8..14) indicate the size of the instruction register, 4bit for TRB chips and 5bit for ROBs chip.

TRB PI Test, code 0x5F:
Arguments:

char brd;	Board number from 0 to 5;
char ignorectrl;	if 1 ignore tests on control line (PROG and WR nets)

Return data:

char 0x60;	
char tss;	Any bit to 1 indicate a corresponding bit error on parallel bus.
char traco[4];	Any bit to 1 indicate a corresponding bit error on parallel bus.
char bti[32];	Any bit to 1 indicate a corresponding bit error on parallel bus.

Return data on error:

char 0xFC;	
char 0x5F;	
char error;	return -1.Indicate an argument error.

Test DACs, code 0x68:

Return data:

char 0xFC;	
char 0x68;	
int result;	0x37F all DACs OK.

Read ADC, code 0x69:
Read 32 ADC channels .

Return data:
char 0x6A;
int adc[32]; ADC conversion values. Element list is: [0]Front-Ends MAD temperature connector 1, [1] Front-ends average temperature connector 1, [2]Front-Ends Vcc connector 1, [3]Front-ends Vdd connector 1, [4]Front-Ends MAD temperature connector 2, [5]Front-ends average temperature connector 2, [6]Front-Ends Vcc connector 2, [7]Front-ends Vdd connector 2, [8]Front-Ends MAD temperature connector 3, [9]Front-ends average temperature connector 3, [10]Front-Ends Vcc connector 3, [11]Front-ends Vdd connector 3, [12]Bias 1, [13]Threshold 1, [14]Bias 2, [15]Threshold 2, [16]Bias 3, [17]Threshold 3, [18]Width, [19]TRBs Fault, [20]Vdd in, [21]Vcc in, [22]Vdd, [23]Vcc, [24]Vdd splitter board, [25]Vcc splitter board, [26]TP1L, [27]TP2L, [28]TP1H, [29]TP1, [30]TP2H, [31]TP2.

Test Fine Delay, code 0x82:

Return data:
char 0xFC;
char 0x82;
char CpuCkdelay:1; This bit indicate the test result of the CPU CK fine delay.
char TpFineDelay1:1; This bit indicate the test result of the Test-Pulse1 fine delay.
char TpFineDelay2:1; This bit indicate the test result of the Test-Pulse2 fine delay.

CCBRdy Pulse, code 0x91:
This command is used only for mini crate tests.
Generate a pulse on line CCB Ready.

Return data:
char 0xFC;
char 0x91;

PIRW, code 0x92:
This command is used only for mini crate tests.
Set state of the parallel interface read/write net.
Arguments:
char state;
Return data:
char 0xFC;
char 0x92;

PIPROG, code 0xAF:
This command is used only for mini crate tests.
Set state of the parallel interface Prog net.
Arguments:
char state;
Return data:
char 0xFC;
char 0xAF;

Compare Chip register, code 0x9C:
Compare BTI register with memory and return XOR result.
Arguments:

char brd;	Board number, form 0 to 7.
char idchip;	Identifier chip 0=TSS 1=TRACO 2=BTI
char chip;	chip number.
char mode;	1=refresh mode (write ram content to chip), 0=read mode.

Return data:

char 0x9D;	
char conf[n];	the n byte of the chip CONF register XOR result. 8bit format for BTI.
char testing[n];	the n byte of the chip TESTIN register XOR result. 8bit format for BTI.

TRB PI Test, code 0xA0:
This command is used only to test TRB parallel interface bus.
Arguments:

char nbrd;	board number.
char idchip;	0=TSS, 1=TRACO 2=BTI.
char nchip;	chip number.
char reg;	register address.
char write;	1=WITE, 0=READ;
char data;	write data;

Return data:

0xFC;	
0xA0;	
char data;	read data or written data.

Test IR, code 0xA2:
This command is used only for TRB diagnostic.
Arguments:

char addr;	jtag address.
int nck;	number of CK.

Return data:

0xFC;	
0xA2;	

Patterns, code 0xB8:
This command fills all TRACO LUT with patterns defined in the
arguments, to generate known mini-crate output.
Arguments:

int pos;	value written in position LUT on all TRACOs.
int ang;	value written in angle LUT on all TRACOs.

Return data:

char 0xFC;	
char 0xB8;	
long error;	return 0 on success, return -2 if DAQ run is in progress and the LUTs are not written, return error > 0 on write error, any bit to 1 from bit 0 to bit 23 identified the TRACO write fault (6 board * 4 TRACO=24).

Write TDC Setup, code 0x21:
Write TDC setup register. The control sequence is automatically execute by CCB.

Arguments:

char brd;	Board number from 0 to 6
char chip;	TDC number from 0 to 3
char setup[81];	Setup register

Return data:

char 0x23;	
char setup[81];	Setup data read from JTAG TDO.

Return data on error:

char 0xFC;	
char 0x21;	
char narg;	return the negative number of the wrong argument.

Special codes:

The special code can be added to the head of all commands. A command can have more than a special code added to the head, but of different value.

Host, code 0xEC:
This code followed by one byte argument can be used from a server program to mark the commands send by client. The return data from the CCB have to the head the same code and argument. The argument can assume any value.

Bridge, code 0xDE:
This code followed by one int (two byte) argument work only on primary port, it is used to communicate with a CCB that have the primary port broken and a branch of the RS485 chain bus interrupted. The CCB that receive a command with this code to the head, send the received command on the own secondary port (interrupted branch of the bus RS485). The two byte argument is reserved and must be zero. Before to send command to the bus RS485 the final destination CCB must be activated by a command BridgeAddr described below.

Bridge Address, code 0xDF:
This is a standard command and must be used before to send command with the special head Bridge code (0xDE). This command work only if it is sent to the primary port. When CCB receive this command it send on the own secondary port the address frame to activate one CCB on bus RS485.
Arguments:
int CcbID; The identifier address of the CCB to activate.
Return data:
char 0xFC;
char 0xDF;

Optical Request, code 0xAB: (Not implemented on BOOT program)
Code transmitted by CCB to request the use of the optical RS232 to setup the link discriminator threshold. The same code must be retransmitted to the CCB to start the procedure. During the procedure the CCB transmit some Ping code (see description below)

Optical Release, code 0xAC: (Not implemented on BOOT program)
Code transmitted by CCB to signal the end of the procedure to set up optical link discriminator threshold.

Ping, code 0xE8:
Code transmitted by CCB to setup threshold of the optical link discriminator. This data must be retransmitted to the CCB.
Data transmitted:
char 0xE8;
char data[size]; size and data assume generic value.

Split, code 0xB6: (Not implemented on BOOT program)
This code followed by one byte argument can be used to split large return data from mini-crate.
When mini-crate receive a command with this code at the head, following by the argument that define the size to split data (size must be < 128), it if the return data exceed the specified size data, transmit only amount of the size data specified. The data are marked at the head by code 0xB6 following by one byte that define the data offset and one byte that define the data size remaining to transmit. To transmit remaining data mini-crate must receive command code 0xB7 describe below.

Send Remaining, code 0xB7: (Not implemented on BOOT program)
This is a standard command used to read remaining return split data.
Arguments:
char offset;
Return data:
char 0xB6;
char offset; data offset.
char remaining; reaming data to transmit.
char data[size]; size is the value specified with the split code.

Hardware problems:

-When power on mini crate if 5V is stable before 3.3V, some time, CCB enter in a undefined state. To prevent this situation following this power sequence : Power on supply for 3.3V and after power on supply for 5V. To power off mini crate , power off supply for 5V and after power off supply for 3.3V.
-When power on TSM Sorter his JTAG bus an parallel interface don't work without a soft reset.
-The power on of the SB_CK block (isolation switch and distribution CK chips), on the SB board, must be done for last, because the power on of the SB FPGA generate a glitch on the reset line of the microcontroller.
-When power off only one TRB_theta some time parasitic voltage, deriving from interconnection signals, is present on board. This voltage will be eliminate by a switch off and a switch on of the VCC_TRB block (supply voltage for all TRB isolation switch chips).
- When power off one TRB_phi with TSM Sorter powered parasitic voltage deriving from interconnection signals is present on the TRB_phi .
- TRGOUT signal on trigger bus data is not always sampled correctly by serializer (lose signal about 0.1%).

Appendix A

IEEE 32bit Format (Precision: 6.5 decimal digits)

sign bit	8bit exp	23bit mantissa

value= $-1^{s} * 2^{(e-127)} * 1.m$

IEEE 64bit Format (Precision: 15 decimal digits)

sign bit	11bit exp	52bit mantissa

value= $-1^{s} * 2^{(e-1023)} * 1.m$

DSP Format (Precision: 4.5 decimal digits)

16bit mantissa	16bit exp

value= $m * 2^{e}$ (no hidden bit)

Conversion from DSP float to IEEE32 float:

```
void DSPtoIEEE32(short DSPmantissa, short DSPexp, float *f)
{
    DSPexp -= 15;
    *f = DSPmantissa * (float)pow( 2.0, DSPexp );
}
```

Conversion from IEEE32 float to DSP float:

```
void IEEE32toDSP(float f, short *DSPmantissa, short *DSPexp)
{
long *pl, lm;
BOOL sign=FALSE;

    if( f==0.0 )
    {
        *DSPexp = 0;
        *DSPmantissa = 0;
    }
    else
    {
        pl = (long *)&f;
        if((*pl & 0x80000000)!=0)
            sign=TRUE;
        lm =( 0x800000 | (*pl & 0x7FFFFF)); // [1][23bit mantissa]
        lm >>= 9; //reduce to 15bits
        lm &= 0x7FFF;
        *DSPexp = ((*pl>>23)&0xFF)-126;
        *DSPmantissa = (short)lm;
        if(sign)
        *DSPmantissa = - *DSPmantissa;   // convert negative value in 2.s
                                         // complement
    }
}
```

Function to calculate the CRC polynomial X16+X12+X5+1:

```
void Crc(int *crc,int data) // Calculate the CRC for the first 8bits of the data
{
unsigned int test;
int dt,i;


      dt=data<<8;
      for(i=0;i<2;++i) //Optimized for execution.
      {
            test=(dt ^ *crc) & 0xF000;
            *crc<<=4;
            *crc^=test;
            test>>=7;
            *crc^=test;
            test>>=5;
            *crc|=test;
            dt<<=4;
      }
}
```

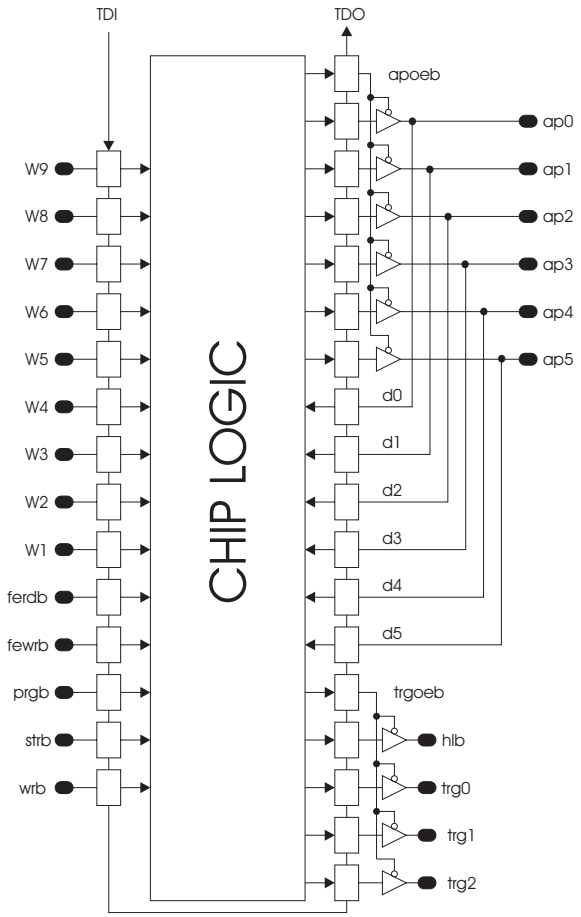
Appendix B

TRACO Boundary Scan Registers chain

Bit Stream order

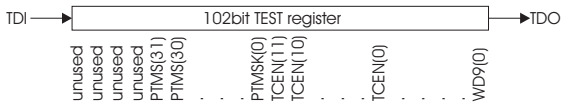
	BYTE	BIT	BSR	Name	BYTE	BIT	BSR	Name	BYTE	BIT	BSR	Name	
TDI->	0	7	188	TRG1	8	7	124	TRG9	16	7	60	SELECT	
		6	187	HLB1		6	123	HLB9		6	59	POSTSELCT	
		5	186	AP1(5)		5	122	AP9(5)		5	58	THETA	
		4	185	AP1(4)		4	121	AP9(4)		4	57	SQADV	
		3	184	AP1(3)		3	120	AP9(3)		3	56	SEQTST	
		2	183	AP1(2)		2	119	AP9(2)		2	55	SQRST	
		1	182	AP1(1)		1	118	AP9(1)		1	54	PROGb	
		0	181	AP1(0)		0	117	AP9(0)		0	53	ILHL	
	1	7	180	TRG2	9	7	116	TRG10	17	7	52	IRHL	
		6	179	HLB2		6	115	HLB10		6	51	OLHL	
		5	178	AP2(5)		5	114	AP10(5)		5	50	ORHL	
		4	177	AP2(4)		4	113	AP10(4)		4	49	FS (DT0)	
		3	176	AP2(3)		3	112	AP10(3)		3	48	OVLP (DT1)	
		2	175	AP2(2)		2	111	AP10(2)		2	47	MSK0 (DT2)	
		1	174	AP2(1)		1	110	AP10(1)		1	46	MSK1 (DT3)	
		0	173	AP2(0)		0	109	AP10(0)		0	45	MSK2 (DT4)	
	2	7	172	TRG3	10	7	108	TRG11	18	7	44	BEND0 (DT5)	
		6	171	HLB3		6	107	HLB11		6	43	BEND1 (DT6)	
		5	170	AP3(5)		5	106	AP11(5)		5	42	BEND2 (DT7)	
		4	169	AP3(4)		4	105	AP11(4)		4	41	BEND3 (DT8)	
		3	168	AP3(3)		3	104	AP11(3)		3	40	BEND4 (DT9)	
		2	167	AP3(2)		2	103	AP11(2)		2	39	BEND5 (DT10)	
		1	166	AP3(1)		1	102	AP11(1)		1	38	BEND6 (DT11)	
		0	165	AP3(0)		0	101	AP11(0)		0	37	BEND7 (DT12)	
	3	7	164	TRG4	11	7	100	TRG12	19	7	36	BEND8 (DT13)	
		6	163	HLB4		6	99	HLB12		6	35	BEND9 (DT14)	
		5	162	AP4(5)		5	98	AP12(5)		5	34	RAD0 (DT15)	
		4	161	AP4(4)		4	97	AP12(4)		4	33	RAD1 (DT16)	
		3	160	AP4(3)		3	96	AP12(3)		3	32	RAD2 (DT17)	
		2	159	AP4(2)		2	95	AP12(2)		2	31	RAD3 (DT18)	
		1	158	AP4(1)		1	94	AP12(1)		1	30	RAD4 (DT19)	
		0	157	AP4(0)		0	93	AP12(0)		0	29	RAD5 (DT20)	
	4	7	156	TRG5	12	7	92	TRG13	20	7	28	RAD6 (DT21)	
		6	155	HLB5		6	91	HLB13		6	27	RAD7 (DT22)	
		5	154	AP5(5)		5	90	AP13(5)		5	26	RAD8 (DT23)	
		4	153	AP5(4)		4	89	AP13(4)		4	25	RAD9 (DT24)	
		3	152	AP5(3)		3	88	AP13(3)		3	24	RAD10 (DT25)	
		2	151	AP5(2)		2	87	AP13(2)		2	23	RAD11 (DT26)	
		1	150	AP5(1)		1	86	AP13(1)		1	22	PI_5	
		0	149	AP5(0)		0	85	AP13(0)		0	21	PI_4	
	5	7	148	TRG6	13	7	84	TRG14	21	7	20	PI_3	
		6	147	HLB6		6	83	HLB14		6	19	PI_2	
		5	146	AP6(5)		5	82	AP14(5)		5	18	PI_1	
		4	145	AP6(4)		4	81	AP14(4)		4	17	PI_0	
		3	144	AP6(3)		3	80	AP14(3)		3	16	PI_6	
		2	143	AP6(2)		2	79	AP14(2)		2	15	PI_7	
		1	142	AP6(1)		1	78	AP14(1)		1	14	PRV_5	
		0	141	AP6(0)		0	77	AP14(0)		0	13	PRV_4	
	6	7	140	TRG7	14	7	76	TRG15	22	7	12	PRV_3	
		6	139	HLB7		6	75	HLB15		6	11	PRV_2	
		5	138	AP7(5)		5	74	AP15(5)		5	10	PRV_1	
		4	137	AP7(4)		4	73	AP15(4)		4	9	PRV_0	
		3	136	AP7(3)		3	72	AP15(3)		3	8	PRV_IO (PRV_6)	
		2	135	AP7(2)		2	71	AP15(2)		2	7	PRV_HL (PRV_7)	
		1	134	AP7(1)		1	70	AP15(1)		1	6	PRV_OEb	
		0	133	AP7(0)		0	69	AP15(0)		0	5	/FS (/PRV_9)	
	7	7	132	TRG8	15	7	68	TRG16	23	7	4	FS_OEb	
		6	131	HLB8		6	67	HLB16		6	3	/COR (/PRV_8)	
		5	130	AP8(5)		5	66	AP16(5)		5	2	COR_OEb	
		4	129	AP8(4)		4	65	AP16(4)		4	1	BTI_STRB	
		3	128	AP8(3)		3	64	AP16(3)		3	0	BTI_STRB_OEb	->TDO
		2	127	AP8(2)		2	63	AP16(2)		2			
		1	126	AP8(1)		1	62	AP16(1)		1			
		0	125	AP8(0)		0	61	AP16(0)		0			

BTI Boundary-Scan

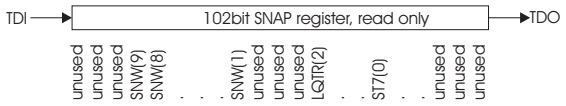


BTI Configuration registers

See BTI reference manual table 4.1.1



See BTI reference manual table 4.2.1



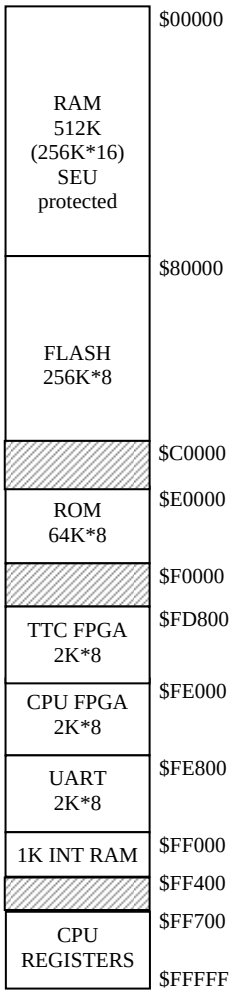
See BTI reference manual table 4.3.1

Appendix C

The CPU address space

In the right figure is represented the CPU address space when running the mini-crate firmware that run in the 512K RAM (SEU protected).
The BOOT program, that run in ROM, generate an equivalent virtual map as show in the figure except for the ROM that is inaccessible from the read commands, because to work CPU must map ROM in the first 64K of the address space and dynamically change the position of the 512K of the RAM when you access to that address space.
Addressing of the not mapped space generate a bus error and consequent reset of the CPU.
The BOOT program to work use only the ROM and 1K of the internal RAM .
The UART and FPGA use only the fist 16 bytes of the 2K space assigned, the next bytes are always the same first 16 bytes.
The flash address space from $80000 to $AFFFF in actual firmware version is reserved for storage mini-crate software.
From the address $B0000 is stored the mini-crate configuration with the format :

```
long size;
char data[size-2];
int crc;
```

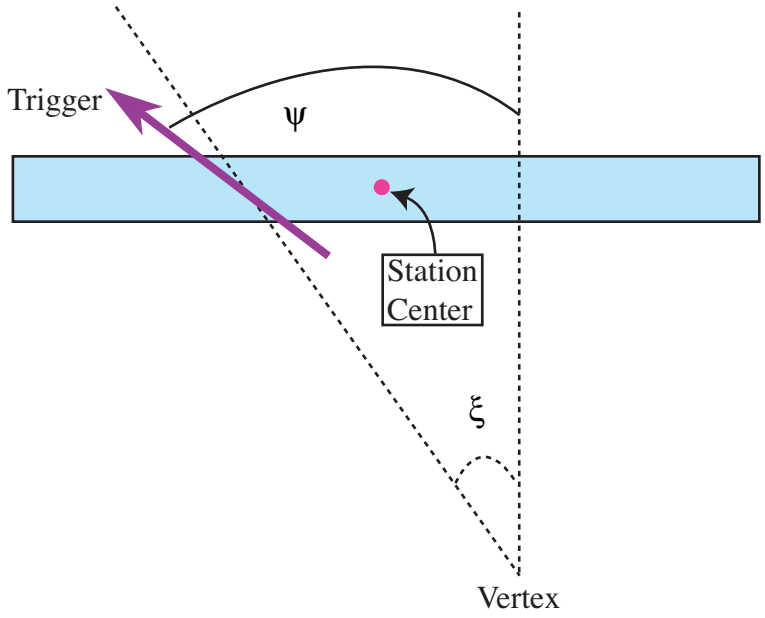


Appendix D

TRACO Look-Up Tables Computation

(S. Vanini and P. Zotto – Padova University)

Angles are define in the following figure



K Table (angle with normal ψ)

The table has 1024 locations.
The index for the calculation is $-512 \le i \le 511$ (corresponding to locations $0 \le j \le 1023$).

For every location the angle is given by

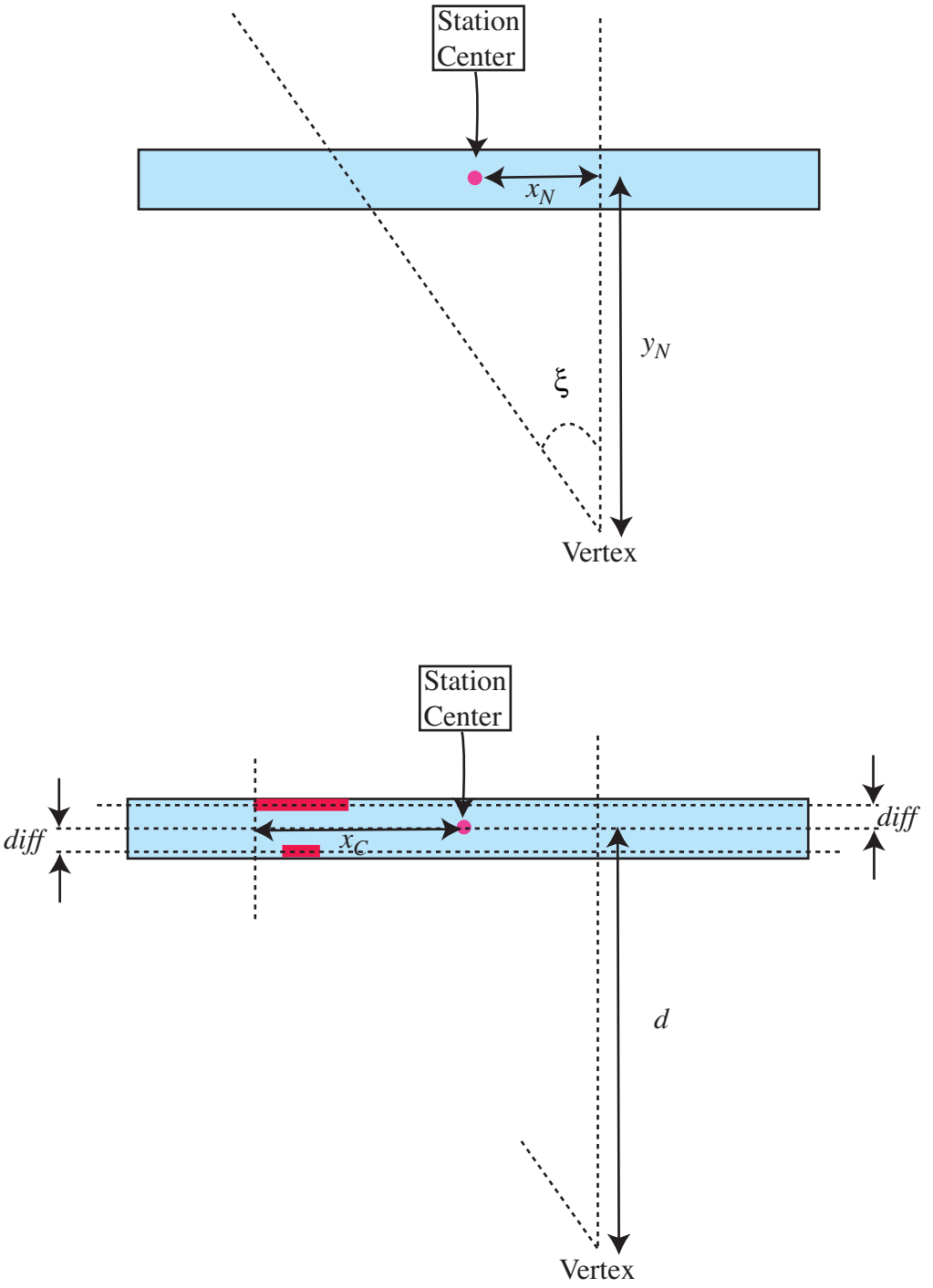
$$\psi = \pm \tan^{-1}\left(\frac{i \times pitch}{D \times ST}\right) \times 512 \qquad \begin{cases} + & \text{for YBj+wheel}(j = 0,1,2) \\ - & \text{for YBj−wheel}(j = 0,1,2) \end{cases}$$

where

$$\begin{cases} pitch = 4.2cm & \text{distance between consecutive wires} \\ D = 23.5cm & \text{distance between SL } \varphi \text{ centres} \\ ST & \text{TRACO BTIC parameter} \end{cases}$$

X Table (radial angle ξ)

The geometrical variables are defined in the following figures



The table has 1536 location assigned as follows:

0 – 511	outer triggers
512 – 1023	inner triggers
1024 – 1535	correlated triggers

For each group the index for the calculation is $0 \le i \le 511$ (with obvious relationship with corresponding locations). Hence

$$\xi = \tan^{-1}\left(\frac{x_C - x_N \mp \frac{pitch}{ST}\times i}{y_N}\right)\times 4096 \qquad \begin{cases} - & \text{for YBj+ wheel } (\mathrm{j}=0,1,2) \\ + & \text{for YBj– wheel } (\mathrm{j}=0,1,2) \end{cases}$$

where

$$\begin{cases} pitch = 4.2cm & \text{distance between consecutive wires} \\ ST & \text{TRACO BTIC parameter} \\ x_N & \text{local x position of normal to station through vertex} \\ y_N & \text{distance from vertex of SL or station centre along normal} \\ x_C & \text{local x position of origin of correlator frame} \end{cases}$$

Called *d* the distance from the vertex to the station center, computed along the normal to the station through the vertex itself, for each trigger kind we have

$$y_N = d + diff \quad \text{with} \quad \begin{cases} diff = 11.75cm & \text{outer triggers} \\ diff = -11.75cm & \text{inner triggers} \\ diff = 0 & \text{correlated triggers} \end{cases}$$

therefore defining $x_{CN} = x_C - x_N$ the final formula is

$$\xi = \tan^{-1}\left(\frac{x_{CN} \mp \frac{pitch}{ST}\times i}{d + diff}\right)\times 4096 \qquad \begin{cases} - & \text{for YBj+ wheel } (\mathrm{j}=0,1,2) \\ + & \text{for YBj– wheel } (\mathrm{j}=0,1,2) \end{cases}$$

Noting that only the position of the first correlator is needed since the position of the other ones can be computed as

$$x_{CN}(i) = x_{CN}(1) \mp 4\times pitch\times(i-1) \qquad \begin{cases} - & \text{for YBj+ wheel} (\mathrm{j}=0,1,2) \\ + & \text{for YBj– wheel} (\mathrm{j}=0,1,2) \end{cases}$$

the needed parameters finally are

$$\begin{cases} pitch = 4.2cm & \text{distance between consecutive wires} \\ ST & \text{TRACO BTIC parameter} \\ x_{CN} & \text{distance first correlator– normal } \textit{(N.B.: signed float)} \\ d & \text{distance vertex– normal} \\ diff & \text{distance of SL from station center } \textit{(constant per trigger group)} \end{cases}$$

Appendix E

The figure 1 and 2 show all the power blocks on SB/CCB boards .
Below there is the correspondence between power blocks name and blocks show in the figure.
RAM block is VCC2
FLASH block is VCC1
ANALOG block is VCCan
CK block is VDDck
RPC block is VCCrpc
LED block is VCCled
TSMS block is VCCso and VDDso
TSMDU block is VCCdu and VDDdu
TSMDD block is VCCdd and VDDdd
SBTH block is VDDth
SBCK block is VCCsw
BUFFTRB block is VDDbf
VCCTRB block in not show on the figure but it power the isolation switch chips on TRB

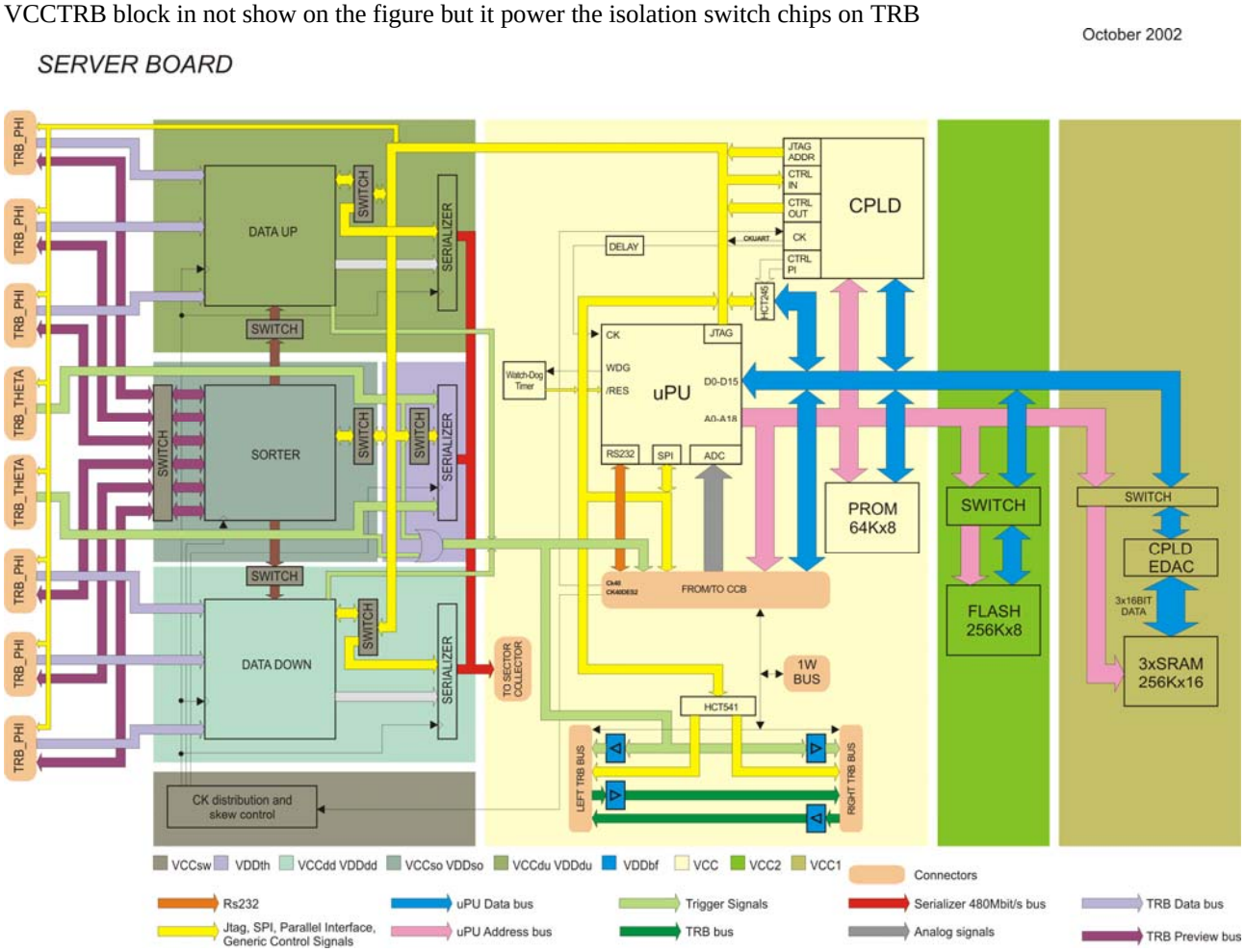


Figure1

CONTROL BOARD

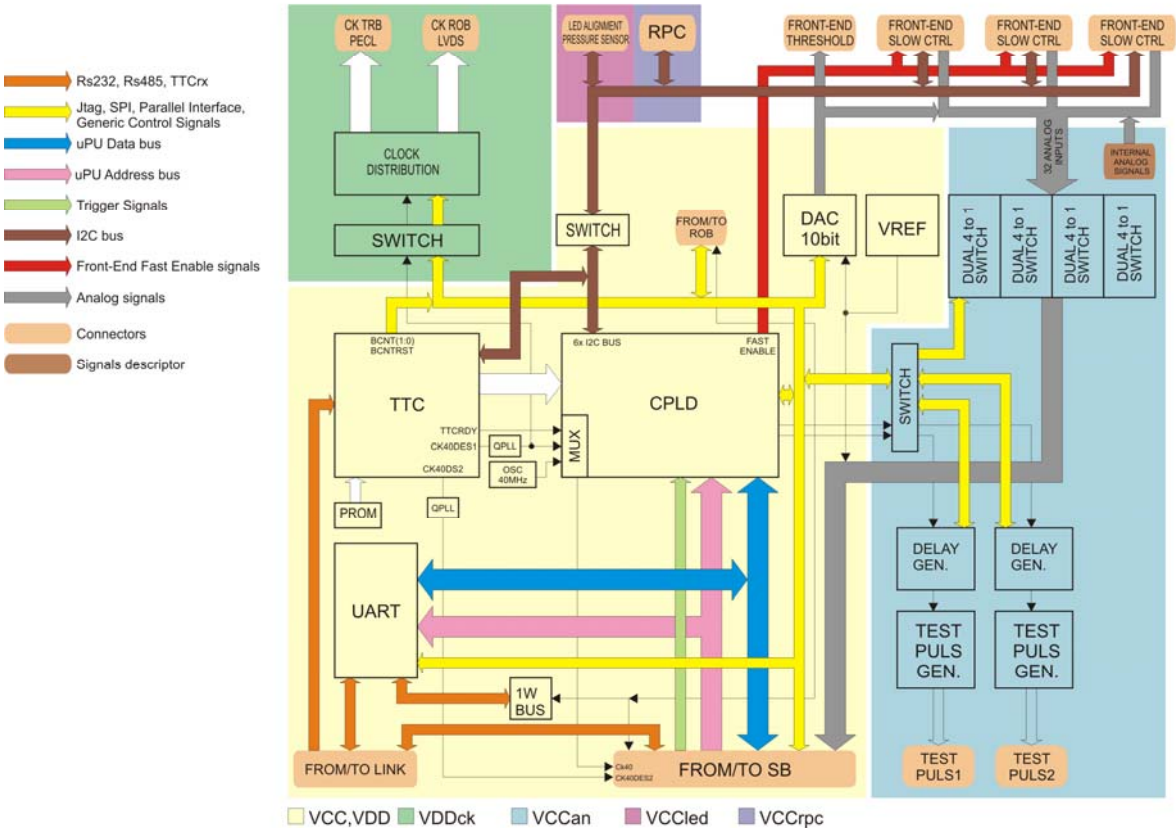


Figure 2