The SALTRO Detector Control System (DCS) raw data format

Version	Comment
20130319	First draft proposal
20130411	specify data words
20130422	number of types and identifiers
20130502	all data are 2 bytes, removed type=software, 0=all removed, split ADC
20140305	Test socket board added. Updates a few things.
20140522	Handling of scanning and automatic power off

The data format is identical for downloading settings and uploading readings.
For reloading already downloaded and locally stored settings will eventually short commands exist.

The format described here is between: server running on PC – DCS master – DCS CPU on 5to1 card.
The format used between: user interface - server running on PC is described in a separate document TO BE WRITTEN!
The description of the DCS system is in a separate document. TO BE UPDATED!
(old versions:
http://www.hep.lu.se/eudet/saltro/i2c-control-system.pdf
http://www.hep.lu.se/eudet/saltro/lvpanel-2.pdf
These documents are the proposals and are not describing the system as of today!
)

The different parts has its own unique identifier defined by
PC server id (6)
master id (7)
5to1 box id (0-3)
cpu address (1-5)
device types as given below
device identifiers which identifies the actual physical device as given below

Hardware device types

1	temperature sensor
2	ADC GET – *OBSOLETE!*
3	ADC IN MCM – Adc that measure the voltages before the regulators
4	ADC OUT MCM – Adc that measure the voltages after the regulators
5	I/O register – power off/on MCMs
6	DAC – settings of e.g. SALTRO reference voltages
7	CPLD – the progammable device on the MCM

Software device types

8	Command
9	ACK CHANGES – sent by PC server when SCANEND received
10	FAIL – reason for automatic power off

Device identifiers are given below for each device type.

Device identifiers are given in hex (nm)

Temperature sensor identifiers

n1	MCM1
n2	MCM2
n3	MCM3
n4	MCM4
n5	MCM5
n6	LV panel

n=0: temperature value (read only) value is raw value as read from tmp101
n=1: low value set in tmp101, value is raw value as given by tmp101
n=2: high value set in tmp101, value is raw value as given by tmp101
n=3: shutdown limit in degrees
n=4: number of cycles above shutdown limit before shutdown

The raw value from tmp101 is the upper 10 bits in a 16 bit word. The uppermost 8 bits are degrees, the lower 2 bit values are for 0 0.25,0.5,075 degrees.

ADC identifiers

n0	IN n where n=1 to 8 (voltage regulator IN)
nm	IN nm where n=1 to 8 (voltage regulator IN) after resistor, m = 1 to 5 (MCMm)
nm	OUT nm where n=1 to 8 (voltage regulator OUT) after regulator, m = 1 to 5 (MCMm)

There is no write for these device identifiers, they can only be read. Values are the raw values as read from the ADC.

n6	IN Shutdown limits in mA for n=1 to 8 (voltage regulator IN). This value is converted by the PC server to a voltage difference before sent to slave(s), the voltage depends on the resistor value.
96	IN Number of cycles above shutdown limit before shutdown, common to all regulators.

I/O register identifier

0m	Voltage regulator where m = 1 to 5 (I.e. voltage regulator for a given MCMm, one bit per voltage). Bit = 0: power off, bit = 1: power on.
07	Testsocketboard (=7) PCA settings (bit pattern) 0 = polarity 1 = shaper3 2 = shaper2 3 = shaper1 4 = preampmode 5 = shutdown 6 = gain2 7 = gain1
17	Testsockerboard (=7) Chip address (bit pattern)

0 = chip0
1 = chip1
2 = chip2
3 = fec0
4 = fec1
5 = fec2
6 = fec3
7 = fec4

DAC identifier

cm MCM number m = 1 to 5, c = channel 1 to 8 on the DAC,
1 = decay
2= refP
3 = inCM
4 = refN
5 = outCM

c7 Testsocketboard (=7) c = channel 1 to 8 on the DAC
1 = decay
2= refP
3 = inCM
4 = refN
5 = outCM
6 = bias

CPLD identifier – NOT YET IMPLEMENTED

Command identifier

Commands can only be sent from a master/server.

11 Scan temperature interval in seconds (0 = no scanning)
12 Scan adc interval in seconds (0 = no scanning)
13 Status of current value (between 0-interval) in temperature scan interval (readonly)
14 Status of current value (between 0-interval) in adc scan interval (readonly)
15 Which CPUs (1-5)/LV board to scan for temperature (bit pattern 0LCCCCC0).
16 Which CPUs (1-5) to scan for ADC (bit pattern 00CCCCC0).

1 power on according to already downloaded settings – NOT IMPLEMENTED
2 power off all regulators – NOT IMPLEMENTED
3 load all DACs with already downloaded settings – NOT IMPLEMENTED
4 configure temperature sensors with already downloaded settings – NOT IMPLEMENTED
5 do 1,3, and 4 – NOT IMPLEMENTED
6 send changes since last reading – NOT IMPLEMENTED
7 store current configuration as default – NOT IMPLEMENTED
8 recall default configuration – NOT IMPLEMENTED
9 ping

10 echo

ACK CHANGES identifiers

Sent by PC server as a acknowledge of received ADC values. Only needed after receiving values changed since last time sent. Used by 5to1 CPU to update the ADC values last sent.

11 DUMMY

FAIL identifiers

When sent by 5to1 CPU: indicates a FAILure. Currently only temperature and high currents, which caused an automatic power off.

01 Failed temperatures which caused an automatic power off – bit pattern:
1-5 MCM number
6 LV panel
02 Failed high currents which caused an automatic power off – bit pattern:
1-5 MCM number

When sent by PC server, the bits tell which FAIL bit to clear.

01 clear temperature fail status – bit pattern
1-5 MCM number
6 LV panel
02 clear high current fail status – bit pattern
1-5 MCM number

Data Format

Data blocks are transferred in binary

Word	content	bytes	comment
0	preamble	2	0xA5A5
1	length	2	length in bytes – excluding itself
2	destination	1	bit 0-2, cpuaddress 0: broadcast to all 5to1 CPUs for the boxid in bit 3-4 1-5: 5to1 CPU 6: monitor computer 7 : master CPU bit 3-4, boxid if 5to1 CPU bit 3-6, 4 LSB of MAC address if master CPU or monitor computer bit 7, not used set to 0
3	source	1	bit 0-2, cpuaddress 1-5: 5to1 CPU 6: monitor computer 7: master CPU

			bit 3-4, boxid if 5to1 CPU bit 3-6, 4 LSB of MAC address if master CPU or monitor computer bit 7, not used set to 0
4	packet number	1	incremented by source for each packet sent (individual per dest)
5	instruction	1	instruction: e.g. read,load,status.
6	time stamp	2	in seconds
7	device types	1	number of device types in packet

Then follows device types, identifiers, and data. There can be more than one device types, and for each device type more than one identifier. Types and identifiers are 1 byte, data 2 bytes.

(1 byte, 1 byte,1 byte, 2 bytes,1 byte, 2 bytes,...)
Device type1, number of identifiers, Device identifier1, data1, device identifier2, data2 ..
...
Device typeN, number of identifiers,Device identifier1, data1, device identifier2, data2 ..

Instructions:
0 = null instruction
1 = load
2 = read
3 = load/read
4 = status
5 = changes
6 = acknowledge
7 = scanend (sent by master when an adc scan has ended, expects ack_changes back)
8 = status of temperature and current scan, CPU will return values of the limit counters

NOTE: Hardware and Software/Command types may not be mixed.
The detailed data size and format for a device identifier is defined by the device type in the identifier word as given below.

In case of odd number of bytes is a 0-byte inserted to get an even number of bytes

Last-1	length	2	length in bytes – excluding itself (same as 1)
Last	checksum	2	xor of all 2-byte words

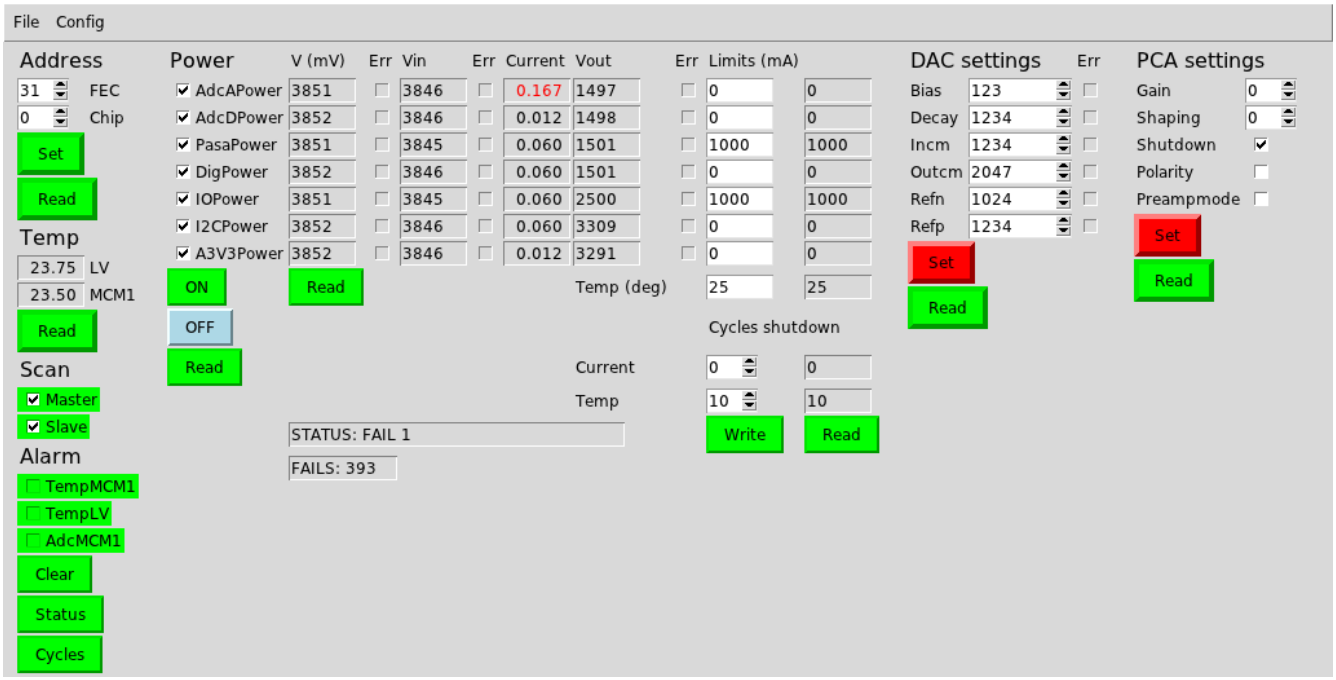
Number of devices and data sizes per device

Device	*bytes*	*LV-board*	*content*
I/O register	2	8	Voltage regulators bit pattern ON/OFF
ADC	2	8	12 bit ADC value INn
ADC	2	40	12 bit ADC value INnm
ADC	2	40	12 bit ADC value OUTnm
Temperature	2	6	temperature in steps of 0.25 degrees
DAC	2	40	12 bit DAC value (DACcm)

Devices per LV board		142	*2 bytes = 284 binary data bytes from each LV board
Per pad module		710	
3 pad modules		2130	

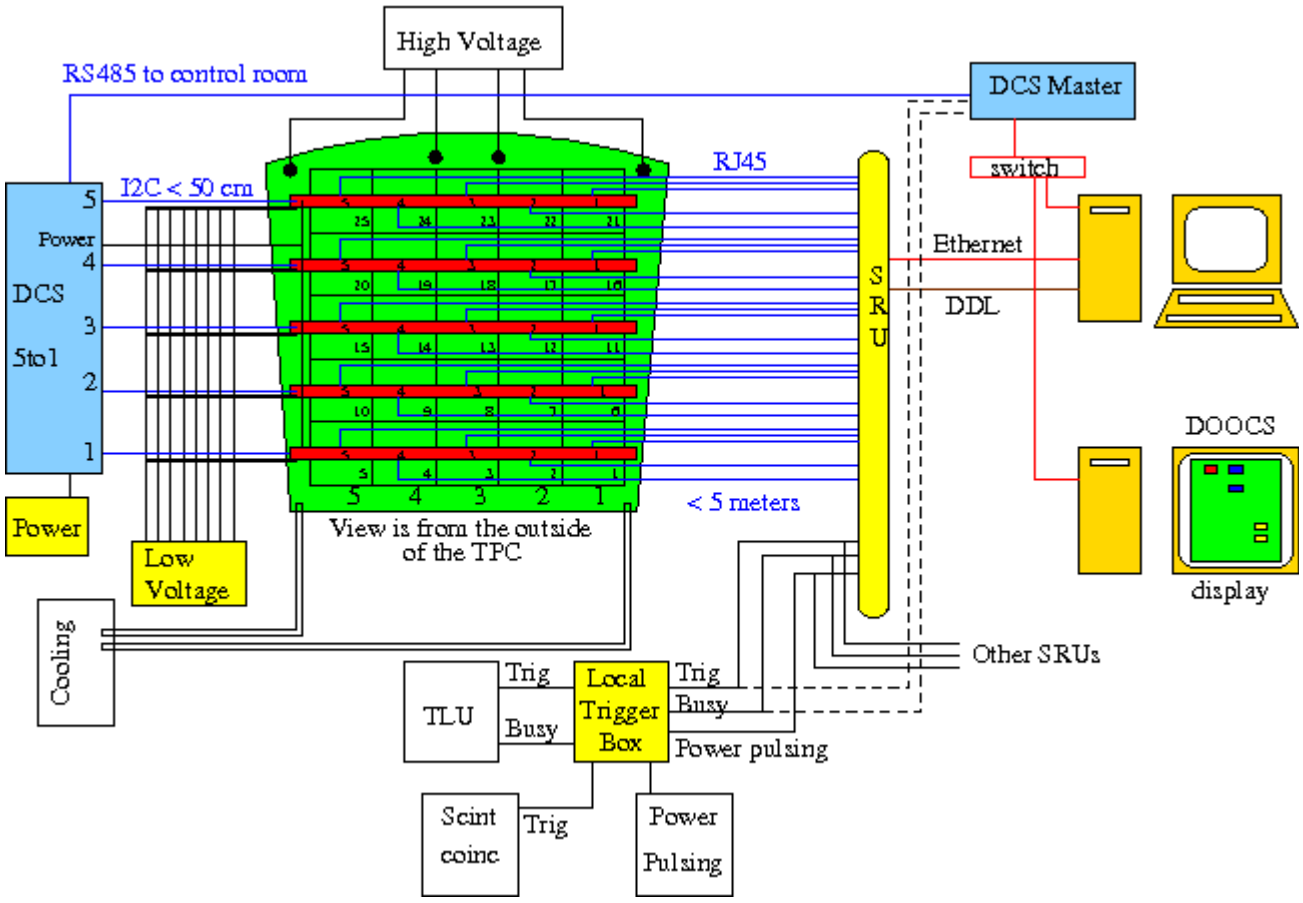
Server and monitor computer

The format between the monitor&control PC server and the master box is the same as described above. The communication is over ethernet. Therefore may the data packet length not exceed the ethernet packet size of 1500 bytes. The monitor and controlling server can be on different computers. The main user interface is proposed to be DOOCS. For a local standalone control will a simple user interface be available (written in python). A user interface has been written for the testsocketboard, which is used for testing carrier boards.



Numbering

PRELIMINARY
The numbering is as seen from the outside of the TPC. The numbering is increasing in the direction of a particle coming from a thought collision point in the center, and is anticlockwise in phi.



Chains selection on the Low Voltage (LV) board:

The connection between the LV board/MCMs to the 5to1 CPUs is I2C. There are eight different I2C chains:

Chain	Content	Available on	
0	none		
1	Regulators + tmp101	LV	protoLV
2	ADC1-8	LV	protoLV
3	ADC9-16	LV	-
4	MCM1	LV	protoLV
5	MCM2	LV	protoLV-Testscocketboard
6	MCM3	LV	-
7	MCM4	LV	-
8	MCM5	LV	-

The chain is selected via an I2C I/O register. This IO register is accessible by the master/server. The 5to1 CPU selects the chain according to the hardware device address.

Some timing:

ReadADC	26 ms	at 100kHz

SendADC-RS485	75ms	38400 baud
Send ethernet	20ms	prescalePrim 10 = 4:1 prescaleSec 000 = 8:1 20*10^6/32 =~ 1Mbit/sec