ZEUS-Note 96-061
18 July 1996

SLOW CONTROL DETECTOR
GENERAL DESCRIPTION

S.G.Basiladze', O.A.Derugin', A.V.Kalagov', V.M.Rybnikov', C.Youngman"
' - Moscow State University, Nuclear Physics Institute
Nuclear Electronics and Automation Laboratory
" - DESY, ZEUS, Hamburg

Contents

More detailed information you can find on WWW:

http://zaxp03.desy.de:8000/

Part I. GENERAL INFORMATION

The Slow Control Detector (SCD) task is to collect an information derived from slow control environment (and possibly from any source is reachable via network) and to put it onto the physics dataflow.

1. SCD design issues

Within the ZEUS DAQ system data taking components are required to insert their data into the dataflow. The dataflow transports data from these components, through the experiment event builder and third level trigger, to the data archiving machine.

The DAQ's slow control system gathers a slow control status information from many networked machines. The Slow Control Manager (SCM) is the end point of this information logging system. The SCD inserts data available at the SCM into the dataflow.

The SCD has been operating since the start of the 1994 data taking period.

2. Main SCD functions

The SCD position and its links to the main ZEUS subsystems are shown in Fig.1.

The sub-system consists of four main parts:

- a number of datataking servers running on machines within the hall;
- a number of the SCD processes on ZAXP03;
- a SCD-to-EVB communication program. Based on 2TP_VME module, one transputer is used by the SCD another one by the EVB;
- the First level trigger interface (FLI) module in the SCD crate.

The following information is currently inserted into the physics dataflow: O4SBOR, O1SETU, C5HIST, O5CSTA and O4SEOR ADAMO banks. O5CSTA bank is enhanced with HERA information.

These banks are identical to those inserted into the dataflow prior to 1994 by the C_RECS-TLT exchange system using the CDAQ-TLT shared disk. The shared disk and C_RECS are now obsolete.

UDP based Zeus Message Passing (ZMP) system is used for all inter-process communications. The TR801 (by CAPLIN) SCSI-TP interface is used in the SCD processes on ZAXP03 and in SCD-to-EVB communication program.

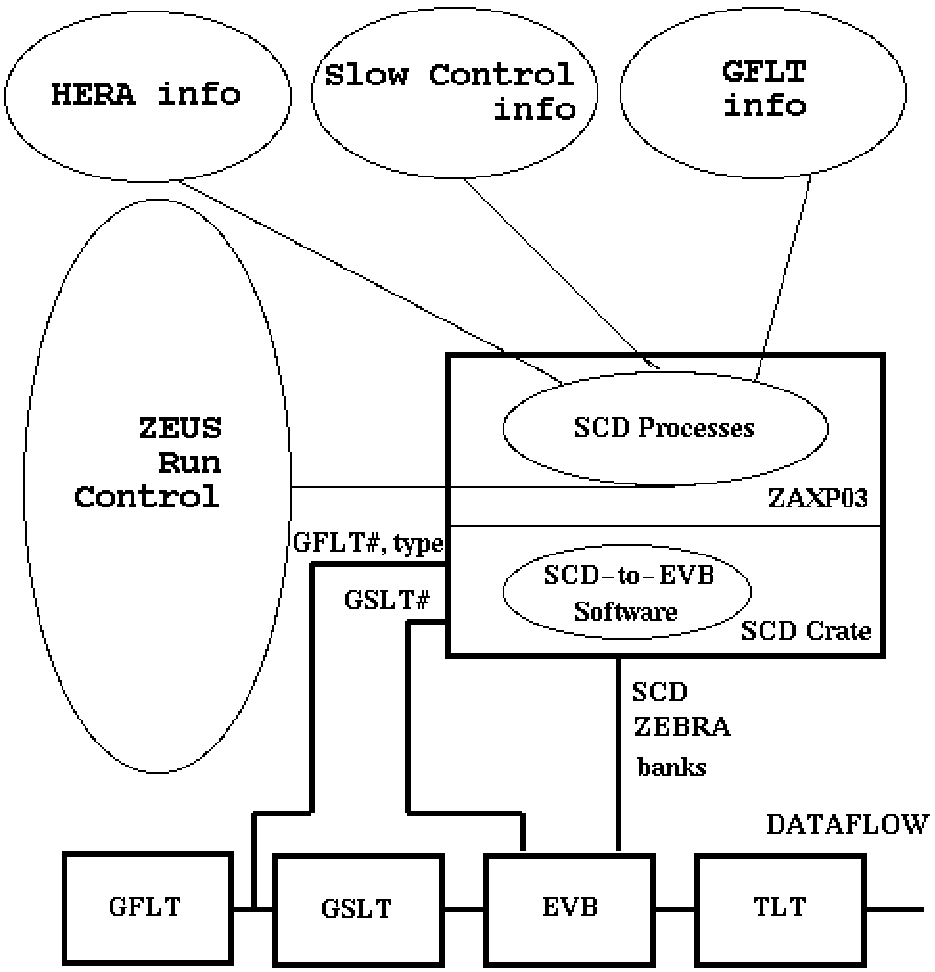

Fig.1. SCD position in ZEUS system

3. SCD structure and locations

The general structure of SCD component hardware is shown in Figure 2. It consists of the next main parts:

- SCD-crate placed in Rack "B" (B-1-12-2), it contains
  - 2TP_VME module for GFLT-to-SCD synchronization process and for SCD-to-EVB communication process (one transputer is used by the SCD, another one by the EVB);
  - First Level trigger Interface (FLI) module for GFLT interface, a custom design module (a spare FLI is available);
- ZAXP03 is the SCD equipment computer, located in room 201 in the HERA south hall, connected to 2TP_VME module in SCD-crate via
  - TR801 device and driver,
  - RS422 to optical links converter.

SCD-CRATE houses 2TP_VME module and FLI module that provides synchronisation feedback to GFLT and can implement most part of work on sampling accepted events numbers, their types, etc.

FLI MODULE. Used in ZEUS GFLT interface (GFLT Document I - interface to component, version 4.0, May 17, 1989) is realised in FLI module for GFLT-SCD interconnection. The FLI module (see Part 2 - 4.1) asserts Bunch Component Number (BCN) every 96 ns. On getting ACCEPT_flag from GFLT the FLI module sets BUSY_flag for one 96 ns clock and stores FLT# (number), BCN, Read-out Type and Test Trigger Type from GFLT in input registers. If no ABORT_flag appears (the event is accepted) the registers contents is stored in FIFOs.

2TP_VME MODULE. SCD uses only one of its transputers. Transputer software designed for the subsystem (see Part 2 - 3.4) runs on it. The second transputer is used for EVB interface in accordance with (Data Flow from the components to the event builder, DESY preprint, DESY 90-127).

The triple port memory of the 2TP_VME modules is used for communication between SCD software and EVB.

TR801. The device is used for program loading in 2TP_VME module and providing it with the SCD banks obtained from the SCD servers.

The 20 MHz link speed is used by SCD.

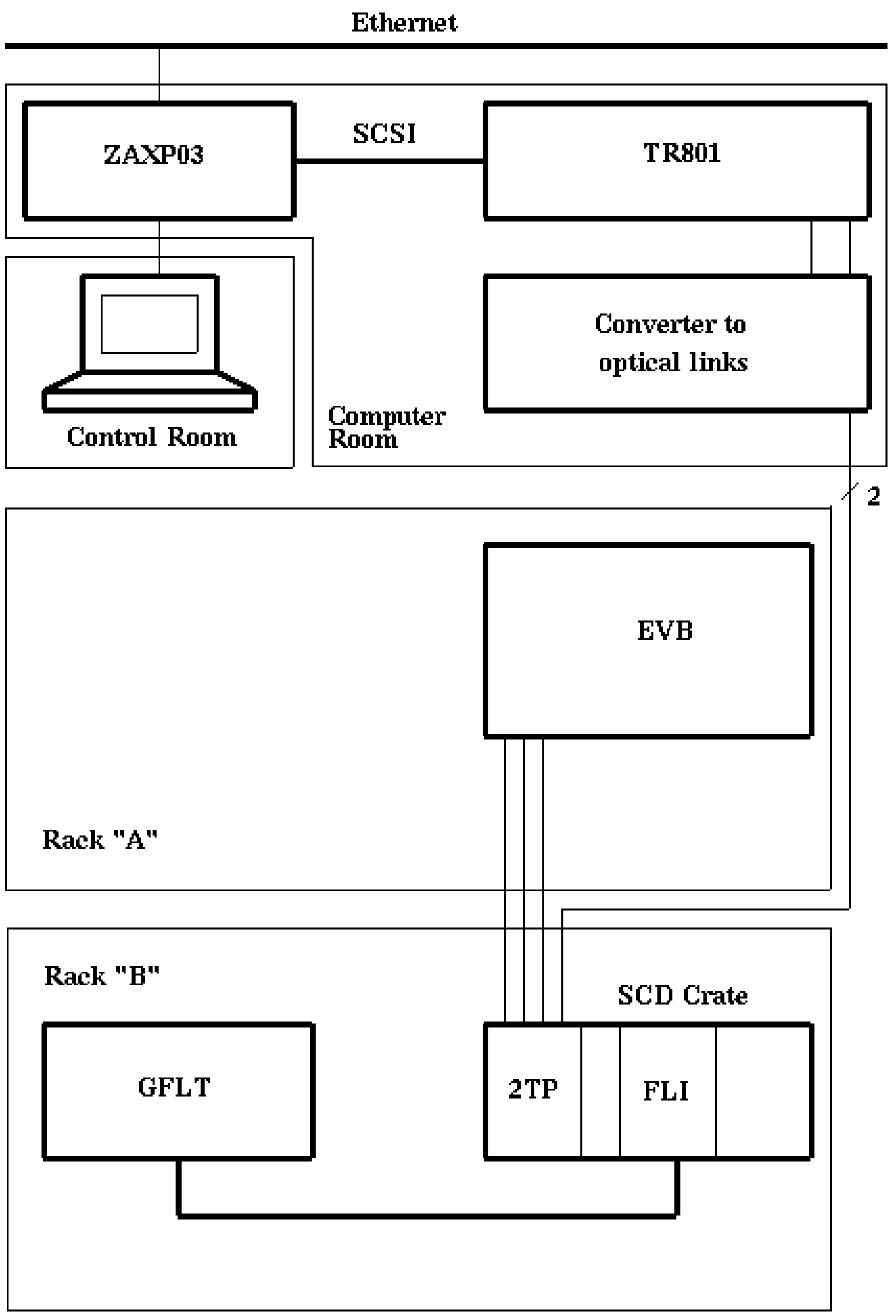

Fig.2. SCD hardware structure

LINKS AND CABLES. GFLT and FLI module are connected through 5 flat cables marked P1-P5 in accordance with the FLI connectors.

One optical link (there are two, 50m each) is usually used for connection 2TP_VME module to TR801 for 2TP and VAX-station programs interconnection. There are two flat cables of 5m each for the connection between the TR801 and the optical link converter.

Part II. IMPLEMENTATION

1. Working configuration

The following equipment should be powered:
- the crate B-1-12-2 (rucksack second floor),
- the EVB crate in the computer room (201) - the optical link adapter is there.

The following computers should be running:
- ZAXP03,
- ZAXP04,
- ZWS005.

The following processes should be running on
- ZAXP03 : SCDC, SCDRC, SCDZC, SCDTR, SCDSS;
- ZAXP04 : O1SETU;
- ZWS005 : O5CSTA, C5HIST, O4SBOR, O4SEOR, HERAIN;

and
- Links from TR801 and EVB should be connected to LA-card in VME-crate in Rack"B" (B-1-12-2);
- 2TP_VME module and FLI module should be installed into VME-crate in Rack"B" (B-1-12-2);
- Cabels from GFLT should be connected to FLI module.

2. Run sequence actions and activity

Initialization:
- executes command file SCD$DIR:SCD$INIT.CMD.

SETUP command:
- executes command file SCD$DIR:SCD$SETUP.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to the IDLE state otherwise.

ACTIVATE command:
- executes command file SCD$DIR:SCD$ACTIVATE.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to the IDLE state otherwise.

PAUSE command:
- executes command file SCD$DIR:SCD$PAUSE.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to IDLE state otherwise.

RESUME command:
- executes command file SCD$DIR:SCD$RESUME.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to IDLE state otherwise.

END command:
- executes command file SCD$DIR:SCD$END.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to IDLE state otherwise.

ABORT command:
- executes command file SCD$DIR:SCD$ABORT.CMD;
- sends STATE OK to RC in case of the normal completion, sends STATE FAILED and goes to IDLE state.

SHUTDOWN command:
- execute command file SCD$DIR:SCD$SHUTDOWN.CMD

3. SCD software

3.1. General description

The SCD software main parts are shown on Fig.3, the subsystem consists of four main parts:
- the SCD processes (SCDC, SCDRC, SCDZC, SCDTR) on ZAXP03;
- the SCD run summary producer (SCDSS) on ZAXP03;
- number of data taking servers (like O4SBOR - one server for each bank);
- SCD-to-EVB communication program (2TP_VME SCD task) based on 2TP_VME module, one transputer is used by SCD another one by EVB.

The tasks to be solved by SCD software are the following:
1) to collect the information required;
2) to transfer this information via the network to the SCD processes;
3) to transfer this information into transputer module's running program;
4) to get GSLT trigger number and trigger type from the EVB fanout;
5) to assemble banks together and declare a event for the EVB;
6) to perform all actions according to Zeus Run Control state.

These tasks are solved by the following processes:
1 - data taking servers;
2 - SCDZC process;
3 - SCDTR process;
4,5 - 2TP_VME based software;
6 - SCDRC, SCDC, SCDZC and SCDTR processes.

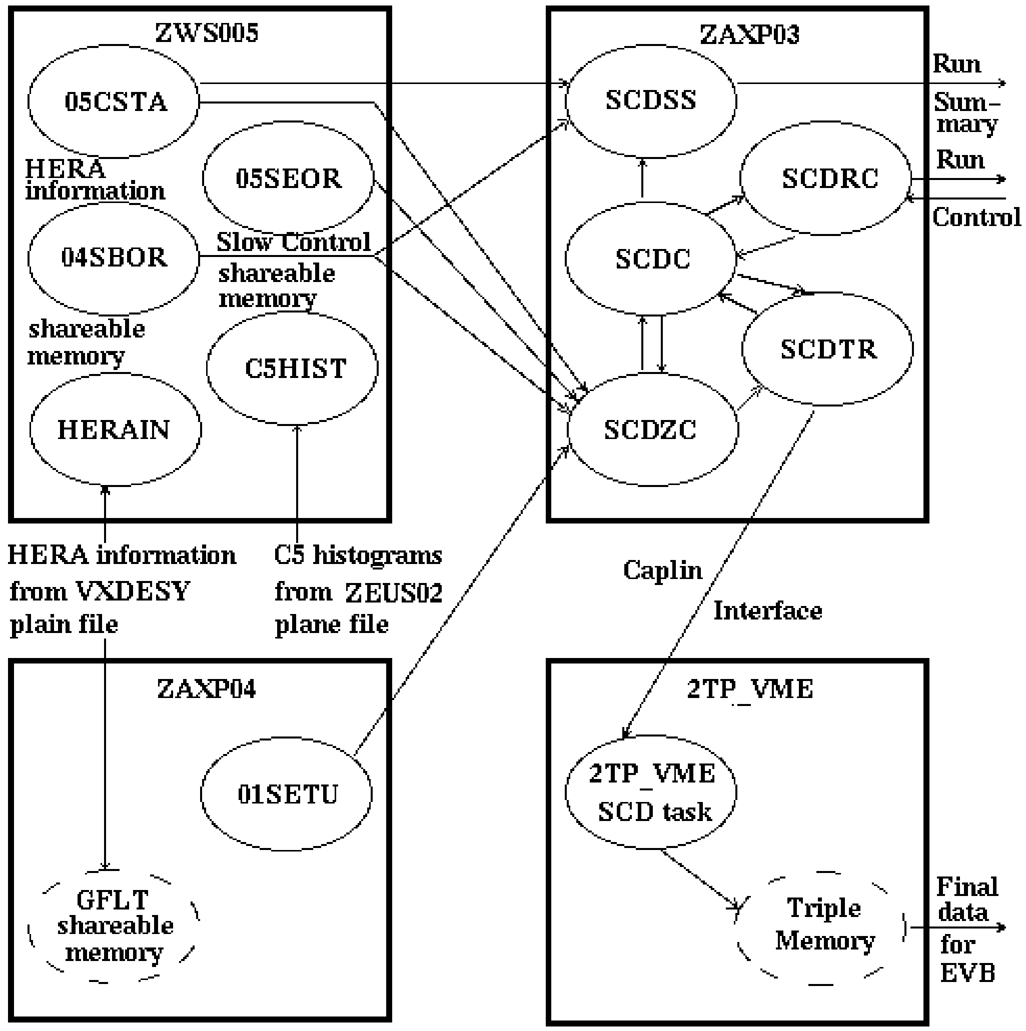

Fig. 3. SCD software structure

The SCD is controlled by ZEUS Run Control.

The SCD is a quite flexible system. The only thing is 'hardcoded' - the run states diagramm (generally speaking IDLE-READY-ACTIVE-IDLE states and SETUP-ACTIVATE-END(ABORT) transitions). The rest of the configuration information is loaded at the run time (with SETUP command).

The SCD is a quite flexible system. The only thing is 'hardcoded' - the run states diagramm (generally speaking IDLE-READY-ACTIVE-IDLE states and SETUP-ACTIVATE-END(ABORT) transitions). The rest of the configuration information is loaded at the run time (with SETUP command).

3.2. Data taking servers

All servers use ZMP for network communications. One server is used for one bank. Every server may support information for any number of clients. Every server currently supports two different requests - the first is "return to sender", the second is more complex - a client may register itself and then the server will send bank periodically (according to the client wishes) until the client cancel registration or a communication error occurs. All the servers send their banks with a predefined header. That header is used by the SCD only. Data itself are represented in network format.

3.2.1. O5CSTA server

This server supplies all the data collected by ZEUS Slow Control system and some general HERA information (available via Netmex). Is running on the SCM computer (ZWS005 now).

3.2.2. O4SBOR server

This server supplies some general information is to be used for the events reconstruction. Is running on the SCM computer (ZWS005 now). This server requests the Run Control infoserver for some additional information.

3.2.3. O4SEOR server

This server is very similar to O4SBOR but the data are supplied with End_Of_Run trigger. Is running on the SCM computer (ZWS005 now).

3.2.4. C5HIST server

This server supplies two raw data arrays from C5 counter. Runs on ZWS005 now.

3.2.5. O1SETU server

This server supplies the GFLT slots prescale information. Is running on the GFLT computer (ZAXP04 now).

3.2.6. HERAIN server

This server collects HERA Netmex information and prepeares it for O5CSTA one. Is running on the SCM machine (ZWS005 now). VXDESY::DISK$MKI:[NETMEX]HERAINFO.MEX file was used during 1995.

3.3. The SCD processes

The SCD itself consists on four processes - SCDRC for the Run Control communication, SCDZC for the datataking servers communication, SCDTR for the 2TP_VME board (via TR801) communication and SCDC for the common scheduling, monitoring and logging. There is an additional process SCDSS to produce the run summary. A separate process is running as a batch job (executes command procedure) to restart any SCD process if that has gone. All of them are running currently on ZAXP03. All of them use ZMP for intercommunication.

3.3.1. SCDRC

SCDRC performs the SCD - Run Control (RC) communication. SCDRC uses ZMP for both RCM and SCDC communications. SCDRC does the following:
- remembers the current state of the SCD system (as run participating component);
- receives commands or messages from RCM;
- checks it against the allowed command list an send warning message to RCM if command is not allowed in the current state or is unknown at all;
- replies immediately if the command is GIVE_STATUS;
- passes the Run Control command to SCDC (if the check-in succeeded);
- passes 'STATE' OK or 'STATE' FAILED messages from SCDC to RCM;
- passes error and diagnostic messages from SCDC to RCO.

3.3.2. SCDZC

SCDZC is a client for all SCD datataking servers. It takes care about all datataking server operations and transfers the received data to SCDTR. It notifies SCDC about succeeded or failed operations too. SCDZC performs its own SETUP-ACTIVATE-END(ABORT) actions (i.e. SCDZC expects that command from SCDC and reports "OK" or "FAILED" result). The instructions for "WhatToDoForEachTransition" are placed in ASCII file SCD$DIR:SCDZC.CONF.

3.3.3. SCDTR

SCDTR performs SCD-2TP_VME program communication. SCDTR uses CAPLIN software and the TR801 device for communications with 2TP_VME module and ZMP for all others communications. SCDTR performs its own SETUP-ACTIVATE-END(ABORT) actions (i.e. SCDTR expects that command from SCDC and reports "OK" or "FAILED" result). SCDTR does the following for SETUP:

- allocates transputer link and boots an executable file into 2TP_VME module,
- reads "WhatToDoForEachTrigger" ASCII file (SCD$DIR:SCDTR.CONF) and instructs 2TP program with the extracted information,
- sends a test request to 2TP program,
- reports SETUP OK or SETUP FAILED;

for ACTIVATE:

- transfers data (received from SCDZC) to the 2TP program,
- sends a test request to the 2TP program periodically and reports it to SCDC,
- reports ACTIVATE OK or ACTIVATE FAILED;

for END or ABORT:

- stops the 2TP program,
- reports END OK or END FAILED.

3.3.4 SCDC

SCDC is belonged to scheduling and monitoring all the SCD activity. SCDC decides what to do for every command from RCM and what to reply to RCM according to the SCDZC and SCDTR activity. SCDC can serve as a communication switchboard for all SCD stuff - you may issue a command from RCO to any process in the SCD system - like that

CORE: SCDTR: SETUP (SETUP will be delivered to SCDTR)
or
CORE: SCDZC: C5HIST: EXIT (EXIT will be delivered to C5HIST server).

SCDC does the following for every transition command from RCM:
- reads SCD$DIR:SCD$'TRANSITION'.CMD ASCII file;
- executes every line from that file as command (usually they are commands for SCDZC or SCDTR, like SETUP or END, and instructions for SCDC itself - whether should SCDC wait for SCDCZ or SCDTR reply or not and if should wait that how long);
- reply to RCM with 'TRANSITION' OK or FAILED according to given instructions;

This scheme allows you to change almost everything in the SCD behavior without code changing (only add or remove lines in *.CMD file).

Another SCDC duty is to maintain SCDLOGDIR:SCD.LOG logfile.

3.3.5 SCDSS

SCDSS produces the Run summary file SCDRUNDIR:SCD_BEG_<n>.RUN$SUMMARY for every ACTIVATE transition and SCDRUNDIR:SCD_END_<n>.RUN$SUMMARY for every END/ABORT transition (where <n> is the run number). SCDSS uses O5CSTA and O4SBOR servers for the data collection.

3.3.6 SCD daemon

All the SCD processes are running as detached processes and there is additional facility to keep all of them alive - the SCD daemon. That's command procedure SCD$DIR:SCD_DAEMON.COM which runs as a batch job. The batch queue name is ZAXP03$BATCH, and the jobname is SCD$DAEMON (currently). It monitors processes existence and restarts them if necessary. SCDC have to be restarted only if SCDRC, SCDZC and SCDRC are already running.

3.4 2TP_VME based software

This part of the SCD component is the endpoint of the "SCD pipeline". It supplies the data to the EVB when required. SCD software is running on the 2TP_VME X transputer and EVB software is running on the 2TP_VME Y transputer. The 2TP_VME module triple memory is the place where SCD leaves the data banks for the EVB.

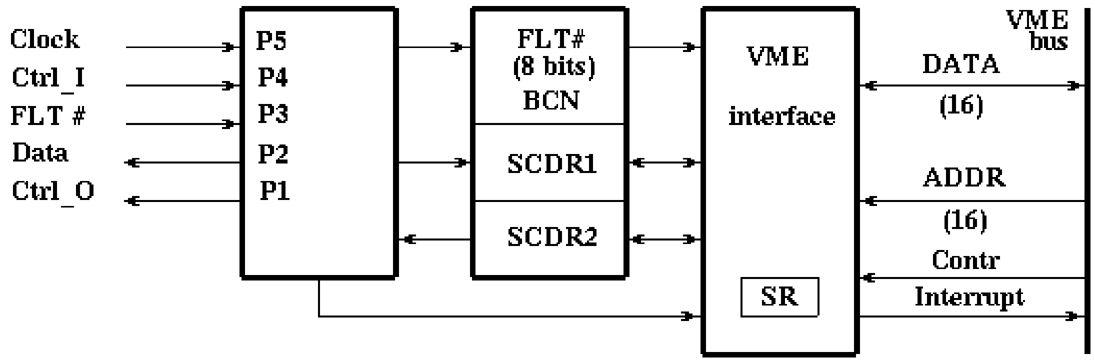

Fig.4. FLI block-diagram

The SCD transputer program, written in CAPLIN parallel C performs number of concurrent actions. These actions, performed by separate threads, are:
- to wait for data banks from SCD (via external link from TR801) and to store them in internal buffers;
- to wait for the GSLT decision (via external link from the EVB fanout);
- if GSLT decision is "accepted" then to generate the ZEBRA physical record for the EVB and declare the event for the EVB.

4. First Level Trigger Interface Module

4.1. FLI functions

The module provides Slow Control data for GFLT and yield FLT# and BCN of accepted events for GSLT in the Slow Control Data taking component.

The FLI block-diagram is shown in Figure 4. The FLI is a A16/D16 slave module. The card contains four registers:
- Status Register (SR,w/r);
- two Slow Control Data Registers (SCDR,w/r);
- FLT# Register (r).

SCDRs(w) are used for storing Slow Control data sent to GFLT, SCDR2(r) is also used for storing Trigger Type from GFLT. FLT# Register stores BCN and FLT# received from GFLT.

FLT-logic is implemented as it is shown in Figure 9 of GFLT Design Document I - interface to component - (version 4.0, May 17, 1989). The duration of BUSY is 1-clock (96 ns). Upon ACCEPT of GFLT (no FAST CLEAR) FLI sets a VME interrupt (if not masked). In case of internal FIFOs for GFLT trigger type, event number and BCN are full the card keeps BUSY asserted until the FIFOs read out via VME (at least once).

4.2. FLI features

The card consists of two standard-size VME-boards (A and B). VME-interface is implemented on the PC-board A only. An output buffer for FLT#-register and SCDR1 (read out) are placed on the PC-board B.

4.2.1. Front panel

Five 17-pair flat cable input/output. They are:

P1	Component_control_signal cable	(output, PC-board A),
P2	Component_data_signal cable	(output, PC-board A),
P3	FLT_number_signal cable	(input, PC-board B),
P4	GFLT_control_signal cable	(input, PC-board B),
P5	GFLT_clock_signal cable	(input, PC-board B).

4.2.2. Power requirements

The card uses +5V and -12V lines of the VME-bus. The current consumption of the module boards is listed on Table 1.

Table 1

Board	**+5 V**	**-12 V**
A	**0.7 A**	**1.75 A**
B	**0.75 A**	**0.5 A**

4.2.3. Twisted pair signals

4.2.3.1. Drivers and Receivers

All signals in the five 17-pair flat cables are differential ECL type signal. MC10124 and MC10125 are used for drivers and receivers respectively.

4.2.3.2. Cable assignment

Output to GFLT.

Cable 1 (FLI Control)

0-2	Reserved	
3	Busy	
4	Fatal_Error_Bit	
5	Trigger_Error_Bit	
6-7	Ambiguity Range	
8-15	Component_Bunch_Crossing_Number	(8-lsb,15-msb)
16	Timing Marker	

Cable 2 (FLI data)

0-15	SCS Data	(0-lsb,15-msb)
16	Timing Marker	

Input from GFLT.

Cable 3 (FLT#)

0-7	FLT_Number	(0-lsb, 7-msb)
8-15	GFLT_Bunch_Crossing_Number	(8-lsb,15-msb)
16	Reserved	

Cable 4 (Control)

0	Reserved	
1	Test_Enable	
2-4	Test_Trigger_Type	
5-6	Readout_Type	(system control)
7-8	Readout_Type	(crossing ambiguity)
9-11	Readout_Type	(component dependent)
12	Abort Flag	
13	Accept Flag	
14-16	Reserved	

Cable 5 (Clock)

0-3	Reserved	
4	ECL-level clock	
5	Crossing_Number_0_Indicator	
6	Proton_Empty_Bunch_Indicator	(not used)
7	Electron_Empty_Bunch_Indicator	(not used)
8-15	Reserved	

4.3. VME interface

1. A16/D16 slave (Supervisor and Data access modes).
2. Address location.
The module decodes A15-A08 and A02,A01 address lines. The A07-A03 address lines should be all "zero". A user can set the A15-A08 values with a DIP switch. The lines A02,A01 are decoded for four internal registers.
3. Only word access is permitted.

4.3.1. Status Register

(Location 00)

Writing

Bit 0	program initialization	(writing 0)
Bit 1	Busy Flag to GFLT lock	(writing 1)
Bit 2	Fatal Error Flag to GFLT	(writing 1)
Bit 3	Trigger Error Flag to GFLT	(writing 1)
Bit 4	Ambiguity Range 0 to GFLT	(writing 1)
Bit 5	Ambiguity Range 1 to GFLT	(writing 1)
Bit 6	Interrupt Request Mask	(1-interrupt enable)
Bit 7-15	not used	

Reading

Bit 0-2	Test Trigger Type from GFLT	
Bit 3	Test Trigger Flag from GFLT	
Bit 4	BUSY flag to GFLT	
Bit 5	Half Full Flag of TT FIFO	
Bit 6	Interrupt Request Mask	
Bit 7	Interrupt Request	(1 - request)
Bit 8	Empty Flag of TT FIFO	
Bit 9	Full Flag of TT FIFO	
Bit 10	Half Full Flag of CN FIFO	
Bit 11	Empty Flag of CN FIFO	
Bit 12	Full Flag of CN FIFO	
Bit 13	Half Full Flag of FLT FIFO	
Bit 14	Empty Flag of FLT FIFO	
Bit 15	Full Flag of FLT FIFO	

4.3.2. Slow Control Data Register 0

(Location 01)

Bit 0-15 (w/r) - Slow Control data to GFLT (0-lsb,15-msb).

4.3.3. Slow Control Data Register 1

(Location 10)

Writing

In this FLI-card version the register is not used for sending Slow Control data to GFLT.

Reading

Bit 0-2	Readout Trigger Type from GFLT	(component dependent)
Bit 3,4	Readout Trigger Type from GFLT	(crossing ambiguity)
Bit 5,6	Readout Trigger Type from GFLT	(system control)
Bit 7-15	not used	

4.3.4. FLT# Register

(Location 11)

Bit 0-7 (r)	FLT# from GFLT	(0-lsb, 7-msb)
Bit 8-15(r)	BCN from GFLT	(8-lsb,15-msb)

4.4. Operation

After the card initialization (Power Up or Program) the Status register and the SCDR1 are loaded via VME-bus. The card asserts BCN (P1) and Slow Control data (P2) every 96 ns. On getting ACCEPT_flag from GFLT the FLI-card sets BUSY_flag for one 96ns-clock and stores FLT#, BCN, Readout Type and Test Trigger Type from GFLT in FIFOs the flags of which can be read from the Status register. If no ABORT_flag appears (the event is accepted) the card sets an interrupt request (Bit 7, Control Register, priority is set by switches). The VME-master (2TP-VME module) serves the request and reads out the FLT# register and SCDR1. The card can send flags of abnormal job to GFLT via writing some bits in the Status register, or in case of internal FIFOs for GFLT trigger type, event number and BCN are full the card keeps BUSY asserted until the FIFO read out via VME (at least once).

Part III. USER GUIDE

The SCD is a full automated system (well, somebody have to purge log files and delete or backup run summaries once per year) and it is fired up with the system on relaited machines. So it must be always running if the computers are in the working state.

The easiest way to see the SCD state - click here for ZAXP03/ZAXP04 and here for ZWS005. You would see the same if login as ONLINE on ZAXP03, ZAXP04 or ZWS005 and type

```
$ @scd$dir:show_scd
```

The first aid is described in "What-to-do-when" page.

1. Startup and shutdown

When ZAXP03, ZAXP04 or ZWS005 boots the SCD$DIR:STARTUP_DETACH.COM is executed. To start (or restart) the whole system by hand login all the machines mentioned above as ONLINE and type

```
$ @scd$dir:start_detached
```

(Note: not SCD$DIR:STARTUP_DETACH.COM !)

To stop the working system send the following commands from RCO window

```
CORE: SCDZC: O5CSTA: EXIT
CORE: SCDZC: O4SBOR: EXIT
CORE: SCDZC: O4SEOR: EXIT
CORE: SCDZC: C5HIST: EXIT
CORE: SCDZC: O1SETU: EXIT
CORE: SHUTDOWN
```

and then stop HERAIN process on ZWS005 by hand. As an alternative way you may type on ZAXP03

```
$ @scd$dir:shutdown_scd
```

and answer "YES" to all questions. This procedure will stop everything even when ZMP is down or processes hung up.

You may stop all the processes by hand in a lack of all communication.

Remember! All things will be respawned in 2-4 min. If you want to stop it forever type

$ define /group SCD$RESTART NO

to disable automatic restart. To enable it again type

$ define /group SCD$RESTART YES

2. Configuration

What are tuneable things for the SCD ? You may change the bank set to collect, the way to collect that, GSLT trigger types to be understandable for the SCD and the bank set for every trigger type. You may change some internal setting for the SCD - TR801 link to use, transputer executable file name and so on. You may execute DCL command procedures or run programms in the transition time.

What is the configuration file set for the SCD? There are two groups of file - SCD$DIR:*.CMD files and SCD$DIR:*.CONF files. Recollect the run sequencing page. Every time the SCD gets 'TRANSITION', INIT or SHUTDOWN command from the Run Control it reads appropriate *.CMD file line by line and makes certain actions. Let us look up throught the files. Common things to remember - one command occupies one line (you can't break the line with hyphen or newline), comment signs for all the SCD ACSII files are # or ! in the first column (you may use any). Comma and asterisk (, and *) are the word separators (whitespace isn't one!).

The first token is command name, all others - parameters.

This is a Command file for the INIT procedure /*comment*/

```
REMAP*              /* SCDC have to ZMP_remap()
                                 SCDRC, SCDZC and SCDTR     */
ZC*REMAP*           /* send REMAP to SCDZC                  */
TR*REMAP*           /* send REMAP to SCDTR                  */
RC*REMAP*           /* send REMAP to SCDRC                  */
RC*MESSAGE the SCD initialized*
                    /* send message to RCO via SCDRC        */
# End
# This is a Command file for SETUP transition
REMAP*
ZC*CONFIG SCD$DIR:SCDZC.CONF*
                    /* send configuration file name
```

```
                            to SCDZC                    */
ZC*REMAP*
ZC*SETUP*          /* send SETUP to SCDZC               */
TR*REMAP*
TR*LINK GKA100* /* send TR801 device name to SCDTR      */
TR*APP SCD$DIR:VFAST.APP*
                   /* send 2TP_VME executable file name
                            to SCDTR                    */
TR*CONFIG SCD$DIR:SCDTR.CONF*
                   /* send configuration file name to SCDTR */
TR*SETUP*          /* send SETUP to SCDTR               */
WAIT$ZC*           /* wait for SCDZC report before send OK
                            or FAILED to Run Control    */
WAIT$TR*           /* wait for SCDTR report before send OK
                            or FAILED to Run Control    */
TIMEOUT*100*       /* if in 100 sec SCDTR and SCDZC doesn't
                            answer OK then send
                            FAILED to Run Control       */
*SCD$DIR:SETUP$SUMMARY*
                   /* execute @SCD$DIR:SETUP$SUMMARY
                            DCL command                 */
# End
```

Note then SCDC doesn't stop and wait at
WAIT$ZC*
WAIT$TR*
TIMEOUT*100*
points but just stores those instructions and then follows them.

All other *. CMD files have the same format. Look at appendix B for a full command list.

SCDTR and SCDZC read their configuration files (the names for the files is to be passed with CONFIG commands - see above) in the SETUP transition time. Look at the string in SCDZC.CONF (. , : are the word separators)

```
O5CSTA :SCD_TEAM,O5CSTA,8,0,0,8,0,
.        .        .       . . . . .<- A rule for ABORT transition
.        .        .       . . . .<--- A rule for RESUME transition
.        .        .       . . .<----- A rule for PAUSE transition
.        .        .       . .<------- A rule for END transitoin
.        .        .       .<--------- A rule for ACTIVATE transition
.        .        .<----------------- ZMP agent name for the dataserver
.        .<-------------------------- ZMP domain name for the dataserver
.<----------------------------------- Bank name (note trailing whitespaces)
```

SCDZC given with SETUP command reads configuration file and determinate what is the set of banks to be collected, the dataserver domain.agent name for each bank and command is to be send to the dataserver for every transition. The given line is translated like that: bank name is 'O5CSTA ' (bank name must be 8-characters long), dataserver to ask for one is SCD_TEAM O5CSTA , the dataserver will be given with START 8 command for ACTIVATE and RESUME transition and START 0 (equivalent to STOP) for all other transitions.

PolarD :SCD_TEAM,DUMMY,ONCE,0,0,AUTO,0,POLAR,HELLO

There is a sample line with two optional fields - for the command header and trailer. This line is translated into: bank name is 'PolarD', dataserver to ask for one is SCD_TEAM DUMMY , the dataserver will be given with POLAR BANK HELLO command for ACTIVATE transition, POLAR START AUTO HELLO for RESUME and POLAR START 0 HELLO for all other transitions. Headers or/and trailers may be useful when you use one dataserver for the set of different banks.

Lets look at SCDTR.CONF file. Note that all the information just passed by SCDTR into transputer running programm. SCDTR reads the file parses it and fills out a binary table is to load into the transputer. There is three lines conserning to Begin_of_run GSLT trigger (. , : are the word separators)

```
0XC0000000:$REAL$:O4SBOR ,O5CSTA ,C5HIST ,O1SETU .
0XC0000000:$DUMMY$:PolarD .
0XC0000000:$TIMEOUT$:400.
```

The first word 0XC0000000 is a GSLT trigger type bits pattern, BOR trigger in this case. Keyword $REAL$ says that listed banks (O4SBOR , O5CSTA , C5HIST and O1SETU) have to be really passed into the transputer at list once. Otherwise transputer programm will wait until timeout expiration (timeout defined with $TIMEOUT$ keyword, 400 corresponds ca. 20 sec) and then declare event for the EVB with the listed banks. The banks what didn't arrive will be declared as zero-length. The banks listed after $DUMMY$ keyword are not expected to arrive at all and always declared as zero-length (reserverd for debugging). The keyword $DEFAULT$ defines a bank set is to be declared into the event for all the triggers which are nod defined explicitly (normal trigger in the current configuration).

The small conclusion - you have to add a new line into SCDZC.CONF and include the bank name for desired triggers into SCDTR.CONF in order to add a new bank in the SCD configuration.

3. Diagnostic and manual control

All the information you need (if you are not the expert) you can get with

```
$ @scd$dir:show_scd
```

command on ZAXP03, ZAXP04 and ZWS005 machines or click the line in the SCD home page. You may find a useful information looking into SCDLOGDIR:SCD.LOG file. The SCD open a new version of the log-file with every END or ABORT command (and when starting of course). All run-time errors and messages are sent to RCO.

You may send any command described (see appendix A) via RCO programm. You don't need to do it under normal conditions.

4. How to see the SCD data

It's reasonable if you would like to see the SCD data stored on the archive machine and use them. You may make use the offline tools such EAZE or TI. You may also see all the data online clicking the lines in the SCD home page.

There is an example for EAZE job (main part of those scripts was written by Mark Lancaster, University of Oxford). You should be sure that the SCD was in the chain while the run wanted. Otherwise the data woun't be found. You should do the following:

1. Copy cdaq_submit, cdaq_go and cdaq.car in your directory on ZOW cluster
2. Supply a wanted filename for the line
   ZeusIO-INFI ZEUS94.RAW.D940615.T1222.R008736C
   in cdaq_go and wanted run and event numbers for the lines
   IF ((RUNNUMBR .EQ. 8737) .OR. (RUNNUMBR .EQ. 8859)) THEN

IF (EVENTNUMBR .EQ. 1) THEN
in cdaq.car

3. Type in
zow01:/zow/userd/jon/example> cdaq_submit
Submitting Job to ZARAH
Job 21128 submitted
4. To see the queue
zow01:/zow/userd/jon/example> jobq
5. After job completion type in
zow01:/zow/userd/jon/example> jobget 21128 -f cdaq.log
and you will find the data in cdaq.log
6. To remove files from the spool zone
zow01:/zow/userd/jon/example> jobpurge 21128
See cdaq_submit, cdaq_go, cdaq.car and cdaq.log texts in Appendix E.
There is an example for TI.

1. Login into ZARAH1
2. Check if wanted file is present in zpool or zdata
zarah1:/zow/userd/jon> sfget -pzpool -k ZEUS94.RAW.D940816.T0315.R009457C
File 'ZEUS94.RAW.D940816.T0315.R009457C' not found in pool 'zpool'
3. Get wanted file from IBM into zpool area
zarah1:/zow/userd/jon>/zeus/bin/stagei ZEUS94.RAW.D940816.T0315.R009457C
4. Find out where the file is
zarah1:/zow/userd/jon> sfget -pzpool -k ZEUS94.RAW.D940816.T0315.R009457C
/shift/zarah1/data15/zeus/jon/ZEUS94.RAW.D940816.T0315.R009457C
5. Logout ZARAH1
6. Login ZOW
7. Ftp wanted file from ZARAH1 to ZOW
zow01:/zow/userd/jon> ftp zarah1
Connected to zarah1.
...
ftp> binary
200 Type set to I.
ftp> cd /shift/zarah1/data15/zeus/jon
250 CWD command successful.
ftp> get ZEUS94.RAW.D940816.T0315.R009457C data
200 PORT command successful.
150 Opening BINARY mode data connection for ZEUS94.RAW.D940816.T0315.R009457C
... wait for a while
8. Start TI when ready
zow01:/zow/userd/jon> ti

```
!————————————————————————
!ADAMO/TI : Table Interaction
!
!TIP without HISTOGRAM, LOOP, LOAD and all PAW commands
!————————————————————————
ADAMO/TAP version 3.3 with GAF version 3.2 starting
ORACLE Driver added
... starting KUIP ...
... try to execute your TI kumac file ...
*** Unknown file ti.kumac
Enter HELP TI for help
TI>
```

9. Open your file
   TI> gaf/open f name=data,filform=EXCH,driver=FZ
   (f is alias here)
10. Get the first event (it should be BOR for any file)
   TI> gaf/next f
   Record found: ZEUSONLINE5
   GAFTyp : BOR
   Nr1 : 9457
   Nr2 : 0
   TStam11 : 0
   TStam12 : 0
   TStam21 : -1073741824
   TStam22 : 0
11. Access this event
   TI> acc f
12. Have enjoy with the data by
   TI> tab/pri ALL table_name

SUPPLEMENT: SCD Commands and Messages

1. Commands may be sent to the SCD from RCO
(prefix RCO is not shown)

* RECONNECT

SCDRC tries to map ZEUS_RUN_CONTROL RC_MAIN (or another ZMP address given before with DOMAIN and AGENT commands from SCDC).

* GIVE_STATUS

SCDRC replies with "SCD STS %s ONLINE".

* SETUP

SETUP command is passed to SCDC and the SCD performs SETUP transition. SCDRC sets logical name RC$NUMBER to the run number.

* ACTIVATE

ACTIVATE command is passed to SCDC and the SCD performs ACTIVATE transition.

* PAUSE

PAUSE command is passed to SCDC and the SCD performs PAUSE transition.

* RESUME

RESUME command is passed to SCDC and the SCD performs RESUME transition.

* END

END command is passed to SCDC and the SCD performs END transition.

* ABORT

ABORT command is passed to SCDC and the SCD performs ABORT transition.

* EXIT

EXIT command is passed to SCDC.

* REMAP

SCDRC tries to map ZEUS_RUN_CONTROL RC_MAIN (or another ZMP address given before with DOMAIN and AGENT commands from SCDC) and SCDC process (SCD_TEAM SCD_CORE).

* CORE:

SCDRC passes the rest of the command to SCDC.

* INIT

Is silently ignored now.

* MAIL

Is silently ignored now.

All the commands listed above may be sent right from the RCO window or automatically by the Run Control programm. All the command listed below are the internal commands and given for an information only. Almost all of them are used for configuration.

2. Commands may be sent to SCDRC from SCDC

* EXIT_RC

SCDRC sends "SCD STS UNDEF OFFLINE" to the Run Control and exits.

* REMAP

SCDRC tries to map ZEUS_RUN_CONTROL RC_MAIN (or another ZMP address given before with DOMAIN and AGENT commands from SCDC) and SCDC process (SCD_TEAM SCD_CORE).

* STATE OK

SCDRC updates the stored run state value (if there is a transition state) and sends "SCD STS %s ONLINE" with the current run state.

* STATE FAILED

SCDRC sends "SCD ERR Failed sends "SCD STS IDLE ONLINE".

* DOMAIN

SCDRC takes the second word as a new value for the Run Control programm domain (ZEUS_RUN_CONTROL is the default value).

* AGENT

SCDRC takes the second word as a new value for the Run Control programm agent name (RC_MAIN is the default value).

* I

SCDRC takes the second word as a new agent name for itself (SCD is the default value).

* TAKE

SCDRC takes the second word as a new run state value and sends "SCD STS %s ONLINE" to the Run Control.

* MESSAGE

SCDRC send the rest of the command to the Run Control as "SCD INF %s".

* WARNING

SCDRC send the rest of the command to the Run Control as "SCD WAR %s".

* ERROR

SCDRC send the rest of the command to the Run Control as "SCD ERR %s".

* FATAL

SCDRC send the rest of the command to the Run Control as "SCD FAT %s".

* LOGFILE

SCDRC send the rest of the command to the Run Control as "SCD LOG %s".

3. Commands may be sent to SCDZC from SCDC

* EXIT

SCDZC exits.

* CONFIG

SCDZC takes the second word as a configuration file name is to be loaded after SETUP. If CONFIG has no argument SCD$DIR:SCDZC.CONF as configuration file name is assumed.

* SETUP

SCDZC loads the configuration file (see CONFIG command) builds a table with the all listed dataservers and maps them. If success then SCDZC sends "MAP %s OK" per each mapped server to SCDC, "REMAP" to all the servers and "SETUP OK" to SCDC and goes to READY state. Otherwise it complaines "MAP

* ACTIVATE

SCDZC sends the requested commands (see "Configuration") to all servers (see SETUP command). If all those commands are delivered then sends "ACTIVATE OK" to SCDC and goes to ACTIVE state. If a send error occures then SCDZC complines "BANK %s BAD".

* PAUSE

SCDZC sends the requested commands (see "Configuration") to all servers (see SETUP command). If all those commands are delivered then sends "PAUSE OK" to SCDC and goes to PAUSED state. If a send error occures then SCDZC complines "BANK %s BAD".

* RESUME

SCDZC sends the requested commands (see "Configuration") to all servers (see SETUP command). If all those commands are delivered then sends "RESUME OK" to SCDC and goes to ACTIVE state. If a send error occures then SCDZC complines "BANK %s BAD".

* END

SCDZC sends the requested commands (see "Configuration") to all servers (see SETUP command). If all those commands are delivered then sends "END OK" to SCDC and goes to IDLE state. If a send error occures then SCDZC complines "BANK %s BAD".

* ABORT

SCDZC sends the requested commands (see "Configuration") to all servers (see SETUP command). If all those commands are delivered then sends "ABORT OK" to SCDC and goes to IDLE state. If a send error occures then SCDZC complines "BANK %s BAD".

* REMAP

SCDZC maps SCDC and SCDTR and exits if error occures.

* DEBUG n

SCDZC sets debug level to n. If there is no n then sets it to 1.

* something:

If the last character of the command is : (without whitespase) the SCDZC tries to send the rest of the command to the dataserver with something name. The dataserver has to be listed in the configuration file and successfully mapped during the last SETUP. Otherwise the command is silently ignored.

4. Commands may be sent to SCDTR from SCDC

* EXIT

SCDTR kills transputer programm and frees the link if loaded and exits.

* CONFIG

SCDTR takes the second command word as configuration file name.

* LINK

SCDTR takes the second command word as device name for the CAPLIN interface.

* SPYLINK

SCDTR takes the second command word as device name for the CAPLIN interface. Is optional (used for debug messages from the transputer programm).

* APP

SCDTR takes the second command word as file name is to be loaded into the transputer.

* SWAP

SCDTR will swap the byte ordering for all the data before transfer them into the transputer. Is used for a pre-compensation (all the data change the byte ordering when are read from the EVB crate on the TLT side via the VME bus). This behavior is default.

* NOSWAP

SCDTR will not swap the byte ordering.

* SETUP

SCDTR loads the configuration file (see CONFIG command) builds a table with given information and store it. Then SCDTR allocates CAPLIN device (see LINK) boots the transputer programm (see APP) and loads the configuration table into the transputer. If there was no error then sends "SETUP OK" to SCDC and goes to READY state. Otherwise it sends "SETUP FAILED" and goes to IDLE state.

* ACTIVATE

SCDTR does nothing. Just replies "ACTIVATE OK" to SCDC and goes to ACTIVE state.

* END

SCDTR kills the transputer programm, frees the CAPLIN device and goes to IDLE state.

* REMAP

SCDTR maps SCDC and SCDZC and exits if error occures.

* DEBUG n

SCDTR sets debug level to n.

5. Commands may be sent from (via) SCDRC to SCDC

* SETUP

SCDC reads and execute SCD$DIR:SCD$SETUP.CMD file line by line. Then SCDC sends "STATE OK" to SCDRC if there was neither "WAIT$TR" not "WAIT$ZC" directive in SCD$DIR:SCD$SETUP.CMD. Otherwise SCDRC waits (waiting time is defined by "TIMEOUT" directive) for reply from SCDZC and/or SCDTR. If the expected replies arrive in time and are positive ("SETUP OK"), SCDC sends "STATE OK" to SCDRC and goes to READY state. Otherwise SCDC sends "STATE FAILED" and goes to IDLE state. If there was any problem with the file SCDC sends "STATE FAILED" and "ERROR Bad file SCD$SETUP.CMD" to SCDRC and goes to IDLE state.

* ACTIVATE

Exactly the same order as with "SETUP" command but the file name is SCD$DIR:SCD$ACTIVATE.CMD.

* PAUSE

Exactly the same order as with "SETUP" command but the file name is SCD$DIR:SCD$PAUSE.CMD.

* RESUME

Exactly the same order as with "SETUP" command but the file name is SCD$DIR:SCD$RESUME.CMD.

* END

Exactly the same order as with "SETUP" command but the file name is SCD$DIR:SCD$ABORT.CMD. Then SCDC closes the logfile SCDLOGDIR:SCD.LOG and opens it again.

* ABORT

SCDC reads and execute SCD$DIR:SCD$ABORT.CMD file line by line and goes to IDLE state. If there was any problem with the file SCDC sends "ERROR Bad file SCD$ABORT.CMD" to SCDRC and goes to IDLE state. Then SCDC closes the logfile SCDLOGDIR:SCD.LOG and opens it again.

* INIT

SCDC reads and execute SCD$DIR:SCD$INIT.CMD file line by line and goes to IDLE state. If there was any problem with the file SCDC sends "ERROR Bad file SCD$INIT.CMD" to SCDRC and goes to IDLE state.

* EXEC

SCDC takes the second command word as the file name and execute it like any from *.CMD files mentioned above.

* REMAP

SCDC maps SCDRC, SCDZC and SCDTR then sends "TAKE %s" to SCDRC to adjust run state if SCDRC has been restarted.

* REMAP_ALL

SCDC maps SCDRC, SCDZC and SCDTR then sends "REMAP" to all of them.

* EXIT

SCDC exits

* SHUTDOWN

SCDC reads and execute SCD$DIR:SCD$INIT.CMD file line by line and exits.

* ZC:

SCDC passes the rest of the command to SCDZC.

* RC:

SCDC passes the rest of the command to SCDRC.

* TR:

SCDC passes the rest of the command to SCDTR.

* @

SCDC takes the second word of the command as filename then spawns subprocess and executes

$ @file_name

DCL command.

6. Messages may be sent from SCDZC to SCDC

* MAP SCDC sends "ERROR Can't map
* SETUP OK

SCDC treats this message as positive reply at SETUP transition.

* SETUP FAILED

SCDC treats this message as negative reply at SETUP transition.

* ACTIVATE OK

SCDC treats this message as positive reply at ACTIVATE transition.

* ACTIVATE FAILED

SCDC treats this message as negative reply at ACTIVATE transition.

* PAUSE OK

SCDC treats this message as positive reply at PAUSE transition.

* PAUSE FAILED

SCDC treats this message as negative reply at PAUSE transition.

* END OK

SCDC treats this message as positive reply at END transition.

* END FAILED

SCDC treats this message as negative reply at END transition.

7. Messages may be sent from SCDTR to SCDC

* SETUP OK

SCDC treats this message as positive reply at SETUP transition.

* SETUP FAILED

SCDC treats this message as negative reply at SETUP transition.

* ACTIVATE OK

SCDC treats this message as positive reply at ACTIVATE transition.

* ACTIVATE FAILED

SCDC treats this message as negative reply at ACTIVATE transition.

* PAUSE OK

SCDC treats this message as positive reply at PAUSE transition.

* PAUSE FAILED

SCDC treats this message as negative reply at PAUSE transition.

* END OK

SCDC treats this message as positive reply at END transition.

* END FAILED

SCDC treats this message as negative reply at END transition.

* ERROR

SCDC passes the rest of the message to SCDRC as ERROR.

* TEST %d %d %d %d %d %d

Statistic and control information from the transputer-runninig programm. SCDC puts this message into the logfile. The meaning of the numbers:

```
TEST %d %d %d %d %d %d
.      .  .  .  .  .  .<– GSLT trigger number*
.      .  .  .  .  .<——- ENV_REC_NUM counter
.      .  .  .  .<———— TPM_FULL counter
.      .  .  .<————— FIFO_FULL counter
.      .  .<——————- GSLT_FULL counter
.      .<——————— counter of TRWriteLink() fails
```

.* (if -2 then "END" command to SCDZC is to be sent by SCDTR)

TRWriteLink() sends a message or a data to the tranpsuter GSLT_FULL arise when there is no more place into internal ring buffer for accepted GSLT decision.

FIFO_FULL arise when there is no more place for GFLT triggers into hardware FIFO in FLI module.

TPM_FULL arise when there is no more place into the transputer triple memory (the EVB can't send the data).

ENV_REC_NUM incremented with every Environment_record trigger from GSLT All counters are cleared every time the message has been sent.

* NO FLI

This messge is sent when no the first level trigger interface module detected. The reasons - no module is installed into the crate or the module is dead or cables are unplugged or there is no clock pulses from the GFLT. SCDC puts this message into the logfile.

* YES FLI

This messge is sent when "NO FLI" error is cleared. SCDC puts this message into the logfile.