Hybrid Test Environment for GPRS-handling in Regional Processors

Jörgen Dahlqvist

LiTH-ISY-EX-3062

14 August, 2000

Hybrid Test Environment for GPRS-handling in Regional Processors

Master thesis performed in telecommunication
at Linköping Institute of Technology
by

Jörgen Dahlqvist

LiTH-ISY-EX-3062

Supervisor: Jonas Waldeck
Ann-Charlotte Linderson
Examiner: Lars Nielsen
Linköping, 14 August, 2000

Avdelning, Institution
Division, department

Department of Electrical Engineering

Datum
Date

2000-08-14

Språk
Language

❑ Svenska/Swedish
☑ Engelska/English

❑ _____

Rapporttyp
Report: category

❑ Licentiatavhandling
☑ Examensarbete
❑ C-uppsats
❑ D-uppsats
❑ Övrig rapport
❑ _____

URL för elektronisk version

ISBN

ISRN

Serietitel och serienummer
Title of series, numbering

ISSN

LiTH-ISY-EX-3062

Titel
Title

Hybrid Test Environment for GPRS-handling in Regional Processors
Hybrid testmiljö för GPRS-hantering i regionala processorer

Författare
Author

Jörgen Dahlqvist

Sammanfattning
Abstract

When adding General Packet Service (GPRS) to the GSM infrastructure, the Base Station Controller (BSC) has to have a new part, the Packet Control Unit (PCU), to handle this new interface. The PCU is based on a new type of Regional Processor, the RPP. To test the software in the RPPs, a hybrid test environment was created. This hybrid test environment has the capacity to verify the software to some extent. To be able to test on a higher level and to increase the availability of the hybrid test environment, it had to be rebuilt.

This work aimed to rebuild the hybrid test environment and integrate/test the new parts needed. The new parts were the Central Processor simulator (SEA), the group switch (GS4M) and the traffic simulator (TSS2000).

The result of the SEA and GS4M integration was a “new” hybrid test environment that has an increased availability. This means that the RPP can be reached from the CP (SEA) and up to 7 users per RPP magazine can use the switching part at the same time and up to 16 RPP magazines can be connected to the switch. In the “old” hybrid test environment only one user could use the switching part.

The TSS2000 is needed in two versions, one software and one hardware based, and how to connect these two are left to be investigated. The hardware TSS2000 was not available in the end of this work. To be able to test on a higher protocol level and save many expensive hours in the AXE target environment, the integration of TSS2000 should proceed.

Nyckelord
Keywords

Abstract

When adding General Packet Service (GPRS) to the GSM infrastructure, the Base Station Controller (BSC) has to have a new part, the Packet Control Unit (PCU), to handle this new interface. The PCU is based on a new type of Regional Processor, the RPP. To test the software in the RPPs, a hybrid test environment was created. This hybrid test environment has the capacity to verify the software to some extent. To be able to test on a higher level and to increase the availability of the hybrid test environment, it had to be rebuilt.

This work aimed to rebuild the hybrid test environment and integrate/test the new parts needed. The new parts were the Central Processor simulator (SEA), the group switch (GS4M) and the traffic simulator (TSS2000).

The result of the SEA and GS4M integration was a “new” hybrid test environment that has an increased availability. This means that the RPP can be reached from the CP (SEA) and up to 7 users per RPP magazine can use the switching part at the same time and up to 16 RPP magazines can be connected to the switch. In the “old” hybrid test environment only one user could use the switching part.

The TSS2000 is needed in two versions, one software and one hardware based, and how to connect these two are left to be investigated. The hardware TSS2000 was not available in the end of this work. To be able to test on a higher protocol level and save many expensive hours in the AXE target environment, the integration of TSS2000 should proceed.

Foreword

This master thesis work has been carried out at Ericsson Radio System AB in Linköping, under the technical supervision of Jonas Waldeck and administrative supervision of Ann-Charlotte Linderson.

The examiner at Linköping Institute of Technology was Professor Lars Nielsen at the Department of Electrical Engineering.

Many thanks to all the helpful people at Ericsson who have supported me in my work. None named, none forgotten.

A special thank to my wife who have supported me through my whole education and taken care of our four kids while I has been, as I feel it, out of reach.

Linköping, July 2000
Jörgen Dahlqvist

Contents

Figures

Tables

1 Introduction

1.1 Background and purpose

An addition to the GSM infrastructure is the General Packet Radio Service (GPRS), which offers packet-oriented data communication. One example of packet-oriented data communication is the Internet. In the traffic interface between the GSM network and the GPRS network is the Base Station Controller (BSC). The BSC node has to be rebuilt, hardware and software, when the GPRS interface are to be added to the GSM network. The new part in the BSC node is the Packet Control Unit (PCU) that is the GPRS handling unit. The major part of the PCU is based on the so-called regional processors.
Function test (FT) of the regional processor software is performed in an AXE target environment, several local GSM/GPRS networks, as well as in a simulated environment.
One more easily accessible alternative to the AXE target environment is a hybrid test environment. In this test environment, the regional processor software can be verified, in real time, to some extent.

The purpose of this thesis work is to analyse and implement some new parts of the existing hybrid testing. The intention with implementation of the new parts is to get a better availability of the hybrid test environment and to get the possibility to perform function test at a higher protocol level.
That opens up the possibility of performing more tests in the hybrid test environment and thereby decreasing the number of expensive hours needed in the AXE target environment.
Since the hybrid test environment is more easily accessible, it should be possible to detect software faults in an earlier phase. This decrease the cost for fault corrections, as the later a fault is detected the more expensive it is to correct.

1.2 Objectives

The objective is to integrate the new parts in the hybrid test environment to get a better performance and availability in the “new” environment. In more detail, the objectives are as follows:

- Integrate a new program suite, called SEA, in the hybrid test environment and also give a short description on the program handling procedure.

- Integrate and describe the performance of the GS4M hardware in the "new" hybrid test environment.
- Analyse the GS4M GUI and propose improvements.
- Investigate the possibility to integrate TSS2000 in the hybrid test environment and give a proposal on how to perform the integration.

1.3 Literature survey

To get an understanding of the hybrid test environment, an extensive literature survey was performed at the Ericsson internal web and document stores. The literature survey resulted in a great amount of documents, where each of the documents describes a small part of interest. Some external references that give some help in understanding the basics in the concerned subject where found, [23, 24].

1.4 Disposition

Chapter 2: Base Station Controller gives a general description of the Base Station Controller with the internal parts, the Packet Control Unit and the Group Switching Subsystem, that are of interest from the hybrid test environment point of view.
Chapter 3: Software test and verification is a brief description of the different test and verification stages in the Packet Control Unit development process.
Chapter 4: Hybrid test environment is a short overview of the existing hybrid test environment and what changes are to made to get the desirable result.
Chapter 5: SEA and RP connection gives a description of the implementation of a program suite called SEA in the hybrid test environment.
Chapter 6: GS4M implementation and verification presents the implementation and verification of the hardware of a part called GS4M. It also gives a short description and analysis of the GS4M graphical user interface.
Chapter 7: TSS2000 is a description of that is needed to implement the TSS200 part in the hybrid test environment. And a proposal of how to make the implementation.
Chapter 8: Result and reflections presents the results of the changes of the hybrid test environment. It also gives some reflections of what has to be done to reach the desirable goals with the "new" hybrid test environment.

2 Base Station Controller

This chapter is a general overview of the BSC node and the interfaces towards the BTS node and the SGSN node. The interface towards the MSC node is not covered in this report. Thus, only the parts that are involved with the GPRS handling in the BSC node are mentioned.

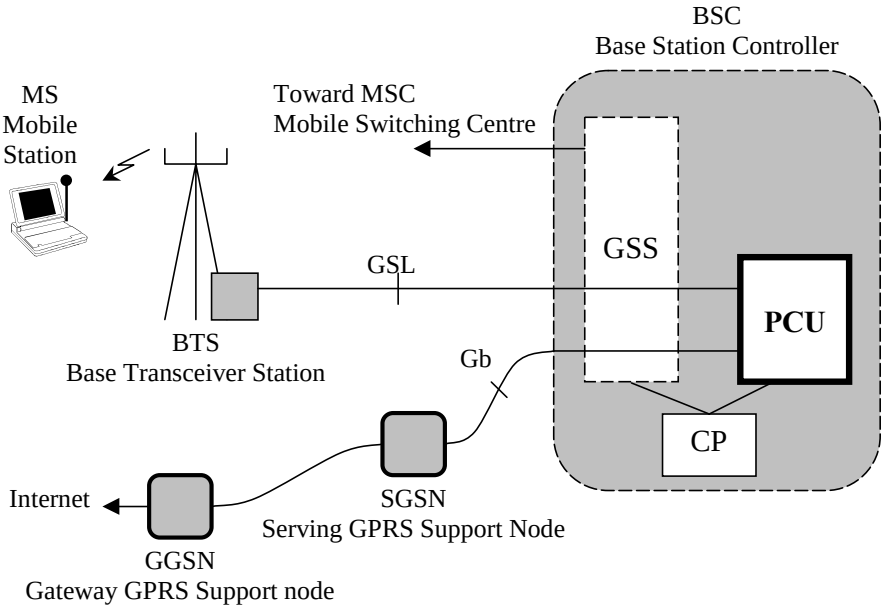

Figure 2.1 *Base station controller.*

The hardware and software structure of the BSC node is based on the telephone exchange system AXE10. The switching and telecommunication part in the BSC node consists of a source system called APT.

The APT implements the functionality of the BSC node including the functionality of the logical GPRS node PCU. The APT is controlled and supervised by a control system called APZ.

The control system APZ is based on a *Central Processor* (CP) that co-operates with *Regional Processors* (RPs) in PCU and the *Group Switch Subsystem* (GSS) connected through a bus called RP Bus (RPB).

The PCU connects to the Gb devices (SGSN) and to the GSL (Abis) device (BTS) via the GSS. The GPRS traffic is multiplexed with the circuit switch traffic in a Subrate Switch (SRS), included in the GSS.

2.1 Central Processor

The CP does not have a central part in the hybrid test environment, therefor this chapter is a very short presentation.

For reliability reasons the CP is duplicated, with the twin processors running in synchronism. They operate in separate states, *executive* (EX) and *standby/working (SB/WO)*. The CP in state *executive* controls the system. The *standby/working* CP takes control of the system instantly, in the event of a fault in the *executive* side.

The maintenance in the APZ, i.e. fault detection, recovery diagnostics and alarm generation, is handled by the Maintenance Subsystem (MAS).

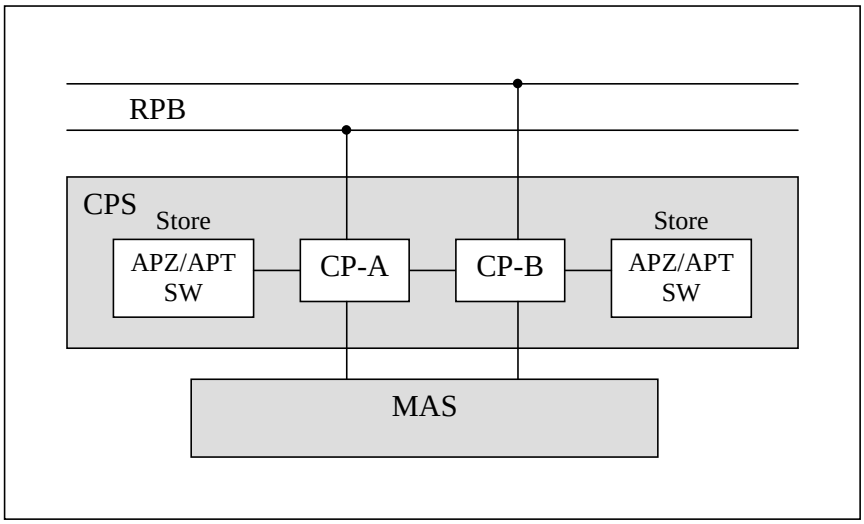

Figure 2.2 *Central Processor Subsystem (CPS) overview.*

All the data in the CP memory is backed up regularly, called *dumping*. The backup is called a *dump*. To load new APZ/APT software into the CP, a manually built *dump* is loaded into the store and then loaded into the CP memory.

The terms *reference dump* and *working dump* are used for *dumps*. The *reference dump* contains APT blocks and corrections. The *working dump* is a *reference dump* with *data transcript* (DT) loaded. The *data transcript* contains information about the configuration of all the different parts of the system and the connections between them.

2.2 Packet Control Unit

The PCU is the GPRS handling part in the BSC node. The major part of the PCU implementation is done in a new type of RP, the RPP (RP with internal PCI bus). The PCU consists of up to 14 RPPs, these RPPs are the GS interface part of the GPRS data transfer (see Figure 2.3). The function of the RPP is to distribute PCU frames to the Gb and the GSL (Abis) interfaces.

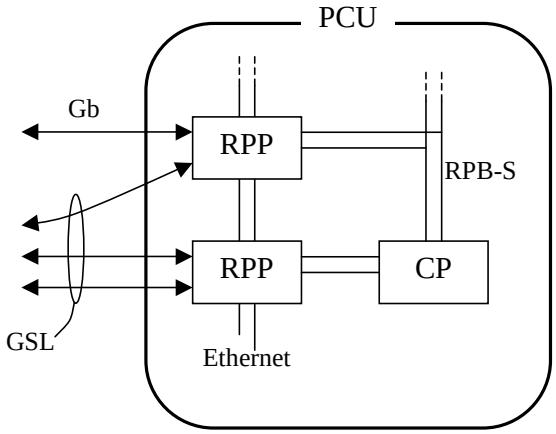

Figure 2.3 *Packet Control Unit*

The data transmission protocol stack in GPRS, except for the GGSN node, is shown in Figure 2.4.
The PCU is responsible for handling the BSSGP and Frame Relay layer of the Gb interface and Radio Link Control (RLC) and Medium Access Control (MAC) protocol layers on the GSL (Abis) interface. The PCU distributes PCU frames to the Gb interface and the GSL interface by segmentation/assembly of LLC frames into/from RLC/MAC radio blocks.

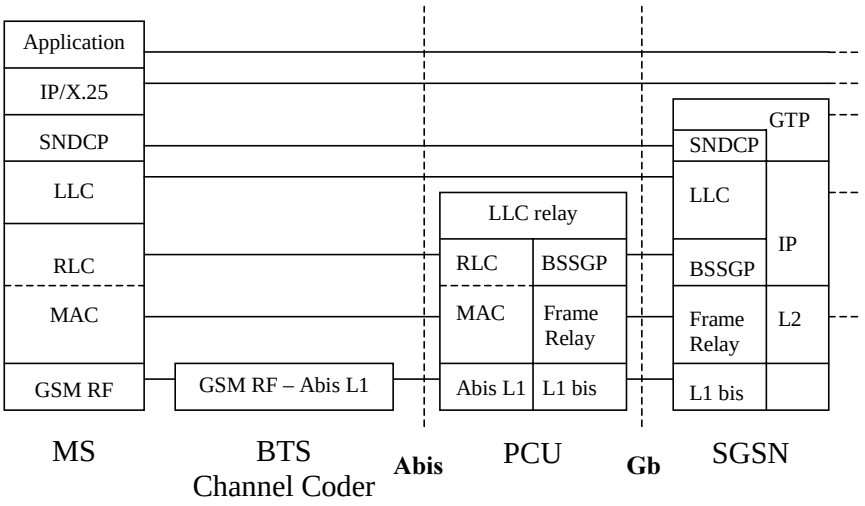

Figure 2.4 *GPRS data transmission protocol stacks.*

The communication in SGSN node to BTS node direction is called *downlink* and communication in the other direction *uplink*.

2.2.1 The RPP platform

General

The RPP provides the BSC node with both Packet Switched traffic and signalling between the SGSN node (Gb) and the BTS node (GSL) on physical connections switched through the *group switch*.
An RPP can work towards both the Gb and the GSL interfaces or towards GSL only. If there is only one RPP in the PCU it will work towards both Gb and GSL. If there are more than one RPP, each RPP may work towards either GSL or towards both Gb and GSL.

Each RPP consists of 2 boards (CPU board and I/O board) and is one half height plug-in unit that physically occupies the space of two plug-in units. The RPP is based on a Power PC hardware platform. It is housed in a GDDM-H with Ethernet connection in the back plane.

The RPP is equipped with two DL2 interfaces (toward Gb/GSL) and two RP bus interfaces (toward CP). It also contains DSPs to provide the DL2 links with the necessary I/O space. Only one DL2 is connected to each card but can be reached between the different cards on a RPP unit via the back plane. A number of RPP units can be interconnected to the Ethernet interface inside and even outside the magazine.

One RPP Central Processing Unit (CPU) can serve eight DSPs.
There are three DSPs dedicated to each DL2 for GSL interface handling and two DSPs handling the Gb interface on one of the DL2s, see Figure 2.5.
Each DL2 consists of 32 64 kbit/s GPH devices. Each GPH device can be configured for the GSL interface as 16 kbit/s PDCHs or for the Gb-interface as 64 kbit/s Gb devices. A number of Gb devices constitute one wide band physical link.

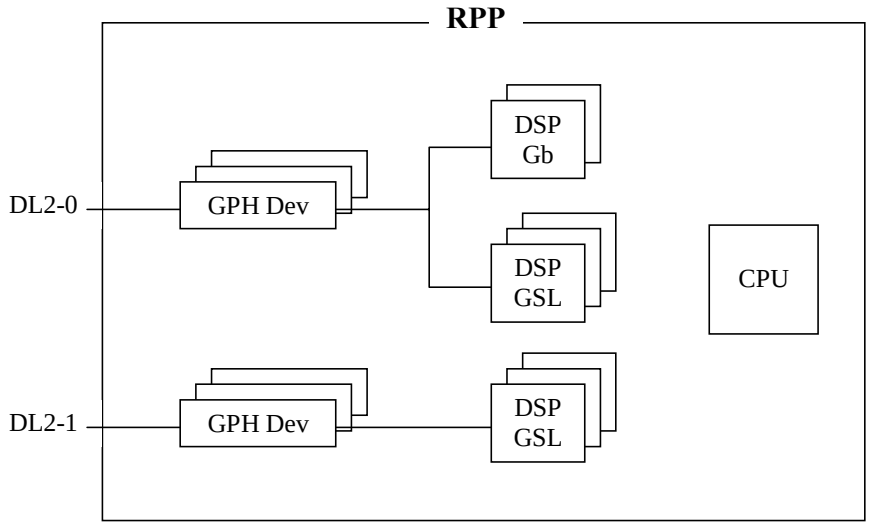

Figure 2.5 *RPP – logical connections between the DSPs and the DL2 links.*

2.3 Group Switch Subsystem

This section is a general overview of the Group Switch Subsystem (GSS) in the BSC node. It is by no means intended to be a complete description of the GSS.

The main functions of the Group Switching Subsystem (GSS) are to connect and disconnect a channel in one PCM system to a channel in another PCM system. In the case of calls using normal both way connections, two paths through the switch are set-up, one in each transmission direction. PCM links from other nodes are connected via Exchange Terminal Circuits (ETCs).

For reliability reasons, the entire switching network is duplicated into two separate synchronously working planes. Three CLock Modules (CLM) perform the timing in the group switch. Synchronisation is implemented

by means of clock reference sources and software algorithms containing a number of synchronisation methods.

Using a time-space-time architecture, the *group switch* (GS) performs switching within and between time-multiplexed buses.
The time switching is performed by the Time Switch Modules (TSMs) and the space switching by the Space Switch Modules (SPMs). For BSC hardware, according to the standard configurations, a Subrate Switch (SRS) is also included in the GS.

The maximum number of ports is 128k, which are all switchable. The BSC node is however only using maximum 64k ports.
The SRSM module is attached to the 64 kbit/s switch ports and thus occupies some of the switching network capacity.

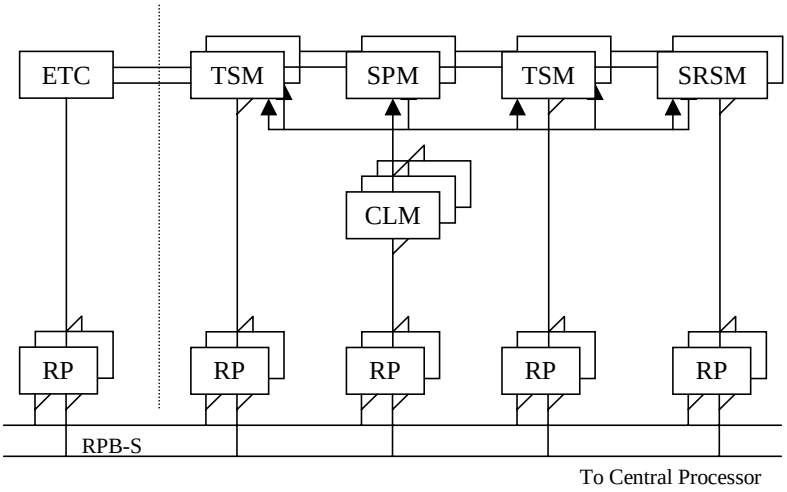

Figure 2.6 *GSS hardware architecture.*

3 Software test and verification

This chapter is a brief description of the different test and verification stages in the PCU development process. It is meant to give an insight about where in the development process the hybrid test environment can be used.

Before the test and verification steps are described a few concepts have to be explained.

- A *module* is one or several logically coherent RP software function(s).
- A *block* is one or several logically coherent CP software function(s).

Below follows a brief description of the different test and verification stages.

- Basic Test (BT) aims to test and verify the code within a *module* or a *block.*
- Process Unit Test (PUT) aims to test and verify the interfaces between *modules*.
- Basic Function Test (BFT) aims to test and verify the interfaces between *blocks* and the interfaces between *blocks* and *modules*.
- Function Test (FT) aims to test and verify the functional interfaces in a target environment.
- System Verification (SV) aims to test and verify the functional interfaces in a target environment from a customers point of view.

The hybrid test environment can be used in Basic Function Test and Function Test.

4 Hybrid test environment

4.1 General

In the existing hybrid test environment, see Figure 4.1, the software can be verified, in real time, to some extent. The hybrid test environment could be improved, better availability and to be able to test on a higher protocol level, if the connections that are not working could be implemented. Some components have to be replaced to make the hybrid test environment work as desired.

The HWemu and RPPsim are simulated equivalencies of the CP and RPP respectively.
RPX is a connection server between HWemu and RPPsim.
Musse is a very simplified equivalent of the GSS. It is a one-card hybrid with the connections through the hardware managed from a GUI on a UNIX workstation, thus not controlled from the HWemu (CP). It has two DL2 interfaces, i.e. it can connect to one RPP at a time.
TSS is a traffic generator that simulates real traffic cases, i.e. simulates the BTS node and SGSN node.

The HWemu, RPX, RPPsim, Adapter and the TSS are software products that run on a UNIX workstation, while the RPP is real hardware. The dashed lines means that the connections are desirable, but not implemented.
The RPPsim and Adapter are not handled in this report.

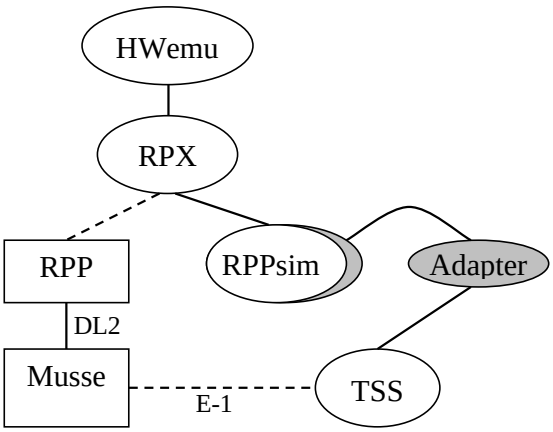

Figure 4.1 *Existing hybrid test environment.*

In Figure 4.2, the HWemu and the RPX have been replaced with SEA and the RP Connection Server (RPCS). The SEA program suite is a new CP emulator that in the future will replace the HWemu program. See more about SEA in chapter 4.2.
TSS2000 is hardware (ST) or software (SFT) based, and will replace the software based TSS that is used in the existing test environment. Musse is replaced by GS4M, which is described in chapter 4.4 and 6.

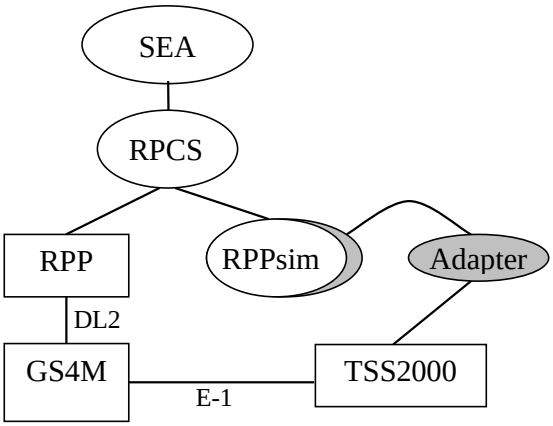

Figure 4.2 *Desirable hybrid test environment.*

The hardware in the desirable hybrid test environment, except the TSS2000, is called the *testrigg*. In the existing hybrid test environment the corresponding hardware is referred to as the *old testrigg*.

4.2 SEA

SEA is developed by Ericsson Infotech EIN/T. It consists of a number of modules that simulates the AXE hardware and software, in this case the BSC node. Some modules are executing the real BSC node software, while others are just simulating the signalling between the CP and the RP.

SEA is a scalable architecture for simulator based test environments and other simulator-based products. This architecture enables users to add their own components to implement new interfaces. SEA is a component oriented architecture using a component model based on the Microsoft Component Object Model, COM.

SEA consists of three layers: osCore, simCore and appCore.
osCore and simCore acts as the operating system in SEA. The appCore contains the simulated AXE components, like a CP part (CP 212 20) and RPPsim. The environment connected to SEA can be set-up dynamically,

there is no static linking between the components. For more information about SEA see [22] and chapter 5.

4.3 RPP magazine

This chapter is a description of the different parts housed in the RPP magazine.

The RPPs are housed in a GDDM-H that is a half height generic subrack for device boards. The standard configuration of the magazine consists of an RP pair (RP4), a DLHB pair, a EPSB pair and 14 slots for 7 RPP plug-in units.

In Figure 4.3 a front-view of the magazine can be seen. In position 0 there is an EMB board (RP4 in the *old testrigg*) and in position 19 an RP4 board. They have control over the addressing in the magazine and distribute the power supply to the backplane, and thus to all the other boards in the magazine (see Figure 4.4).

In the configuration with connection to Musse, there was also one board in position 1, which distributes two DL2 links from a front-connector to the backplane (not included in the figures).
In the configuration with connection to GS4M, there is one DLMUX board (DLHB), in position 1. There can also be one DLMUX board in position 18 for redundancy purpose, but that is not connected. The DLHB has one front-connector with a DL3 interface. It splits the DL3 into 16 DL2s and distributes them to the backplane. More about this in chapter 6.

As mentioned before the magazine has an Ethernet connection in the backplane. One Ethernet switchboard (EPSB) is mounted in the magazine to connect the backplane to the Ethernet. All the RPP units are reached from the Ethernet through that switchboard.

There are only five RPPs mounted in the magazine, which leaves four positions vacant. Those can be used for two RPPs or ETC boards that are needed when connecting TSS2000 (see chapter 4.5).

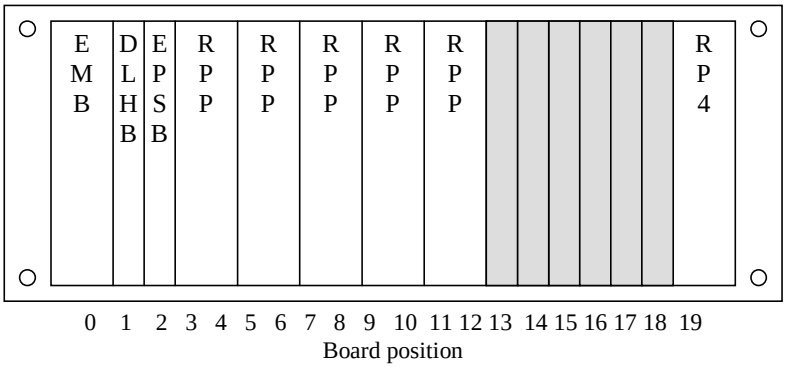

Figure 4.3 *RPP magazine front view.*

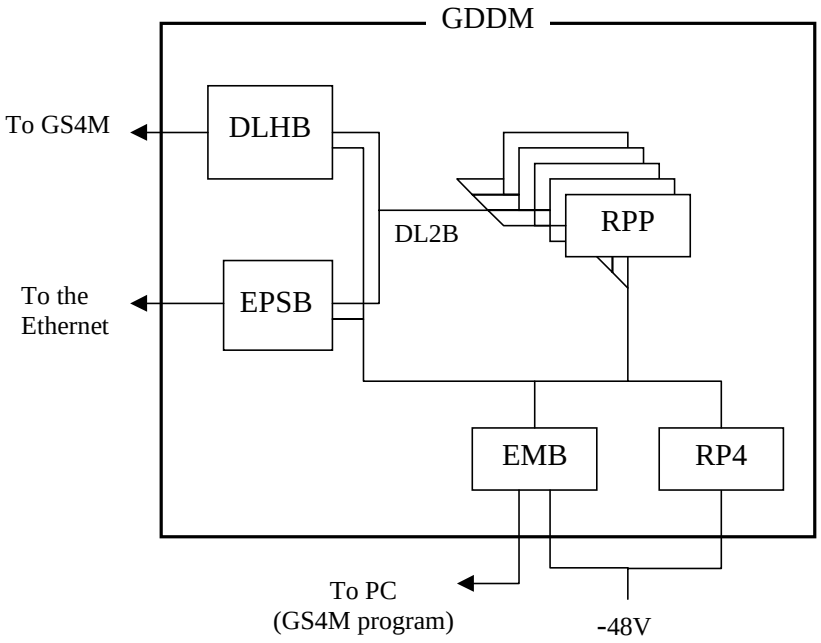

Figure 4.4 *RPP magazine structure with GS4M interface.*

In the original configuration of the *testrigg* RPP magazine the boards in position 0,1 and 2 should be in pair with the boards in position 19, 18 and 17 respectively. But since the *testrigg* has been used with Musse at the same time as the GS4M has been implemented the configuration is as mentioned above.

4.4 GS4M

4.4.1 General

The GS4M contains all the necessary functionality to be a complete switch. It consists of the switch core (SPMs and TSMs), three clock modules (CLMs), subrate switch (SRS) and all the RPs needed to control this (see Figure 4.5).
There are also clock references, one RCLB or two ICBs (not included in Figure 4.5). Those are not mentioned further because they are not included in the *testrigg* GS4M.

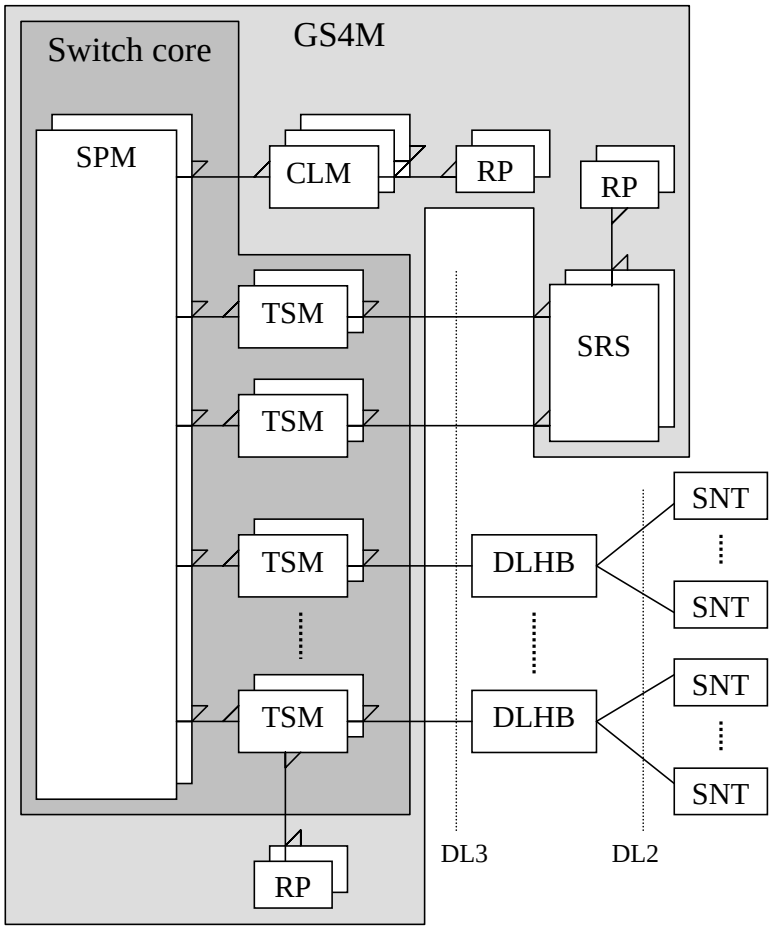

Figure 4.5 *Complete GS4M.*

The DLHBs are boards included in the RPP magazine.

The Switching Network Terminals (SNTs) are the common name for terminators in the AXE. In the *testrigg* a SNT can be seen as a DL2 in the RPP magazine, connected to a RPP or an ETC when connecting TSS2000 to GS4M.

The GS4M that is used in the *testrigg* is not a complete GS4M (see Figure 4.6). The clock part only consists of one clock module and all the RPs are present but not all are connected. The subrate switch is not a part in the *testrigg* GS4M because the purpose is not to handle circuit switch traffic.

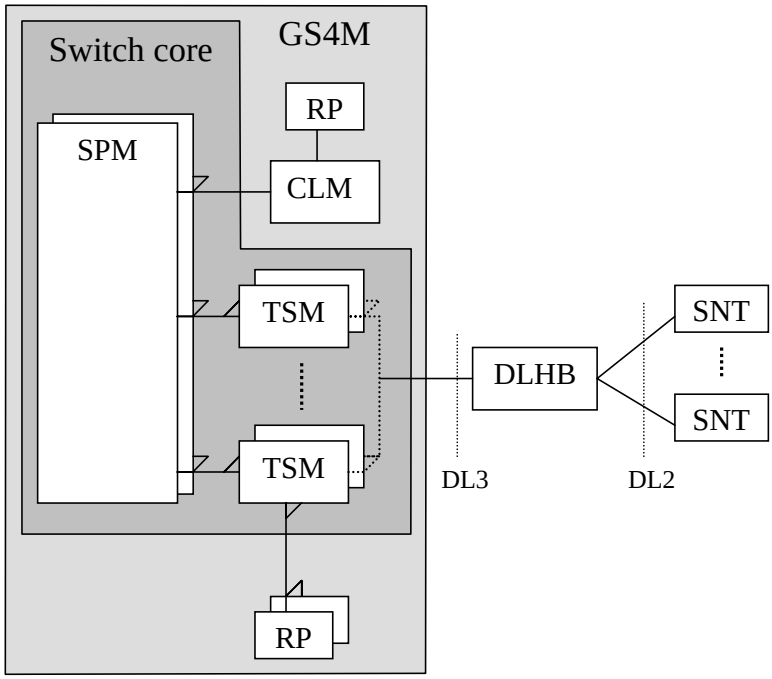

Figure 4.6 *GS4M in the new testrigg.*

The GS4M subrack is logically divided into left side and right side, plane A and plane B (see Figure 4.7). The dark grey marked boards are not included in the *testrigg* subrack and the pale grey marked boards are present but not connected.

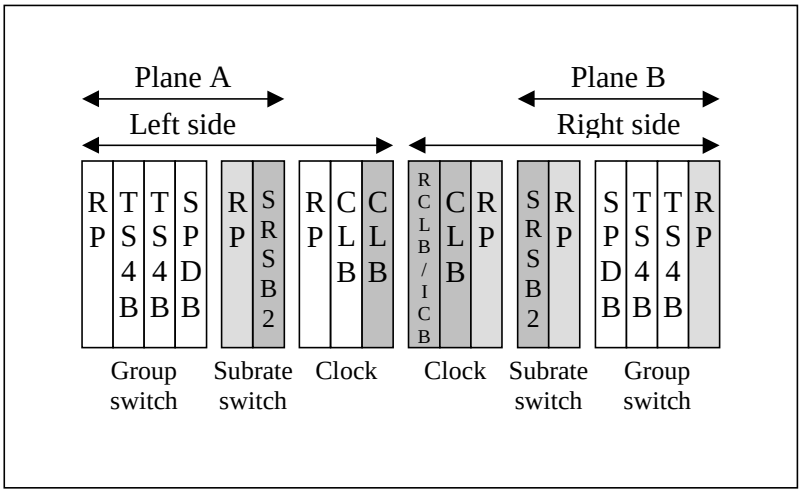

Figure 4.7 *GS4M subrack overview.*

In the original subrack plane A and plane B are working in parallel for redundancy purpose. The left most and right most RPs, called TSM-RPs, control the TS4Bs in each plane. If one of the TSM-RPs is faulty the other TSM-RP takes control over both planes.
As can be seen in the figure, the TSM-RP in plane B is not connected, thus the other TSM-RP has control over both planes. The controlling TSM-RP is connected to the TS4Bs in plane A through the backplane and the TS4Bs in plane B through two cables.

4.4.2SPM

In the GS4M there are two SPDBs and each contains one SPM.
One SPM has 32 inputs and 32 outputs, a so-called 32x32 matrix. Each output is implemented as a multiplexer. This means that every output can be connected to any of the inputs. At the two SPMs in GS4M are only 8 inputs and 8 outputs used. This means that the SPMs can connect to 2x8 TSMs together.

The SPDB normally takes in three clock signals and performs a majority vote on these. If a clock signal is dead, it does not participate in the majority vote. This means that one CLM is sufficient, as in the *testrigg*.

4.4.3 TSM

A TSM has one logical input and one logical output. A number of timeslots are time multiplexed on these. The function of a TSM is to change the order of the time multiplexed timeslots.
One TSM can handle 512 MUPs. In the GS4M there are four TS4Bs and each contains four TSMs. Thus, the TSMs can together handle 16x512 MUP.
Each TS4B has four DL3 front-connectors. So, without any redundancy, the GS4M should be capable of handling 16 RPP magazines. This has not been verified because there is only one RPP subrack present in the *testrigg*.

4.5 TSS2000

TSS 2000 is a product for test and simulation, which can be used for different purposes including traffic case test with user-defined traffic mixes. It makes it possible to simulate the BTS node and the SGSN node in the hybrid test environment. The physical capacity limit for TSS 2000 is 64 PCM systems and 256 time slots, more than enough. The simulation is controlled by test programs, which are a number of instructions written in a high-level language. The TSS 2000 user interface is executed on a UNIX workstation.
TSS 2000 provides built-in communication protocols such as Signalling System No 7, LAPD, and STC-STR. The Signalling System No 7 and LAPD protocols are needed to implement TSS2000 in the hybrid test environment.

The connections toward GS4M will be made via ETC boards that can be mounted in the RPP subrack. This means that the hardware based TSS2000 has to be used.
More about this in chapter 7.

5 SEA and RP connection

This section is a description on the work around the connection between SEA and the RP hardware. There are several ways to establish the connection but I'm just describing the way that I used.

Figure 5.1, is a very rough overview of the SEA components, i.e. only the components that are of interest for the connection between SEA and RP hardware are added.
The CP21220 component, which is loaded with the *dump*, simulating the CP in the PCU and are connected to the RPBH-S. The RPBH-S is a component that controls the serial RP-bus (RPB-S).
The dashed RP component is created by the SEA Configuration Wizard (see chapter 5.1.1) but has to be replaced by the RPProxy to be able to connect to the external RP Connection Server (RP CS). The Connection Server is a small program that communicates with both the RPProxy and the RP hardware.

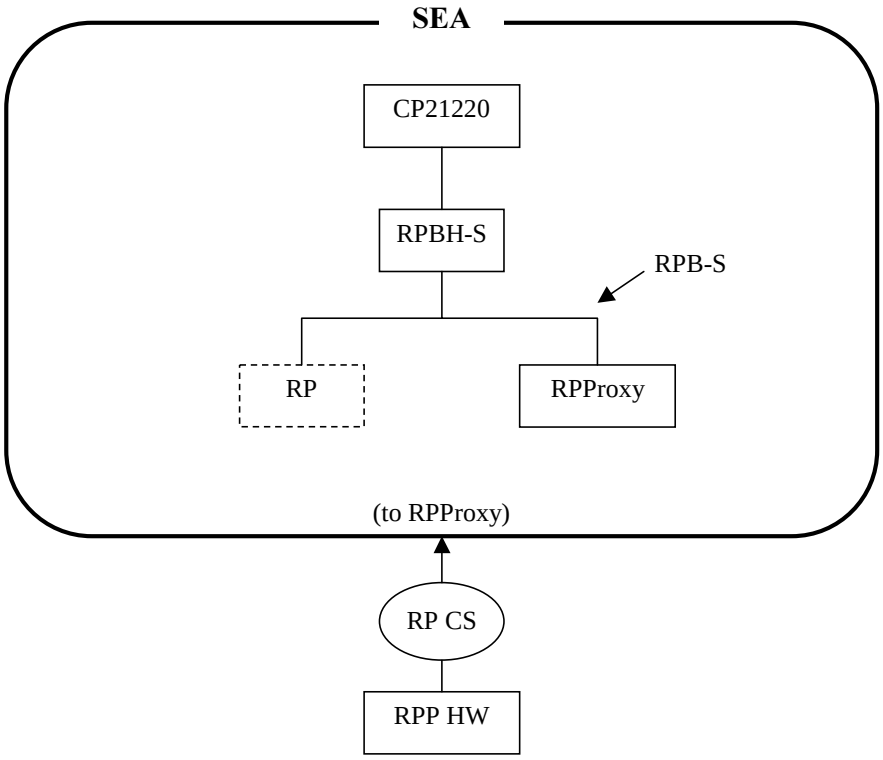

Figure 5.1 *SEA components.*

5.1 The configuration file

The configuration file (appendix 11.1) contains information about the components defined on the *dump* and how they are configured and connected to each other. The SEA Configuration Wizard creates this configuration file from the *dump*.
The first thing to do is to create the configuration file. The next step is to alter the configuration file and tell SEA that, instead of using an RPP simulation, you want to use real RP hardware i.e. replacing the RP with RPProxy, se Figure 5.1.
There are a few steps to take to get the hybrid environment up and running:

- Create the configuration file.
- Alter the configuration file to add the RP proxy.
- Start the OSE system daemon and the link handler.
- Start SEA with the new configuration file.

5.1.1 Creating the configuration file

The SEA Configuration Wizard (se Figure 5.2) is used to create the configuration file. The *wizard* can be started from the "Tools" menu in the SEA Control Center or from a shell with the *seaconfig* command. Below follows a short description of the procedure used to create the configuration file.

To create the configuration file, in the configuration wizard, is a straightforward procedure. Besides the two fields under the "Dump" tab, "CP dump" and "AXE Manager Configuration File", there are a choice of "CP Model" under the "CP" tab that has to be done, in this case 21225. After this configuration it is time to launch the wizard, by clicking the "GO" button.

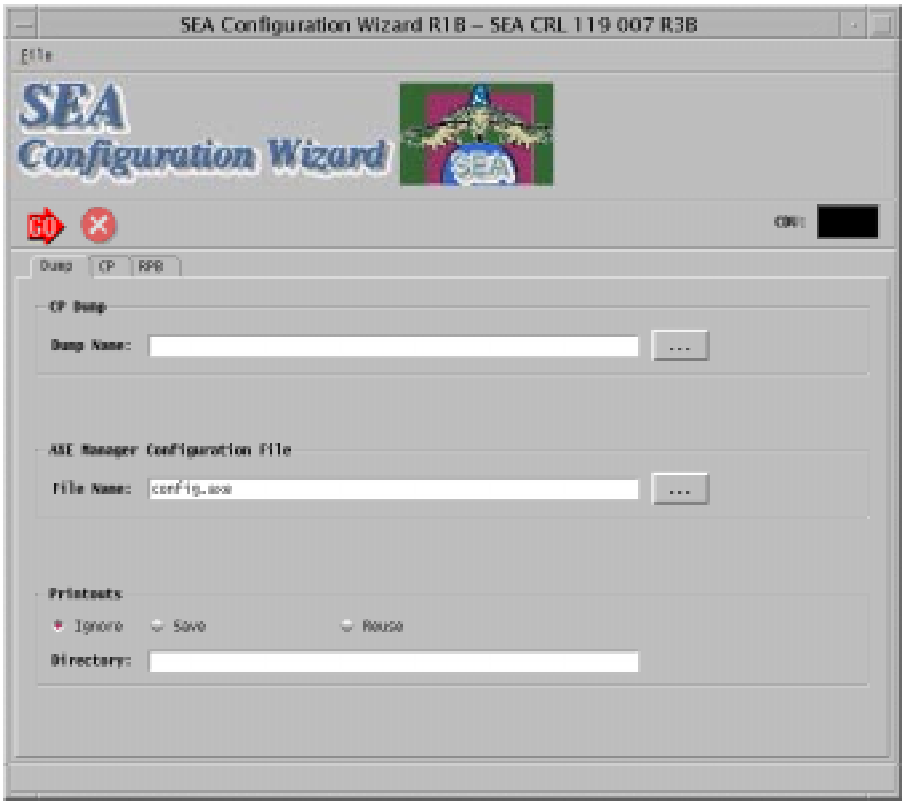

Figure 5.2 *SEA Configuration Wizard.*

While the *wizard* is launching, it generates questions like that in Figure 5.3. In this case, the “EMRPSIM.EMRPD1.1” part should be selected.

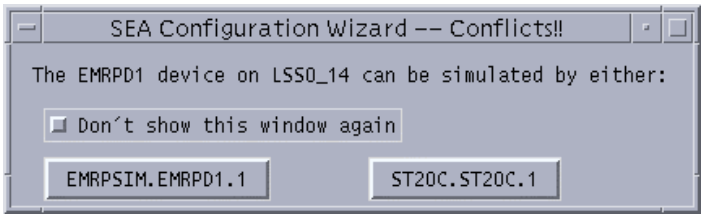

Figure 5.3 *SEA Configuration Wizard –Conflicts.*

When the *wizard* is done, a configuration file is created with a path and name defined in the “AXE Manager Configuration File” field.

5.1.2Altering the Configuration File

As stated earlier, it is necessary to alter the configuration file generated by the SEA Configuration Wizard. A configuration file consists of *create*,

connect and *config* statements. The complete file can be seen in appendix 11.1, but selected parts of the file is viewed below:

```
# Create the CP
create CP21220.CP21220.1            CP
# Create the RPBH
create RPBH.RPBHS.1                 RPBH_3
# Create the RP
create RPSIM.RPPS1.1                RP_98
# Connect RPBH to CP
connect RPBH_3          CP          IrphbServer         0
# Connect RP to RPBH
connect RP_98           RPBH_3      IRpbSServer         30
```

There may also be RP software simulations attached to the simulated RP that are to be exchanged by an RP proxy. The *create* and *connect* statements for these RP software simulations could be removed from the configuration file, since there is no reason to connect RP software simulations to an RP proxy.

To let RP position number 98 be handled by real RP hardware, it is necessary to change the statements above to this:

```
# Create the CP
create CP21220.CP21220.1            CP
# Create the RPBH
create RPBH.RPBHS.1                 RPBH_3
# Create the RPPROXY
create RPPROXY.RPPROXYS.1           RPP_HW
# Connect RPBH to CP
connect RPBH_3          CP          IrphbServer         0
# Connect RP to RPBH
connect RPP_HW          RPBH_3      IRpbSServer         { 98 RPPHW }
```

Note that the *connect* statement between the RP and the RPBH (RPP_HW and the RPBH_3) now contains a second argument. That argument is used to set the name of the RP proxy. This is the name the connection server uses to connect to the RP proxy component.
The complete altered configuration file can be seen in appendix 11.2.

5.1.3 The external connection

In order to get the connection server (RPCS) to work it is necessary to have the OSE system daemon (osesysd) running on the workstation. Only

one instance of this daemon should be running on the workstation at one time.
Secondly, it is necessary to have a link handler running on the workstation. If the RP board is connected to the workstation on the serial port, the link handler to use is “lapimhp”.
In this case, to connect to the RP via the Ethernet, the link handler “lnhethc” is used.

The connection server could ether be started from the command line, from the home page of the RP proxy component or from within the configuration file. In this case it is started from the configuration file. By adding a *config* statement to the configuration file, the connection server will automatically start the next time SEA is launched.
The *config* statement looks like this:

```
config RPP_HW {
rpproxy_setstate -bl 1
rpproxy_startcs -trace -hp "rp1111/CP_sim"
}
```

5.2 Connection test

The only programs that are started in the RP on power on is the operating system (OSE Delta) and the system program, the API named Razor. The next steps to get the RP running is to load, if it is not already loaded, and start the application programs. The load/start is done by the CP and that procedure is called *deblock*.

The tests in this chapter only consider the connection between SEA and the RP hardware. The test case was to check if it was possible to get a stable connection between SEA and the RP hardware. *Deblocking* and *blocking*, i.e. start and stop the application software in the RP, are a critical part of the communication between SEA and RP, so if that works the connection test case is successful.

When the tests started, the latest version of SEA was built for the APZ P6 version of the control system. The operating system in the RP did not support the APZ P6 so it had to be updated. As a consequence of that, the *dump* had to be rebuilt, i.e. it was not built for the new software in the RP. The rebuilding of the *dump* is described in chapter 5.3.

The testing started with SEA version R3A but the *deblocking* failed because of lack of support in SEA for the RP type to connect to. There came one patch (SP1) for the R3A version and a new version R3B. But it

was after frequent contact with the SEA developers and testing that patch number one and two (SP1/SP2) to version R3B came. This made the *deblocking* and *blocking* successful, i.e. at least version R3B with patch number two is needed.

If the loading part is performed when *deblocking*, it fails in the first when the *transmit timer* in the link handler time-out the load transmission. The *deblocking* then has to be performed by sending the *deblock* command a few times. In total the *deblocking* with loading from SEA takes about 10-15 minutes.
As a comparison, the *deblocking* can also be performed with a program called *rpload*, the loading with that takes about 10-15 seconds without any timing problem.
So, the problem with loading from SEA should be solveable, by the developers, by increasing the load speed.

To start the testing on SEA, the SEA Control Center is started with the "sea" command in the same shell as that the OSE system daemon and the link handler is started.

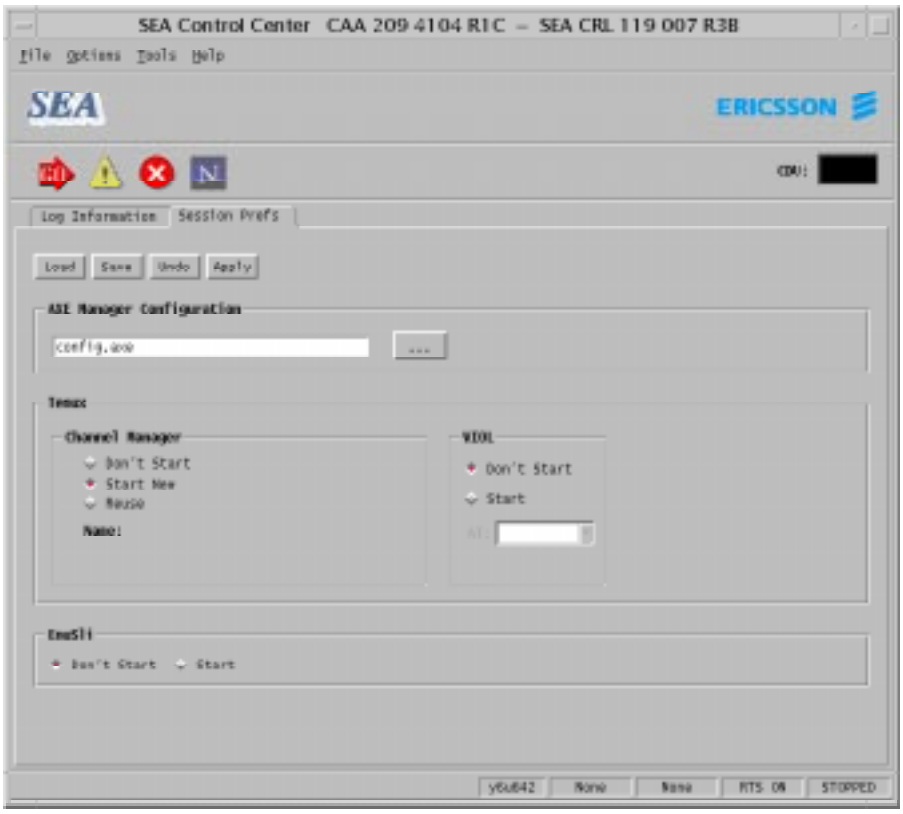

Figure 5.4 *SEA Control Center.*

To *deblock* a real RP, it is necessary to turn on the Real Time Simulation (RTS), under the "Options" menu. This is necessary, because the CP components are too quick for the RP hardware interface, i.e. the RP will be timed out and the *deblocking* will fail.

Choose the "Session Prefs" tab, write the path and name of the configuration file in the "AXE Manager Configuration" field and press "Apply". Choose the "Log Information" tab and launch the *dump* by pressing the "GO" button.
A typical log is shown below:

```
#------------------------------------------------------------------------
# SEA Control Center Log File
# User: ejorgen
# Date: Wed Mar  8 09:07:01 MET 2000
#------------------------------------------------------------------------

CHAMS started. (pid: 10393)
Info: SEA is started (pid: 10395)
HTTP server started on port 5000
/layer/appCore/273c0e38-5787-11d2-bf6a-0800208818ed >Loading
'workingdump.21220_emudump.gz' dump
Allocating memory: DRS=256 MW32, PS=32 MW32, PSCM=512 KW32
Loading RS ...
Loading PS ...
Loading PS cache ...
Loading DS ...
Loading 'workingdump.21220_emudump.gz' dump completed
/ >MPH started on port 38167
/layer/appCore/273c0e38-5787-11d2-bf6a-0800208818ed
>/layer/simCore/13fa39cc-6031-11d3-827a-08002082b68e
>/layer/simCore/13fa39cc-6031-11d3-827a-08002082b68e
>/layer/appCore/273c0e38-5787-11d2-bf6a-0800208818ed
>/layer/appCore/273c0e38-5787-11d2-bf6a-0800208818ed
>/layer/appCore/273c0e38-5787-11d2-bf6a-0800208818ed >
RPPHW: searpcs_rpbs: No connection with RP! Trying to (re)connect...
 searpcs_rpbs: Connection with RP established!
RPPHW: searpcs_rpbs: Connected to RPPHW proxy on channel 0
```

Note the two last rows, that shows that the connection server has established connection both to the RP hardware and the RP proxy within SEA, i.e. the Connection Server connection is successful.
The next step is to open a Wiol console, to be able to send the *deblock* and *block* commands to the CP. This console is opened from the “Tools” menu.

In Figure 5.5, a view of a successful *deblock* from a Wiol console is shown. The response of the “exrpp:rp=98;” command is a view of the RP state. In the first “exrpp:rp=98;” call the RP is *manual blocked* (MB), as stated in the *dump*, and after the *deblock* command (blrpe:rp=98;) the RP is in working (WO) state. If the *deblocking* fails the RP will be *automatic blocked* (AB), i.e. the CP will *block* the RP.

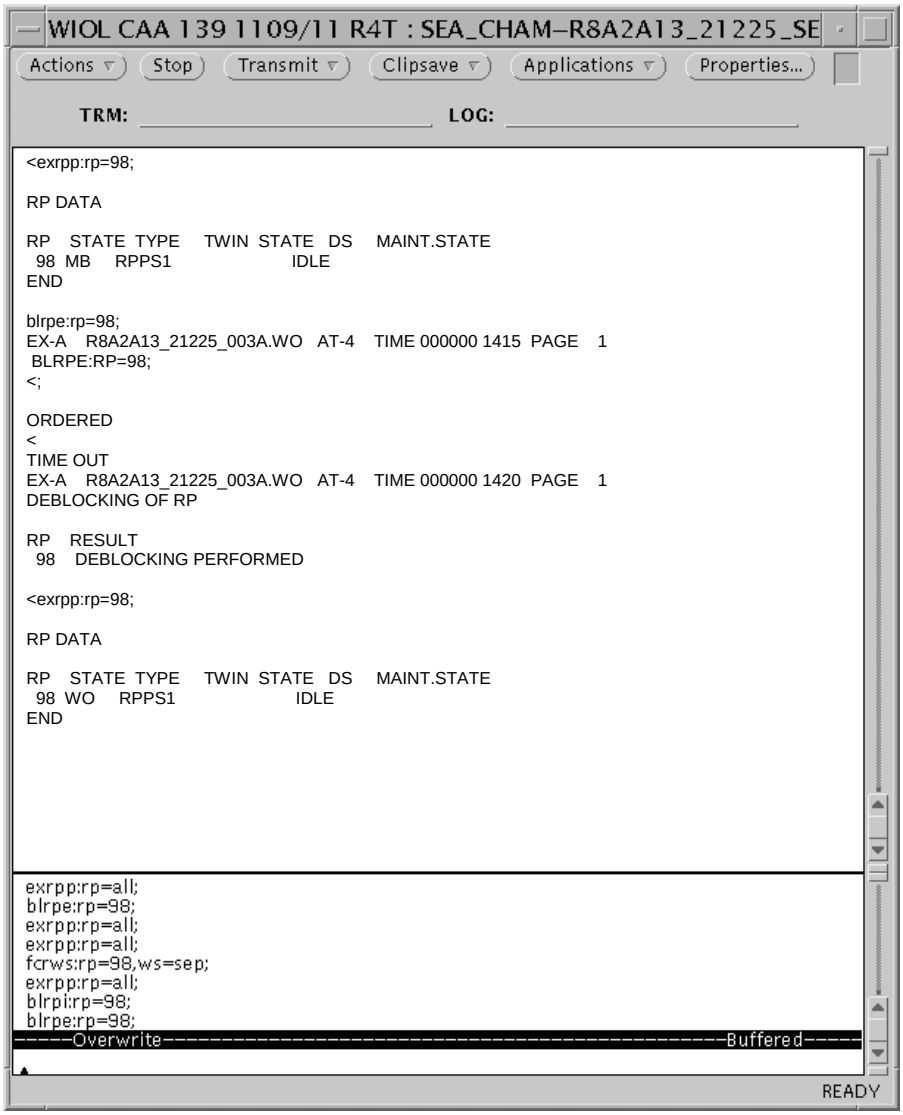

Figure 5.5 *Wiol Console*

5.3 Dump rebuilding

The purpose with the *dump* rebuilding is to change a Software Unit (SU), a so called Function Change (FC), in the *dump* (RP). This unit, named “RPEXR”, is the OS unit connected to the RP. This chapter describes a general rebuilding procedure from a RPP perspective.

To start with, the dump to rebuild (R8A2A13_21225_003a.wo.tar.Z) has to be loaded into SEA. The commands needed are entered in a Wiol

console (MML commands) and the *Command Field* in *SEA Control Center.*

SEA simulates one of the CP sides, so first that CP side has to be configured with following MML commands:

SYATI;

This command is used to start passive blocks in the SB side of CP.

PTSWI;

This command is used to change CP side. The CP side that was SB will become EX and the CP side that was EX will become SB.

To change one software unit, all the logical connections to it have to be removed. Some units in the RPP have logical connection to so called Extension Modules (EMs). The EMs in the RPP are numbered 0-3 and have the purpose to hold the address of the unit. The address is used by the corresponding block in the CP to recognise the RP unit.

The “RPEXR” unit is not connected to an EM, so the commands bellow that handles EMs are not used when changing that unit.

FCRWS: RP=rp, WS=SEP;

The command changes the state of the RP to *separated.*
When the state of the RP is changed to *separated* (parameter SEP), the RP becomes accessible only to a *separated* SB side of the CP.
In CP-EX the RP is interpreted as *blocked.*
The controlled equipment (EM) is *blocked* in CP-EX.

BLRPI: RP=rp;

This command is used when *blocking* an RP, in other words the logical state of the RP is set to *manually blocked* (MB).
All EMs linked to the RP must be *blocked* in CP-EX before the command can be accepted.

BLEMI: RP=rp, EM=em;

This command is used when *blocking* an EM that is controlled by a RP. *Blocking* means that the supervision of the EM is stopped and the recall of the associated programs in the RP ceases.

EXEME: RP=rp, EM=em;

The command is used in order to remove the definition of an EM.

EXRUE: RP=rp, SUID=suid (SUNAME=suname);

Command “EXRUE” is used to delete defined software units in an RP.

The specified RP must be *manually blocked*. Parameter SUID must be specified if more than one version of the specified software unit is defined.

When the connections have been removed, the new software unit has to be loaded into the CP memory. The directory (called IOG directory) path to the unit has to be defined in the *Command Field* in *SEA Control Center* with the following commands:

.cd ~CP (To communicate with the CP in SEA.)
set-iog-directory <IOG directory>

The directory has to have a specific structure, as in Figure 5.6.

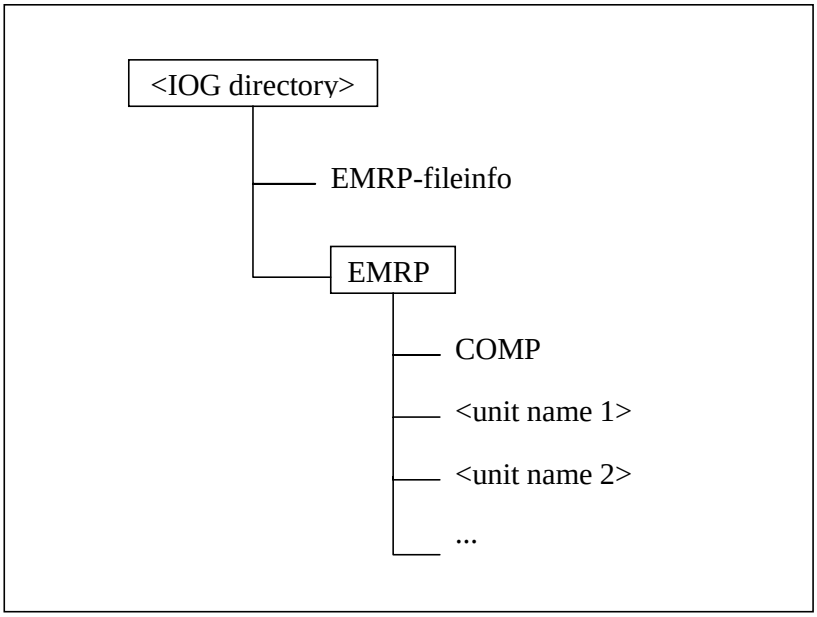

Figure 5.6 *IOG directory structure.*

The “EMRP-fileinfo” file will have the following contents.

rlength 512
type seq
fclass cmp

The “EMRP-fileinfo” file and the “EMRP” directory can be created with a MML command.

INFII: FILE=EMRP, RLENGTH=512, FCLASS=CMP, TYPE=SEQ;

The “COMP” file consists of the names (SUNAMEs) and identities (SUIDs) of the unit files that are placed in the “EMRP” directory.

As mentioned before, the unit has to be loaded into the CP memory. This is made with the following MML command.

LAEUL:SUID=suid (ALL);

If “ALL” is specified, all software units in the “EMRP” directory are loaded.

When the new software unit is loaded, the logical connections have to be restored.

EXRUI: RP=rp, SUID=suid (SUNAME=suname);

Command EXRUI is used when defining a specified SU for an RP. Parameter SUID must always be used if more than one version of the SU is defined in the CP. The specified RP must be *manually blocked.*

EXEMI: SUID=suid (SUNAME=suname), EQM=eqm, RP=rp, EM=em;

This command is used for defining an EM and for linking equipment to a software unit.
The command may be given to both *blocked* and *deblocked* RPs.
Parameter SUID must always be used if more than one version of the software unit is defined in CP.
The RP, which is specified in the parameter RP, is set as the home address for the EM.
The EM is set to the state *manually blocked.*

BLEME: RP=rp, EM=em;

This command is used when *deblocking* an EM that is controlled by a RP.
Deblocking means that the supervision of the EM is started and the recall of the associated programs in the RP is activated.

When all the connections are made, the *dump* is saved with the following command in the *Command Field* in *SEA Control Center.*

save-dump <dump name>

The dump is placed in the “IOG” directory.

6 GS4M implementation and verification

As mentioned before the GS4M is replacing the Musse board in the *testrigg*. The GS4M used is a modified variant, see chapter 4.4.
There where three main goals in the evaluation of the GS4M and they where:

- Investigate if the GUI is competent, otherwise which new demands are needed.
- Integration of the GS4M hardware in the *testrigg*.
- Verify that DSP joint test is possible with the GS4M.

6.1 GS4M graphical user interface

After inquiry by the developers (UAB) and search for the documentation that should belong to the GUI, I had to give up - there is no documentation on the GUI. So, the primary demand is that documentation has to be created. However, some parts of the GUI were described in another document, [21], written about another test case. With help from that, some work could be done with the GUI. But only the parts that were necessary to go on with the implementation of the GS4M are mentioned.

6.1.1Investigation

Make connections

Figure 6.1 shows the main window of the GUI and there are only a few parts that will be used.
The cross squares in field (2) have to be marked to tell the switch that an external GDM magazine is connected.
Field (1) is where the internal connections through the switch are configured. The “Source/Destination” rows are where the choice of which EM (DL3), Link (DL2) and TimeSlot to connect from/to is made. More about the external EM/Link (DL3/DL2) front-connections in chapter 6.2. In this case only one timeslot in each direction can be selected at a time, i.e. one timeslot to another. To make a connection in both directions, the connection has to be made in two steps. An example is shown below.

Example 6.1:

The x/x/x stands for EM/Link/TimeSlot.
Connect 0/1/2 with 3/4/5 in both directions.

Step 1: Fill out source field: EM=0, Link=1, TimeSlot=2.
Fill out destination field: EM=3, Link=4, TimeSlot=5.

Click the Connect button.

Step 2: Fill out source field: EM=3, Link=4, TimeSlot=5.
Fill out destination field: EM=0, Link=1, TimeSlot=2.
Click the Connect button.

///

To reset a two-way connection, the reset also has to be done in two steps. An example is shown below.

Example 6.2:

The x/x/x stands for EM/Link/TimeSlot.

Step 1: Reset link 0/1/2.
Fill out source field: EM=0, Link=1, TimeSlot=2.
Fill out destination field: EM=0, Link=1, TimeSlot=2.
Click the Connect button.

Step 2: Reset link 3/4/5.
Fill out source field: EM=3, Link=4, TimeSlot=5.
Fill out destination field: EM=3, Link=4, TimeSlot=5.
Click the Connect button.

///

To reset all the connections, the “Init All” button is selected.

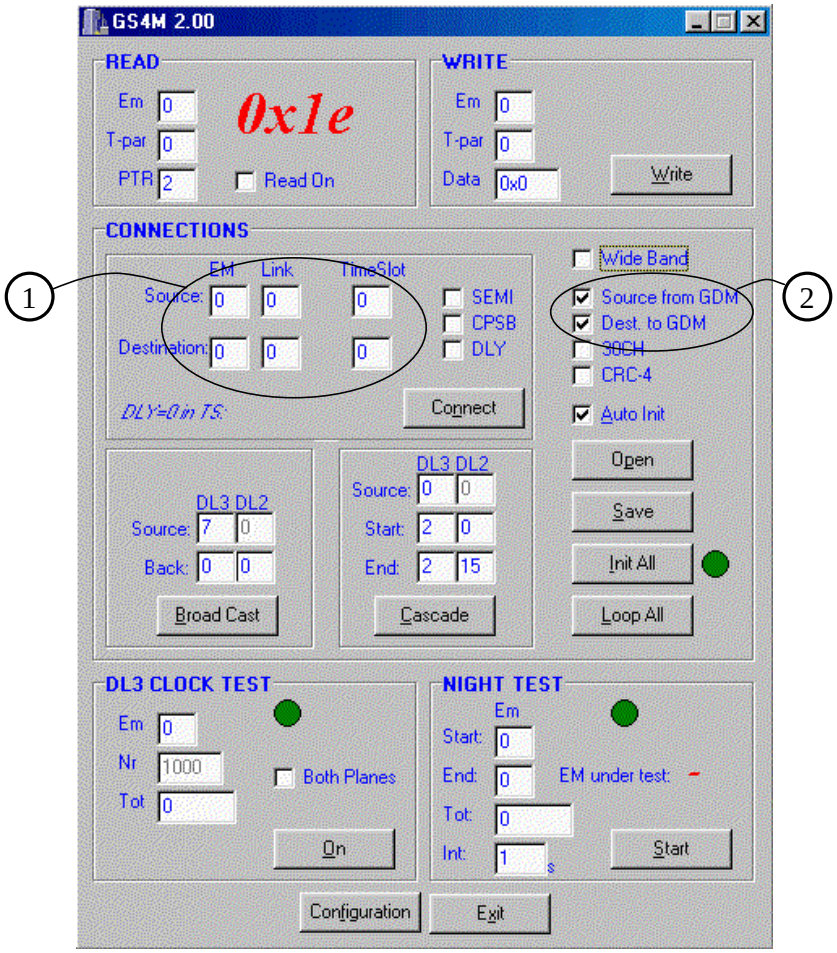

Figure 6.1 *GS4M graphical user interface – main window.*

In Figure 6.2 the “Wide Band” cross square is marked and that makes it possible to connect two ranges of timeslots to each other at a time. In this case, a connection/disconnection in both directions is done in the same way as for a single timeslot, but with a range of timeslots.

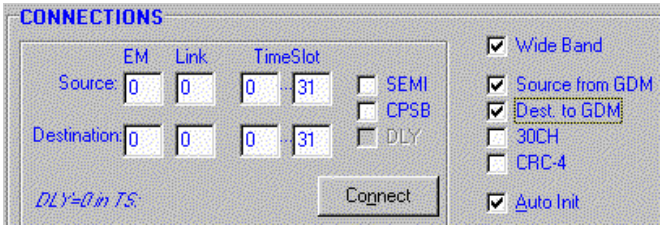

Figure 6.2 *GS4M graphical user interface – wide band.*

Logging

All the connections are logged in two *.log files, “current.log” and “previous.log”. These files are just logging the connections made in the program and are not connected to the hardware. This means that all the information about the connections in the hardware relies on these files. This could be a problem.
As an example, the files could be damaged if there are problems with the host that the program runs on and that means that all the information could be lost.
Another problem is that old information in the log files is not updated, i.e. if one connection on a timeslot is changed the old information will remain in the log. Admittedly, all the necessary information is available, but when the files grows the information becomes immense. This could lead to problems if there are several users of the system, and that is likely. Thus, the log information should be based on the hardware directly and a graphical representation should be present because of the problem to get an overview of the connections.

Card configuration and alarms

When the Configuration button in the main window is selected, the Configuration window appears, Figure 6.3. This window shows the cards that are installed in the GS4M magazine and alarms on the different front-connectors. In this case, it is the marked front-connectors (field (1-3)) that are connected to other front-connectors, in the same or other magazine.

An alarm on a front-connector is indicated in red, green means no alarms. In Figure 6.3 are the field (1) connectors indicating green, the rest are red, except for the connectors in field (3) that have no colour indication. The “EMB C” (field (2)) alarm is ok, it is just indicating that the CLB card installation is not what is expected, only one CLB card instead of three, as in the original GS4M magazine.

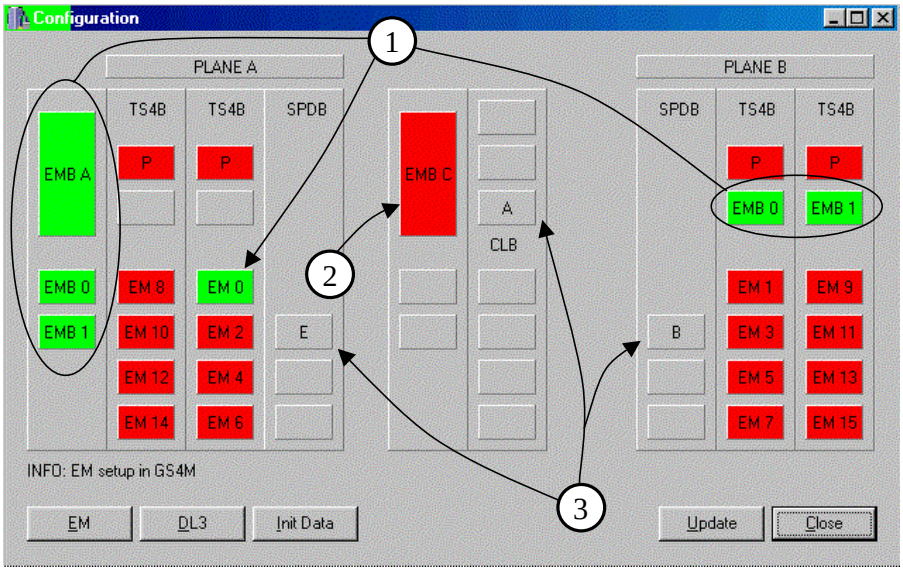

Figure 6.3 *GS4M graphical user interface – configuration window*

If the “EM 0” in field (1) is selected an “Alarms” window will appear, Figure 6.4. As can be seen, the “EM 0” front-connector is not indicating any alarms but the “Alarms” window shows an alarm on bit “CL_L”. This alarm is ok, it is just indicating that the clock configuration is not what is expected. But the indication bug should be corrected.

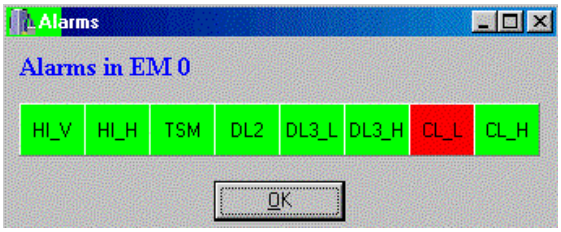

Figure 6.4 *GS4M graphical user interface – alarms in EM.*

6.2 Hardware integration

6.2.1 External connections

In Figure 6.5 the external cable connections of GS4M are shown, the internal cable connections are shown in Figure 6.6.
The GS4M GUI is running on a PC with Microsoft Windows 95/98. The communication between the GUI and the *testrigg* is performed via internal PC cards (PCEM).

The EMBA cards are the same as the RPs mentioned in chapter 4.4. The TS4B card has four DL3 connections (only one connected) to communicate with the RPP, but the communication is performed via a DLMUX (DLHB) that splits the DL3 link into sixteen DL2 links and distributes them to the back plane.
In the figure, the EPSB card that connects the back plane on the RPP magazine with the Ethernet can also be seen.

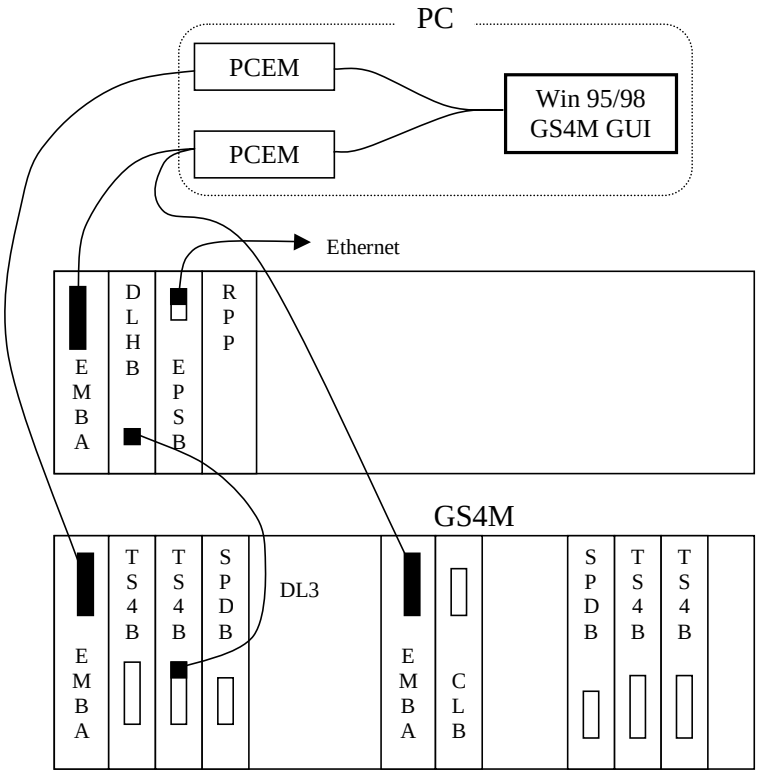

Figure 6.5 *External connections of GS4M.*

6.2.2Internal connections

As can be seen in Figure 6.6 only the leftmost EMBA card is powered with –48V. That card distributes the power to the back plane, i.e. to all the other cards in the magazine.
The clock from the clock card (CLB) is distributed to both the SPDBs. The GS4M magazine installed in the *testrigg* is only equipped with one clock board (CLB) because the reliability demands are not as rigorous as in a real AXE exchange.

As mentioned in chapter 4.4, the TSM-RP can handle two TS4Bs, but if one of the RPs fails the other RP will take control of all the four TS4Bs. In the current rack the leftmost EMBA (TSM-RP) is the only TSM-EMBA, thus that EMBA has control of all four TS4Bs.

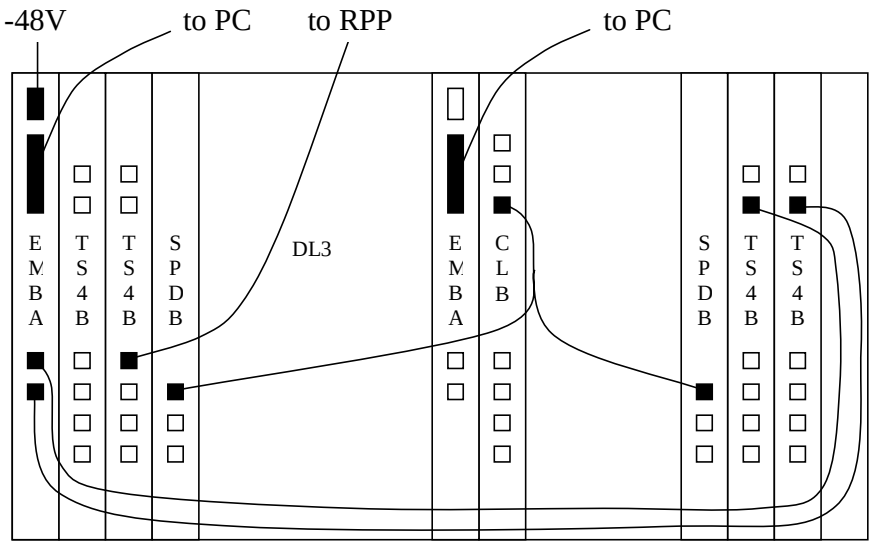

Figure 6.6 *Internal connections of GS4M.*

6.3 DSP joint test

There are a few programs needed to make a DSP joint test, and they are the slotlogtool, Distributed Debug Server (DDS) and GS4M programs. The slotlogtool and DDS are described later in this chapter.

6.3.1Description

The DSP joint test is performed to verify that the synchronisation, when a connection is estalished, on the GSL interface between the PCU (BSC) and the BTS node is working properly.

Figure 6.7 is an overview of the logical connections needed to perform the joint test. The sending DSP is named DSP 3 and the receiving DSP is named DSP 2.
The test case is that one DSP (DSP 3) in the RPP “simulates” the BTS node and sends synchronisation frames (a special sets of data, not explained further, see appendix 11.3) through the GS4M and back to another DSP (DSP 2). In the receiving DSP the GSL interface has to be activated. The receiving DSP tries to synchronise on the received frames

and if the synchronisation is successful, the DSP responds back to the sender on the same channel, otherwise no response is sent.

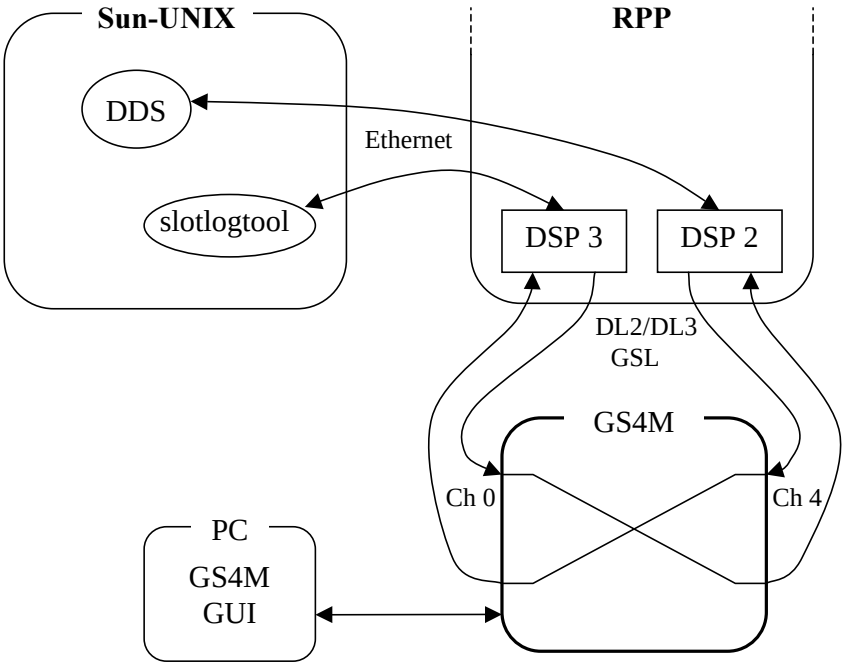

Figure 6.7 *DSP joint test - logical connections.*

Slotlogtool

The slotlogtool program runs on a Unix workstation and communicates with the DSP through the Ethernet. To run the program the “osesysd” daemon and the link handler “lnhethc” has to be running, apart from that the RPP has to be *deblocked.*
The program can both send scripts and log (to shell or file) on one channel. It uses DSP number (0-7), DL2 link number (0-1), channel number (0-127) and a script name as arguments.
If the script name is not added the program will only log on the specified channel, DL2 and DSP. Otherwise it both sends the script and logs on that channel.

Distributed Debug Server

The DDS program runs on a Unix workstation and communicates with the RPP through the Ethernet. To run the program the “osesysd” daemon and the link handler “lnhethc” have to be running.

The DDS is used to “look in the system” on process level, mainly signal tracing and process characteristics.
In the DSP joint test, the DDS is used to activate the GSL interface in the DSP that will be the receiver, the RPP has to be *deblocked* before activation is possible.
The second use of DDS is to trace the signals between the involved blocks in the DSP, if/when the DSP tries to and succeeds to synchronise on the incoming frames.

6.3.2 Verification

The verification is done in four major steps.

1. Start the daemon (osesysd) and the link handler (lnhethc) and also *deblock* the RPP.
2. Activate the GSL interface in a DSP. (see below)
3. Configure the GS4M. (see below)
4. Send synchronisation frames and analyse the result. (see below)

Step 3 and 4 are done one time for each test case mentioned below.

GSL activation

The activation is actually done at a channel level and then a GSL manager inside the RPP connects that channel to a specific DSP according to a load sharing algorithm. The activation is done from the DDS program by sending a signal named “RPRSTARTGSL” to a process named “RGRLCR”. The signal contains, among other data, which channel to activate. The channels 4 and 132 were activated in this case and that means that the GSL manager selected DSP 2 and DSP 4 respectively. The information about which DSP is selected, can be traced in the DDS program. A DDS trace log can be seen in appendix 11.5.

GS4M configuration

One DSP can handle 256 channels divided on two DL2 links, i.e. channels 0 to 127 on DL2 link 0 and channels 128 to 255 on DL2 link 1. In the GS4M the timeslots are connected in groups of four channels, i.e. timeslot 0 in GS4M means channels 0 to 3. So, connecting timeslots 0 and 1 in GS4M means connecting channels 0 and 4 and so on.
In Table 6.1 the logical connections (toward a part of the RPP magazine) between GS4M DL2 links and RPP DL2 links can be seen. In the DSP the DL2 links are numbered 0/1 and from GS4Ms point of view the DL2 links are numbered after board position. Thus, for RPP 0 for example, DL2 0/1 are the same as DL2 1/2 from GS4Ms point of view.

GS4M		RPP magazine		
DL3 (EM)	**DL2 (Link)**	**DL2**	**Board Position**	**Board Type**
-	-	-	For left	EMB
-	-	-	To the right of EMB	DLHB
0-15	0	-	0	EPSB
0-15	1/2	0/1	1/2	RPP 0
0-15	3/4	0/1	3/4	RPP 1

Table 6.1 *DL2 links between GS4M and RPP magazines.*

The test connections through the GS4M were configured in 4x4 different ways. To test the connection through all the TS4B boards, the configurations below are done for DL3 (EM) front-connector 0, 1, 8 and 9.
The test cases were for DSP sender and receiver as follows:

- Same RPP (1) and same DL2, to test both send and receive connections toward the first position on one RPP.
- Same RPP (1) and same DL2, to test both send and receive connections toward the second position on one RPP.
- Same RPP (1) and different DL2, to test connections toward both positions on the same RPP at the same time.
- Different RPP and different DL2, to test connections between two different RPPs.

The DSPs and channels are always different if the RPP is the same. In Table 6.2 the test case configuration can be seen. All the connections in the GS4M are done in both directions.

GS4M				RPP magazine							
Source		**Destination**		**Sender**				**Receiver (GSL activated)**			
Link	**TS**	**Link**	**TS**	**RPP**	**DSP**	**DL2**	**Ch**	**RPP**	**DSP**	**DL2**	**Ch**
3	0	3	1	1	3	0	0	1	2	0	4
4	0	4	1	1	3	1	128	1	4	1	132
3	0	4	1	1	3	0	0	1	4	1	132
1	0	4	1	0	3	0	0	1	4	1	132

Table 6.2 *DSP joint test case configuration.*

Result

When the receiving DSP is synchronising on an incoming frame set, a signal trace is performed in the DDS. From the trace log it can be seen

that the synchronisation was successful. The trace log looks the same for all the test cases. A signal trace log can be seen in appendix 11.7.

The slotlogtool is logging on a channel, the logging is on raw data, i.e. almost impossible to construe. Therefor a tool called “decoder” is used to interpret the data. In appendix 11.4 that interpreted slotlogtool response log can be seen. That log looks the same for all test cases and the response back to the sending DSP was correct.
Thus, the communication in both directions through the GS4M is correct, i.e. the DSP joint test was successful.

7 TSS2000

7.1 General

To enable integration of the TSS2000, the hybrid test environment has to have a signalling interface toward it, Signalling System No 7 and LAPD. In the real GSM/GPRS network the signalling in the BSC node is performed through some hardware parts that are not implemented in the *testrigg*. Therefore the signalling connections have to be made between TSS2000 and the simulated part of the hybrid test environment, SEA. SEA should therefore be able to handle the Signalling System No 7 and LAPD protocols.

7.2 TSS2000 toward SEA

SS7 is a signalling link on the so-called A interface, i.e. toward the MSC. This signalling is needed even when only GPRS traffic is handled.
LAPD is a signalling link on the Abis interface, i.e. toward the BTS node.

The signalling from TSS2000 ST is performed through LAPD and SS7 boards, but the hardware in the *testrigg* is not capable of handling this. The alternative remaining to achieve the desired connection is to have TSS2000 SFT.

There is already work going on with the connections between SEA and TSS2000 SFT and is not a subject for this thesis work. Still, a brief description follows below.

The LAPD interface in SEA is managed by an emulated RPD or RPG (RPemu), see Figure 7.1. The RPD and RPG are RPs that are used for different tasks in the AXE, they are not described further in the report.
The RPemu is created when launching the *dump* in the SEA Configuration Wizard.
The LAPD interface in the TSS2000 SFT is managed by the RPD adapter.
The glue between the two interfaces is a Connection Server (CS).
The LAPD signalling is, when this is written, up and running.

SS7 is mapped directly to the CP through the Message Protocol Handler (MPH) in SEA. MPH is an interface in SEA that allows external tools to directly access SEA components.

The C7 adapter in TSS2000 SFT performs the SS7 connection toward the CP. The tests performed on the SS7 connection (at the time of writing) indicates that it functions correctly.

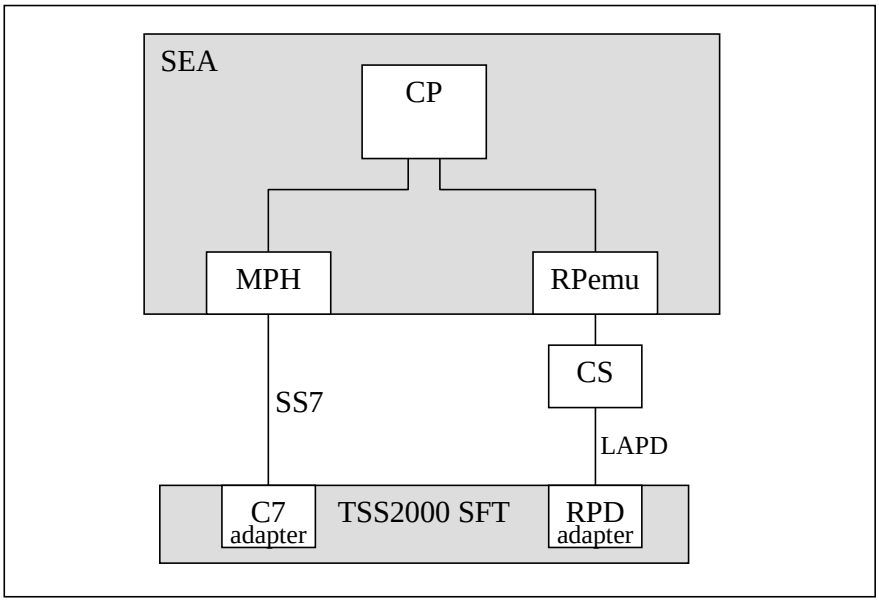

Figure 7.1 *TSS2000 SFT toward SEA connection – overview.*

More information about the signalling interface in TSS2000 can be found in [20].

7.3 TSS2000 toward GS4M

In the connection between TSS2000 and GS4M it is the hardware based TSS2000 that is needed, i.e. TSS2000 ST. The TSS2000 ST has an optional GPRS system. This system has E1 interfaces toward the *testrigg* so the connections have to be made through ETC boards.
Thus, the connections toward GS4M have to be made via the present DLHB board and two ETC boards that can be mounted in the RPP magazine, see Figure 7.2.

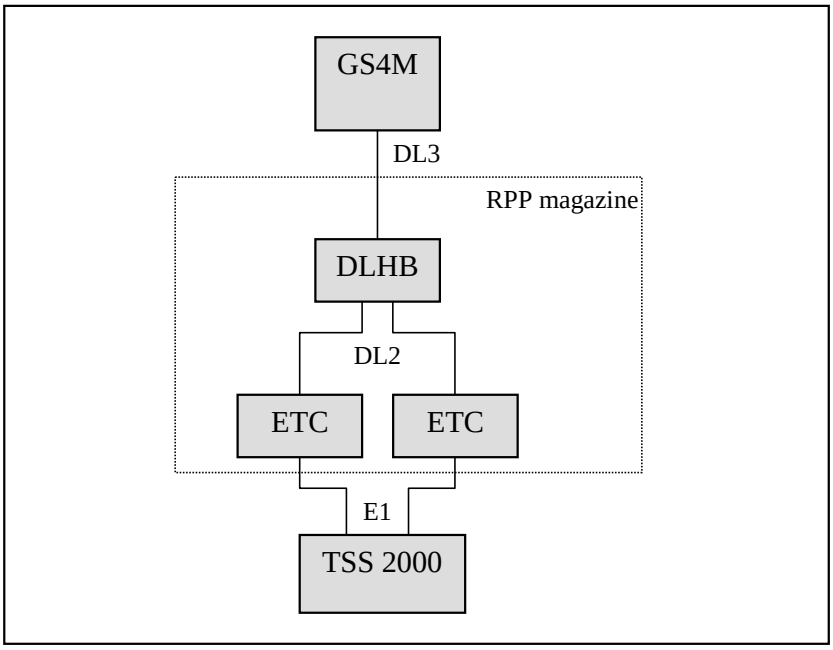

Figure 7.2 *TSS2000 connections toward GS4M.*

8 Result and reflections

The result of this thesis work is a “new” hybrid test environment that has some extensions from the “old” hybrid test environment. An overview of the “new” hybrid test environment can be seen in Figure 8.1. The connections with RPPsim, Adapter and TSS2000 SFT is not, as mentioned before, included in this thesis work. Others have made these connections.

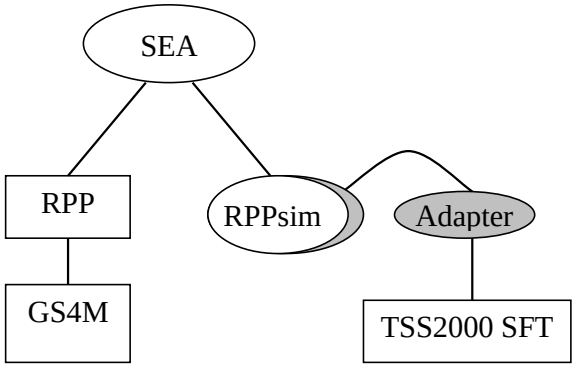

Figure 8.1 *Overview of the “new” hybrid test environment.*

The SEA integration is one extension from the “old” environment that makes it possible to control the RPPs from a CP. The connection between SEA and the RPPs was successful, but the low performance in the load procedure from SEA to RPP can set a limit of the use of it.

Another extension is the GS4M. With Musse connected to the RPP magazine, only one RPP at a time could reach the switching part. The GS4M makes it possible to connect all the RPPs to the switching part simultaneously. That increases the availability of the hybrid test environment so that more users can use it. I.e. it is the number of RPPs installed that sets the limit of number of users, not the switch. As mentioned before, the existing *testrigg* can hold up to 7 RPPs.

The capacity of GS4M makes it also possible to extent the hybrid test environment with up to 16 RPP magazines. That decreases the cost if/when more RPPs are needed.
If, in a future, a subrate switch is integrated in the GS4M, the extension possibility decreases by 2 RPP magazines, because the subrate switch occupies 2 DL3 front-connectors.

The problems that occurred with GS4M are related to the GUI, with the lack of documentation, the alarm indication bug and the weak connection between the log files and the hardware.
Another “problem” with the GS4M is that the hardware is controlled from a PC with Windows 98, which is not allowed to be connected to the Ericsson Ethernet. This means that the flexibility in the use of the hybrid test environment decreases, due to that the users can have different locations while the control of GS4M have to be made from the PC that is connected to the *testrigg*.
But, if the GS4M could be connected to and controlled by the CP (SEA), like it is in a real AXE, these problems would be solved and the same applies to the GUI problems.
Another solution on the location problem could be to develop a UNIX based GUI if the PCEM cards can be used or if corresponding cards are developed for UNIX workstations.
The problems that occurred with the GUI should be solved by the GS4M developers.

With TSS2000 a new problem has occurred. The signalling connection between TSS2000 and SEA demands the SFT variant and the TSS2000 to GS4M connection demands the ST variant. So, with these demands, the hybrid test environment should be like in Figure 8.2.

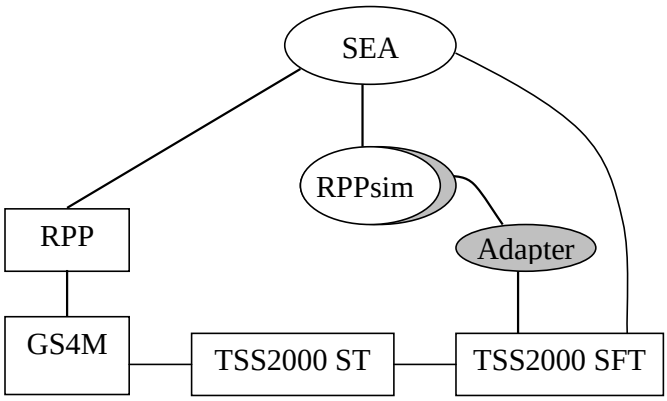

Figure 8.2 *Overview of the “new” hybrid test environment with TSS2000.*

The SEA to TSS2000 SFT connection is already going on in another location inside Ericsson Radio, so this does not have to be investigated further.

Since the TSS2000 ST is not available at the time of writing, the connection between GS4M and TSS2000 ST remains to be investigated. This should not meet with any fundamental technical problems, since both sides have standard interfaces.

The great challenge in the TSS2000 implementation is the connection between the two different versions, because this is a new type of connection that, apparently, has not been done before. This has not been investigated in this thesis work. Thus, it remains to be investigated how such a connection would be made.

The whole cost for the *testrigg* is about 600.000 SEK were the TSS2000 ST stands for about 500.000 SEK, but to make it possible to test the RP software on a higher protocol level the TSS2000 has to be implemented in the *testrigg*. It can be said that the TSS2000 will make a big difference in how many testing hours that can be moved from the AXE target environment to the hybrid test environment.
As a reference can be said that about 10.000 man-hours were spent for Function Test in the latest project and the cost per hour for the AXE target environment is 1.000 SEK.

9 Abbreviations

AB	Automatic Blocked
Abis	Transmission interface BSC-BTS
API	Application Program Interface
APT	AXE Application system
APZ	AXE Control System
AXE	Ericssons total exchange
BSC	Base Station Controller
BSS	Base Station System (BSC and BTS)
BSSGP	BSS GPRS Protocol
BTS	Base Transceiver Station
C7	Signalling System Number 7
CH	Channel
CLB	Clock Board
CLM	Clock Module
COM	Component Object Model
CP	Central Processor
CPS	Central processor subsystem
CPU	Central processor unit
CS	Connection Server
DDS	Distributed Debug Server
DL	Digital link
DL2	Digital link version 2 (2 Mbit/s)
DL2B	Digital link version 2 Backplane
DL3	Digital link version 3 (16 DL2s)
DLHB	Digital Link Half Board (converts one DL3 to 16 DL2)
DLMUX	Digital Link Multiplexor
DSP	Digital Signal Processor
DT	Data Transcript
E1	PCM link for the European standard (2Mbit/s, 32 time slots)
EM	Extension module
EMB	Extension module bus
EPSB	Ethernet packet switch board
ETC	Exchange Terminal Circuit
EX	Executive, working state in CP
FC	Function Change
FT	Function Test
Gb	Transmission interface BSC-SGSN
GDDM	Generic Device and Datacom Magazine

GDDM-H	GDDM -Halfheight
GDM	Generic Device Magazine
GGSN	Gateway GPRS Support Node
GPH	GPRS Packet Handler
GPRS	General Packet Radio Service
GS	Group Switch
GS4M	Group Switch 4k
GSL	GPRS Signalling Link
GSM	Global System for Mobile Communication
GSS	Group Switching Subsystem
GUI	Graphical User Interface
HW	Hardware
ICB	Incoming Clock Board
KMUP	1024 MUP
LAPD	Link Access Procedure for the D-channel
LLC	Logical Link Control
MAC	Medium Access Control
MAS	Maintenance subsystem
MB	Manual Blocked
MML	Man machine language
MPH	Message Protocol Handler
MS	Mobile Station
MSC	Mobile Services Switching Centre
MUP	Multiple Position in the group switch
OS	Operating System
PC	Personal Computer
PCEM	PC mounted GS4M control card
PCI	Peripheral component interconnect
PCM	Pulse Code Modulation
PCU	Packet Control Unit
PDCH	Packet Data Channel
PS	Packet Switched
RCLB	Reference Clock Board
RLC	Radio Link Control
RP	Regional Processor
RP4	Regional Processor version 4
RPB	Regional Processor Bus
RPBH	RPB Handler
RPBH-S	RPB Handler – Serial bus
RPB-S	Regional Processor Bus - Serial
RPCS	RP Connection Server
RPP	Regional Processor with PCI interface
RPX	Regional Processor Cross Connect

SB	Standby, working state in CP
SE	Separated, working state in CP
SEA	Simulator Environment Architecture
SFT	Simulated Function Test
SGSN	Serving GPRS Support Node
SNT	Switching Network Terminal
SPDB	Space Switch Diagonal Board
SPM	Space Switch Module
SRS	Subrate Switch
SRSM	Subrate Switch Module
SS7	Signalling System No. 7
ST	System Test
SU	Software Unit
SUID	Software Unit Identity
SUNAME	Software Unit Name
TS4B	Time Switch Board with 4 TSM Functions
TSM	Time Switch Module
TSM-RP	TSM controlling RP
TSS	Telecom Simulation Systems
WO	Working, working state in CP

10 References

10.1 Internal Ericsson

1. *GS4M switch HW functional unit*
   1551-COA 213 108 Uen

2. *GPRS design guidelines*
   BD/159 41-4/FCP 103 1229/2 Uen

3. *DSP manager in RPP*
   29/15519-ANZ 212 08

4. *BSC Hardware Dimensioning Document*
   1555-AXE 105 07 Uen B

5. *Subsystem GSS*
   1551-ANT 213 17/3 Uen A

6. *GSL handling*
   1551-CAA 204 1063

7. *SYATI: System Function, Application System Start, Initiate*
   7/190 82-CNZ 214 145 Uen B

8. *PTSWI: Processor Test, Switch CP Sides, Initiate*
   28/190 82-CNZ 214 188 Uen A

9. *FCRWS: Function Change RP Working State, Set*
   3/190 82-CNZ 212 225 Uen A

10. *BLRPI: Blocking Functions RP, Initiate*
    2/190 82-CNZ 212 199 Uen A

11. *BLEMI: Blocking Functions, Blocking of EM, Initiate*
    3/190 82-CNZ 212 318 Uen A

12. *EXEME: Exchange Data EM, End*
    1/190 82-CNZ 212 080 Uen B

13. *EXRUE: Exchange Data, RP Software Unit, End*
    3/190 82-CNZ 212 245 Uen A

14. *INFII: IO Subsystem Functions File Administration, Initiate*
6/190 82-CNZ 213 1010 Uen G

15. *LAEUL: Loading Administration, Regional Software Unit, Load*
2/190 82-CNZ 212 278 Uen A

16. *EXRUI: Exchange Data, RP Software Unit, Initiate*
2/190 82-CNZ 212 245 Uen A

17. *EXEMI: Exchange Data Function, EM Initiate*
1/190 82-CNZ 212 338 Uen A

18. *BLEME: Blocking Functions, Blocking of EM, End*
2/190 82-CNZ 212 269 Uen A

19. *CCITT7, Common Channel Signalling Subsystem*,
CCS-7 1551-ANT 218 14/1 Uen C

20. *TSS 2000, Network Traffic Interface*
5/155 19-CRL 113 112 Uen

21. *USER INSTRUCTION FOR GS4M-GROUP SWITCH, IN RIVA*
1/19817-162/FCPW 101 19 Uen PA4

22. *http://infotech.ericsson.se/tsp/products/sea/docs/OnlineHelp/latest/help.html* (Acc 00-06-20)

10.2External

23. *Telecommunications, Telephone Networks 1* (1986),
Ericsson, Televerket and Studentlitteratur
ISBN 91-44-24521-1

24. Walke, Bernhard (2000), "*Mobile Radio Networks, Networking and Protocols*", Chichester, England: John Wiley & Sons Ltd
ISBN 0-471-97595-8

11 Appendices

11.1 Wizard created configuration file

```
# /var/tmp/ejorgen/config/configR3B_ad13_sim.axe
# ------------------------------------------------------------------------
# This is a configuration file for SEA.
# It describes which components that should be created and how they
should
# be connected inbetween themselves to simulate an AXE-10
# To run SEA on this configuration file give the following command:
# sea -config /var/tmp/ejorgen/config/configR3B_ad13_sim.axe
# ------------------------------------------------------------------------
# This file has been automatically generated by SEA Configuration
Wizard R1B
# Executed by ejorgen@y6u643   Thu Apr 13 11:33:45 MET DST 2000
# ------------------------------------------------------------------------

# Creating the CP...
#=================================================
create CP21220.CP21220.1            CP

# Creating RPBHs...
#=================================================
create RPBH.RPBHP.1                 RPBH_0
create RPBH.RPBHS.1                 RPBH_1
create RPBH.RPBHP.1                 RPBH_2
create RPBH.RPBHS.1                 RPBH_3

# Creating RPs...
#=================================================
create RPSIM.RPV.1                  RP_1
create RPSIM.RP4L1G.1               RP_32
create RPSIM.RP4L1G.1               RP_33
create RPSIM.RP4L1G.1               RP_34
create RPSIM.RP4L1G.1               RP_35
create RPSIM.RP4L1G.1               RP_36
create RPSIM.RP4L1G.1               RP_37
create RPSIM.RP4L1G.1               RP_38
create RPSIM.RP4L1G.1               RP_39
create RPSIM.RP4L1A.1               RP_40
create RPSIM.RP4L1A.1               RP_41
```

```
create RPSIM.RP4L1A.1           RP_42
create RPSIM.RP4L1A.1           RP_43
create RPSIM.RP4L1A.1           RP_44
create RPSIM.RP4L1A.1           RP_45
create RPSIM.RP4S1A.1           RP_46
create RPSIM.RP4S1A.1           RP_47
create RPSIM.RP4S1A.1           RP_48
create RPSIM.RP4S1A.1           RP_49
create RPSIM.RPV.1              RP_4
create RPSIM.RPG2A.1            RP_50
create RPSIM.RPG2A.1            RP_51
create RPSIM.RPG2A.1            RP_52
create RPSIM.RPG2A.1            RP_53
create RPSIM.RP4S1A.1           RP_54
create RPSIM.RP4S1A.1           RP_55
create RPSIM.RPG2A.1            RP_56
create RPSIM.RPG2A.1            RP_57
create RPSIM.RPG2A.1            RP_58
create RPSIM.RPG2A.1            RP_59
create RPSIM.RP4S1A.1           RP_96
create RPSIM.RP4S1A.1           RP_97
create RPSIM.RPPS1.1            RP_98
create RPSIM.RPPS1.1            RP_99

# Creating STRs...
#================================================
create STRSIM.STR2C.1           E1A_STR_A
create STRSIM.STR2C.1           E1A_STR_B

# Creating RP software components...
#================================================
create GSS.GSS128.2             GSS
create GSS.TSMR128.2            RP_32_0
create GSS.TSMR128.2            RP_32_1
create RPCMSIM.RPMBHR.1         RP_32_28
create GSS.TSMR128.2            RP_32_2
create RPCMSIM.RPFDR.1          RP_32_30
create RPCMSIM.RPMMR.1          RP_32_31
create GSS.TSMR128.2            RP_32_3
create GSS.TSMR128.2            RP_32_4
create GSS.TSMR128.2            RP_32_5
create GSS.TSMR128.2            RP_32_6
create GSS.TSMR128.2            RP_32_7
```

create RPCMSIM.RPMBHR.1 RP_33_28
create RPCMSIM.RPFDR.1 RP_33_30
create RPCMSIM.RPMMR.1 RP_33_31
create GSS.TSMR128.2 RP_34_0
create GSS.TSMR128.2 RP_34_1
create RPCMSIM.RPMBHR.1 RP_34_28
create GSS.TSMR128.2 RP_34_2
create RPCMSIM.RPFDR.1 RP_34_30
create RPCMSIM.RPMMR.1 RP_34_31
create GSS.TSMR128.2 RP_34_3
create GSS.TSMR128.2 RP_34_4
create GSS.TSMR128.2 RP_34_5
create GSS.TSMR128.2 RP_34_6
create GSS.TSMR128.2 RP_34_7
create RPCMSIM.RPMBHR.1 RP_35_28
create RPCMSIM.RPFDR.1 RP_35_30
create RPCMSIM.RPMMR.1 RP_35_31
create GSS.TSMR128.2 RP_36_0
create GSS.TSMR128.2 RP_36_1
create RPCMSIM.RPMBHR.1 RP_36_28
create GSS.TSMR128.2 RP_36_2
create RPCMSIM.RPFDR.1 RP_36_30
create RPCMSIM.RPMMR.1 RP_36_31
create GSS.TSMR128.2 RP_36_3
create GSS.TSMR128.2 RP_36_4
create GSS.TSMR128.2 RP_36_5
create GSS.TSMR128.2 RP_36_6
create GSS.TSMR128.2 RP_36_7
create RPCMSIM.RPMBHR.1 RP_37_28
create RPCMSIM.RPFDR.1 RP_37_30
create RPCMSIM.RPMMR.1 RP_37_31
create GSS.TSMR128.2 RP_38_0
create GSS.TSMR128.2 RP_38_1
create RPCMSIM.RPMBHR.1 RP_38_28
create GSS.TSMR128.2 RP_38_2
create RPCMSIM.RPFDR.1 RP_38_30
create RPCMSIM.RPMMR.1 RP_38_31
create GSS.TSMR128.2 RP_38_3
create GSS.TSMR128.2 RP_38_4
create GSS.TSMR128.2 RP_38_5
create GSS.TSMR128.2 RP_38_6
create GSS.TSMR128.2 RP_38_7
create RPCMSIM.RPMBHR.1 RP_39_28

```
create RPCMSIM.RPFDR.1          RP_39_30
create RPCMSIM.RPMMR.1          RP_39_31
create RPCMSIM.RPMBHR.1         RP_40_28
create RPCMSIM.RPFDR.1          RP_40_30
create RPCMSIM.RPMMR.1          RP_40_31
create RPCMSIM.RPMBHR.1         RP_41_28
create RPCMSIM.RPFDR.1          RP_41_30
create RPCMSIM.RPMMR.1          RP_41_31
create RPCMSIM.RPMBHR.1         RP_42_28
create RPCMSIM.RPFDR.1          RP_42_30
create RPCMSIM.RPMMR.1          RP_42_31
create RPCMSIM.RPMBHR.1         RP_43_28
create RPCMSIM.RPFDR.1          RP_43_30
create RPCMSIM.RPMMR.1          RP_43_31
create GSS.CLTR.2               RP_44_0
create GSS.CLTR.2               RP_44_1
create RPCMSIM.RPMBHR.1         RP_44_28
create GSS.CLTR.2               RP_44_2
create RPCMSIM.RPFDR.1          RP_44_30
create RPCMSIM.RPMMR.1          RP_44_31
create GSS.CLTR.2               RP_45_0
create GSS.CLTR.2               RP_45_1
create RPCMSIM.RPMBHR.1         RP_45_28
create GSS.CLTR.2               RP_45_2
create RPCMSIM.RPFDR.1          RP_45_30
create RPCMSIM.RPMMR.1          RP_45_31
create RPCMSIM.RPMBHR.1         RP_46_28
create RPCMSIM.RPFDR.1          RP_46_30
create RPCMSIM.RPMMR.1          RP_46_31
create RPCMSIM.RPMBHR.1         RP_47_28
create RPCMSIM.RPFDR.1          RP_47_30
create RPCMSIM.RPMMR.1          RP_47_31
create RPCMSIM.RPMBHR.1         RP_48_28
create RPCMSIM.RPFDR.1          RP_48_30
create RPCMSIM.RPMMR.1          RP_48_31
create RPCMSIM.RPMBHR.1         RP_49_28
create RPCMSIM.RPFDR.1          RP_49_30
create RPCMSIM.RPMMR.1          RP_49_31
create SS7ST.C7ST2CR.1          C7ST2C-0&&-3
create RPCMSIM.RPFDR.1          RP_50_31
create RPCMSIM.RPFDR.1          RP_51_31
create RPCMSIM.CSLSNTR.1        RP_53_0
create RPCMSIM.RPFDR.1          RP_53_19
```

```
create CSLM7R.CSLM7R.1          RP_53_1
create RPCMSIM.RPMBHR.1         RP_54_28
create RPCMSIM.RPFDR.1          RP_54_30
create RPCMSIM.RPMMR.1          RP_54_31
create RPCMSIM.RPMBHR.1         RP_55_28
create RPCMSIM.RPFDR.1          RP_55_30
create RPCMSIM.RPMMR.1          RP_55_31
create RPCMSIM.RPFDR.1          RP_56_31
create RPCMSIM.RPFDR.1          RP_57_31
create RPCMSIM.RPFDR.1          RP_58_31
create RPCMSIM.RPFDR.1          RP_59_31
create RPCMSIM.RPMBHR.1         RP_96_28
create RPCMSIM.RPFDR.1          RP_96_30
create RPCMSIM.RPMMR.1          RP_96_31
create RPCMSIM.RPMBHR.1         RP_97_28
create RPCMSIM.RPFDR.1          RP_97_30
create RPCMSIM.RPMMR.1          RP_97_31
create RPCMSIM.RPIFDR.1         RP_98_29
create RPCMSIM.RPFDR.1          RP_98_31
create RPCMSIM.RPIFDR.1         RP_99_29
create RPCMSIM.RPFDR.1          RP_99_31

# Creating EMRPs...
#================================================
create EMRPSIM.EMRP3.1          E1A_EMRP_0
create EMRPSIM.EMRPD1.1         E1A_EMRP_8

# Creating EMRP Software...
#================================================
create EMRPCMSIM.EMGFDR.1       E1A_EMRP_0_EMGFDR
create EMRPCMSIM.EMGFDR.1       E1A_EMRP_8_EMGFDR

# Connecting RPBHs to the CP...
#================================================
connect RPBH_0          CP      IRphbServer     0
connect RPBH_1          CP      IRphbServer     1
connect RPBH_2          CP      IRphbServer     2
connect RPBH_3          CP      IRphbServer     3

# Connecting RPs to their RPBHs...
#================================================
connect RP_1            RPBH_0  IRpbServer      1
connect RP_32           RPBH_1  IRpbSServer     32
```

```
connect RP_33                RPBH_1          IRpbSServer   33
connect RP_34                RPBH_1          IRpbSServer   34
connect RP_35                RPBH_1          IRpbSServer   35
connect RP_36                RPBH_1          IRpbSServer   36
connect RP_37                RPBH_1          IRpbSServer   37
connect RP_38                RPBH_1          IRpbSServer   38
connect RP_39                RPBH_1          IRpbSServer   39
connect RP_4                 RPBH_0          IRpbServer    4
connect RP_40                RPBH_1          IRpbSServer   40
connect RP_41                RPBH_1          IRpbSServer   41
connect RP_42                RPBH_1          IRpbSServer   42
connect RP_43                RPBH_1          IRpbSServer   43
connect RP_44                RPBH_1          IRpbSServer   44
connect RP_45                RPBH_1          IRpbSServer   45
connect RP_46                RPBH_1          IRpbSServer   46
connect RP_47                RPBH_1          IRpbSServer   47
connect RP_48                RPBH_1          IRpbSServer   48
connect RP_49                RPBH_1          IRpbSServer   49
connect RP_50                RPBH_1          IRpbSServer   50
connect RP_51                RPBH_1          IRpbSServer   51
connect RP_52                RPBH_1          IRpbSServer   52
connect RP_53                RPBH_1          IRpbSServer   53
connect RP_54                RPBH_1          IRpbSServer   54
connect RP_55                RPBH_1          IRpbSServer   55
connect RP_56                RPBH_1          IRpbSServer   56
connect RP_57                RPBH_1          IRpbSServer   57
connect RP_58                RPBH_1          IRpbSServer   58
connect RP_59                RPBH_1          IRpbSServer   59
connect RP_96                RPBH_3          IRpbSServer   96
connect RP_97                RPBH_3          IRpbSServer   97
connect RP_98                RPBH_3          IRpbSServer   98
connect RP_99                RPBH_3          IRpbSServer   99


# Connecting RP software components to their RP...
#================================================
connect RP_32_0              GSS             IGSS          TSM-0
connect RP_32_0              RP_32           IRpCmServer   {}
connect RP_32_1              GSS             IGSS          TSM-1
connect RP_32_1              RP_32           IRpCmServer   {}
connect RP_32_2              GSS             IGSS          TSM-2
connect RP_32_2              RP_32           IRpCmServer   {}
connect RP_32_28             RP_32           IRpCmServer   28
```

connect RP_32_3	GSS	IGSS	TSM-3
connect RP_32_3	RP_32	IRpCmServer	{}
connect RP_32_30	RP_32	IRpCmServer	30
connect RP_32_31	RP_32	IRpCmServer	31
connect RP_32_4	GSS	IGSS	TSM-4
connect RP_32_4	RP_32	IRpCmServer	{}
connect RP_32_5	GSS	IGSS	TSM-5
connect RP_32_5	RP_32	IRpCmServer	{}
connect RP_32_6	GSS	IGSS	TSM-6
connect RP_32_6	RP_32	IRpCmServer	{}
connect RP_32_7	GSS	IGSS	TSM-7
connect RP_32_7	RP_32	IRpCmServer	{}
connect RP_33_28	RP_33	IRpCmServer	28
connect RP_33_30	RP_33	IRpCmServer	30
connect RP_33_31	RP_33	IRpCmServer	31
connect RP_34_0	GSS	IGSS	TSM-8
connect RP_34_0	RP_34	IRpCmServer	{}
connect RP_34_1	GSS	IGSS	TSM-9
connect RP_34_1	RP_34	IRpCmServer	{}
connect RP_34_2	GSS	IGSS	TSM-10
connect RP_34_2	RP_34	IRpCmServer	{}
connect RP_34_28	RP_34	IRpCmServer	28
connect RP_34_3	GSS	IGSS	TSM-11
connect RP_34_3	RP_34	IRpCmServer	{}
connect RP_34_30	RP_34	IRpCmServer	30
connect RP_34_31	RP_34	IRpCmServer	31
connect RP_34_4	GSS	IGSS	TSM-12
connect RP_34_4	RP_34	IRpCmServer	{}
connect RP_34_5	GSS	IGSS	TSM-13
connect RP_34_5	RP_34	IRpCmServer	{}
connect RP_34_6	GSS	IGSS	TSM-14
connect RP_34_6	RP_34	IRpCmServer	{}
connect RP_34_7	GSS	IGSS	TSM-15
connect RP_34_7	RP_34	IRpCmServer	{}
connect RP_35_28	RP_35	IRpCmServer	28
connect RP_35_30	RP_35	IRpCmServer	30
connect RP_35_31	RP_35	IRpCmServer	31
connect RP_36_0	GSS	IGSS	TSM-16
connect RP_36_0	RP_36	IRpCmServer	{}
connect RP_36_1	GSS	IGSS	TSM-17
connect RP_36_1	RP_36	IRpCmServer	{}
connect RP_36_2	GSS	IGSS	TSM-18
connect RP_36_2	RP_36	IRpCmServer	{}

connect RP_36_28	RP_36	IRpCmServer	28
connect RP_36_3	GSS	IGSS	TSM-19
connect RP_36_3	RP_36	IRpCmServer	{}
connect RP_36_30	RP_36	IRpCmServer	30
connect RP_36_31	RP_36	IRpCmServer	31
connect RP_36_4	GSS	IGSS	TSM-20
connect RP_36_4	RP_36	IRpCmServer	{}
connect RP_36_5	GSS	IGSS	TSM-21
connect RP_36_5	RP_36	IRpCmServer	{}
connect RP_36_6	GSS	IGSS	TSM-22
connect RP_36_6	RP_36	IRpCmServer	{}
connect RP_36_7	GSS	IGSS	TSM-23
connect RP_36_7	RP_36	IRpCmServer	{}
connect RP_37_28	RP_37	IRpCmServer	28
connect RP_37_30	RP_37	IRpCmServer	30
connect RP_37_31	RP_37	IRpCmServer	31
connect RP_38_0	GSS	IGSS	TSM-24
connect RP_38_0	RP_38	IRpCmServer	{}
connect RP_38_1	GSS	IGSS	TSM-25
connect RP_38_1	RP_38	IRpCmServer	{}
connect RP_38_2	GSS	IGSS	TSM-26
connect RP_38_2	RP_38	IRpCmServer	{}
connect RP_38_28	RP_38	IRpCmServer	28
connect RP_38_3	GSS	IGSS	TSM-27
connect RP_38_3	RP_38	IRpCmServer	{}
connect RP_38_30	RP_38	IRpCmServer	30
connect RP_38_31	RP_38	IRpCmServer	31
connect RP_38_4	GSS	IGSS	TSM-28
connect RP_38_4	RP_38	IRpCmServer	{}
connect RP_38_5	GSS	IGSS	TSM-29
connect RP_38_5	RP_38	IRpCmServer	{}
connect RP_38_6	GSS	IGSS	TSM-30
connect RP_38_6	RP_38	IRpCmServer	{}
connect RP_38_7	GSS	IGSS	TSM-31
connect RP_38_7	RP_38	IRpCmServer	{}
connect RP_39_28	RP_39	IRpCmServer	28
connect RP_39_30	RP_39	IRpCmServer	30
connect RP_39_31	RP_39	IRpCmServer	31
connect RP_40_28	RP_40	IRpCmServer	28
connect RP_40_30	RP_40	IRpCmServer	30
connect RP_40_31	RP_40	IRpCmServer	31
connect RP_41_28	RP_41	IRpCmServer	28
connect RP_41_30	RP_41	IRpCmServer	30

```
connect RP_41_31        RP_41       IRpCmServer   31
connect RP_42_28        RP_42       IRpCmServer   28
connect RP_42_30        RP_42       IRpCmServer   30
connect RP_42_31        RP_42       IRpCmServer   31
connect RP_43_28        RP_43       IRpCmServer   28
connect RP_43_30        RP_43       IRpCmServer   30
connect RP_43_31        RP_43       IRpCmServer   31
connect RP_44_0         GSS         IGSS          {CLT-0 0}
connect RP_44_0         RP_44       IRpCmServer   {}
connect RP_44_1         GSS         IGSS          {CLT-1 1}
connect RP_44_1         RP_44       IRpCmServer   {}
connect RP_44_2         GSS         IGSS          {CLT-2 2}
connect RP_44_2         RP_44       IRpCmServer   {}
connect RP_44_28        RP_44       IRpCmServer   28
connect RP_44_30        RP_44       IRpCmServer   30
connect RP_44_31        RP_44       IRpCmServer   31
connect RP_45_28        RP_45       IRpCmServer   28
connect RP_45_30        RP_45       IRpCmServer   30
connect RP_45_31        RP_45       IRpCmServer   31
connect RP_46_28        RP_46       IRpCmServer   28
connect RP_46_30        RP_46       IRpCmServer   30
connect RP_46_31        RP_46       IRpCmServer   31
connect RP_47_28        RP_47       IRpCmServer   28
connect RP_47_30        RP_47       IRpCmServer   30
connect RP_47_31        RP_47       IRpCmServer   31
connect RP_48_28        RP_48       IRpCmServer   28
connect RP_48_30        RP_48       IRpCmServer   30
connect RP_48_31        RP_48       IRpCmServer   31
connect RP_49_28        RP_49       IRpCmServer   28
connect RP_49_30        RP_49       IRpCmServer   30
connect RP_49_31        RP_49       IRpCmServer   31
connect RP_50_31        RP_50       IRpCmServer   31
connect RP_51_31        RP_51       IRpCmServer   31
connect RP_53_0         RP_53       IRpCmServer   0
connect RP_53_19        RP_53       IRpCmServer   19
connect RP_54_28        RP_54       IRpCmServer   28
connect RP_54_30        RP_54       IRpCmServer   30
connect RP_54_31        RP_54       IRpCmServer   31
connect RP_55_28        RP_55       IRpCmServer   28
connect RP_55_30        RP_55       IRpCmServer   30
connect RP_55_31        RP_55       IRpCmServer   31
connect RP_56_31        RP_56       IRpCmServer   31
connect RP_57_31        RP_57       IRpCmServer   31
```

```
connect RP_58_31            RP_58          IRpCmServer   31
connect RP_59_31            RP_59          IRpCmServer   31
connect RP_96_28            RP_96          IRpCmServer   28
connect RP_96_30            RP_96          IRpCmServer   30
connect RP_96_31            RP_96          IRpCmServer   31
connect RP_97_28            RP_97          IRpCmServer   28
connect RP_97_30            RP_97          IRpCmServer   30
connect RP_97_31            RP_97          IRpCmServer   31
connect RP_98_29            RP_98          IRpCmServer   29
connect RP_98_31            RP_98          IRpCmServer   31
connect RP_99_29            RP_99          IRpCmServer   29
connect RP_99_31            RP_99          IRpCmServer   31


# Adding ETC  Hardware
#=================================================

# Connecting RP software components to Group Switch...
#=================================================
connect C7ST2C-0&&-3        GSS            IDL2Connect   TSM-9-8
connect C7ST2C-0&&-3        RP_50          IRpCmServer   0
connect RP_53_1             RP_53          IRpCmServer   1
connect RP_53_1             GSS            IDL2Connect   TSM-9-14

# Connecting RPs to GSS...
#=================================================

# Connecting STRs to STCs or GSS....
#=================================================

# Connecting EMRPs to their STRs/RPBCs...
#=================================================
connect E1A_EMRP_0          E1A_STR_A      IEmrpbServer  {0 A}
connect E1A_EMRP_0          E1A_STR_B      IEmrpbServer  {0 B}
connect E1A_EMRP_8          E1A_STR_A      IEmrpbServer  {8 A}
connect E1A_EMRP_8          E1A_STR_B      IEmrpbServer  {8 B}

# Connecting EMRP Software to their EMRP...
#=================================================
connect E1A_EMRP_0_EMGFDR           E1A_EMRP_0
            IEmrpCmServer   {406 }
connect E1A_EMRP_8_EMGFDR           E1A_EMRP_8
            IEmrpCmServer   {406 }
```

```
# Configuring the CP...
#================================================
config CP {
set-processor-configuration -model 21225 -frequency 3
set-memory-configuration -drsdevice 2 -drsboards 2 -psdevice 1 -
pscmdevice 2
load-dump
/var/tmp/ejorgen/dump/R8A2A13_21225_SEA_ejorgen.21220_emudum
p.gz

}
config C7ST2C-0&&-3 {
c7st2c_setdevice -low 0 -high 3
}

#
# List of devices found on dump which are currently not supported
# or not added to the SEA Configuration Wizard.
# ================================================
#



# Unsupported RPs: (0)
# ----------------------------

# Unsupported RP software: (22)
# -----------------------------
# C7ST2CR               RP-51
# ETRALTR               RP-54 RP-55
# ETRBLTR               RP-48 RP-49 RP-54 RP-55
# ETRTGR                RP-54 RP-55
# ETRTTR                RP-48 RP-49
# FSIR                  RP-98 RP-99
# RCLCCHR               RP-56 RP-57 RP-58 RP-59
# RCSCBR                RP-56 RP-57 RP-58 RP-59
# RGRLCR                RP-98 RP-99
# RGSERVR               RP-98 RP-99
# RHLAPDR               RP-56 RP-57 RP-58 RP-59
# RHSNTR                RP-56 RP-57 RP-58 RP-59
# RMPAGR                RP-56 RP-57 RP-58 RP-59
# RQRCQSR               RP-56 RP-57 RP-58 RP-59
```

```
# RQUNCR            RP-56 RP-57 RP-58 RP-59
# RTGBR             RP-98 RP-99
# RTGPHDVR          RP-98 RP-99
# RTTF1S1R          RP-46 RP-47
# RTTF1S2R          RP-46 RP-47
# RTTH1SR           RP-46 RP-47
# SRSR              RP-40 RP-41 RP-42 RP-43
# TERTR             RP-32 RP-33 RP-34 RP-35 RP-36 RP-37
                    RP-38 RP-39 RP-40 RP-41 RP-42 RP-43
                    RP-44 RP-45 RP-46 RP-47 RP-48 RP-49
                    RP-54 RP-55 RP-96 RP-97

# Unsupported STRs: (0)
# ----------------------------

# Unsupported EMRPs: (0)
# ----------------------------

# Unsupported EMRP software: (7)
# ----------------------------
# EXAL0R        E1A_0
# RICSR         E1A_0
# RILCOR        E1A_8
# RILTR         E1A_0
# RITSR         E1A_0
# TEETR         E1A_0
# TWR           E1A_0

# Default Software Simulations Used For The Following RP/EMRP
Software
#================================================
# "C7ST2CR      9000/CAA 140 053/3A         R1B01"
# "RPMBHR      1/CAA 135 2518/RPMBH         R1B01"
# "TSMR         1/CAA 135 2511/TSM          R1A17"
#================================================
```

11.2Altered configuration file

```
# /var/tmp/ejorgen/config/configR3B_ad13.axe
# ------------------------------------------------------------------------
# This is a configuration file for SEA.
# It describes which components that should be created and how they
should
# be connected inbetween themselves to simulate an AXE-10
# To run SEA on this configuration file give the following command:
# sea -config /var/tmp/ejorgen/config/configR3B_ad13.axe
# ------------------------------------------------------------------------
# This file has been created from
/var/tmp/ejorgen/config/configR3B_ad13.axe
# and altered to fit to RP HW connection.
# ------------------------------------------------------------------------

# Creating the CP...
#=================================================
create CP21220.CP21220.1          CP

# Creating RPBHs...
#=================================================
create RPBH.RPBHP.1               RPBH_0
create RPBH.RPBHS.1               RPBH_1
create RPBH.RPBHP.1               RPBH_2
create RPBH.RPBHS.1               RPBH_3

# Creating RPs...
#=================================================
create RPSIM.RPV.1                RP_1
create RPSIM.RP4L1G.1             RP_32
create RPSIM.RP4L1G.1             RP_33
create RPSIM.RP4L1G.1             RP_34
create RPSIM.RP4L1G.1             RP_35
create RPSIM.RP4L1G.1             RP_36
create RPSIM.RP4L1G.1             RP_37
create RPSIM.RP4L1G.1             RP_38
create RPSIM.RP4L1G.1             RP_39
create RPSIM.RP4L1A.1             RP_40
create RPSIM.RP4L1A.1             RP_41
create RPSIM.RP4L1A.1             RP_42
create RPSIM.RP4L1A.1             RP_43
create RPSIM.RP4L1A.1             RP_44
```

```
create RPSIM.RP4L1A.1           RP_45
create RPSIM.RP4S1A.1           RP_46
create RPSIM.RP4S1A.1           RP_47
create RPSIM.RP4S1A.1           RP_48
create RPSIM.RP4S1A.1           RP_49
create RPSIM.RPV.1              RP_4
create RPSIM.RPG2A.1            RP_50
create RPSIM.RPG2A.1            RP_51
create RPSIM.RPG2A.1            RP_52
create RPSIM.RPG2A.1            RP_53
create RPSIM.RP4S1A.1           RP_54
create RPSIM.RP4S1A.1           RP_55
create RPSIM.RPG2A.1            RP_56
create RPSIM.RPG2A.1            RP_57
create RPSIM.RPG2A.1            RP_58
create RPSIM.RPG2A.1            RP_59
create RPSIM.RP4S1A.1           RP_96
create RPSIM.RP4S1A.1           RP_97
#create RPSIM.RPPS1.1           RP_98
create RPSIM.RPPS1.1            RP_99

create RPPROXY.RPPROXYS.1       RPP_HW

# Creating STRs...
#================================================
create STRSIM.STR2C.1           E1A_STR_A
create STRSIM.STR2C.1           E1A_STR_B

# Creating RP software components...
#================================================
create GSS.GSS128.2             GSS
create GSS.TSMR128.2            RP_32_0
create GSS.TSMR128.2            RP_32_1
create RPCMSIM.RPMBHR.1         RP_32_28
create GSS.TSMR128.2            RP_32_2
create RPCMSIM.RPFDR.1          RP_32_30
create RPCMSIM.RPMMR.1          RP_32_31
create GSS.TSMR128.2            RP_32_3
create GSS.TSMR128.2            RP_32_4
create GSS.TSMR128.2            RP_32_5
create GSS.TSMR128.2            RP_32_6
create GSS.TSMR128.2            RP_32_7
create RPCMSIM.RPMBHR.1         RP_33_28
```

create RPCMSIM.RPFDR.1	RP_33_30
create RPCMSIM.RPMMR.1	RP_33_31
create GSS.TSMR128.2	RP_34_0
create GSS.TSMR128.2	RP_34_1
create RPCMSIM.RPMBHR.1	RP_34_28
create GSS.TSMR128.2	RP_34_2
create RPCMSIM.RPFDR.1	RP_34_30
create RPCMSIM.RPMMR.1	RP_34_31
create GSS.TSMR128.2	RP_34_3
create GSS.TSMR128.2	RP_34_4
create GSS.TSMR128.2	RP_34_5
create GSS.TSMR128.2	RP_34_6
create GSS.TSMR128.2	RP_34_7
create RPCMSIM.RPMBHR.1	RP_35_28
create RPCMSIM.RPFDR.1	RP_35_30
create RPCMSIM.RPMMR.1	RP_35_31
create GSS.TSMR128.2	RP_36_0
create GSS.TSMR128.2	RP_36_1
create RPCMSIM.RPMBHR.1	RP_36_28
create GSS.TSMR128.2	RP_36_2
create RPCMSIM.RPFDR.1	RP_36_30
create RPCMSIM.RPMMR.1	RP_36_31
create GSS.TSMR128.2	RP_36_3
create GSS.TSMR128.2	RP_36_4
create GSS.TSMR128.2	RP_36_5
create GSS.TSMR128.2	RP_36_6
create GSS.TSMR128.2	RP_36_7
create RPCMSIM.RPMBHR.1	RP_37_28
create RPCMSIM.RPFDR.1	RP_37_30
create RPCMSIM.RPMMR.1	RP_37_31
create GSS.TSMR128.2	RP_38_0
create GSS.TSMR128.2	RP_38_1
create RPCMSIM.RPMBHR.1	RP_38_28
create GSS.TSMR128.2	RP_38_2
create RPCMSIM.RPFDR.1	RP_38_30
create RPCMSIM.RPMMR.1	RP_38_31
create GSS.TSMR128.2	RP_38_3
create GSS.TSMR128.2	RP_38_4
create GSS.TSMR128.2	RP_38_5
create GSS.TSMR128.2	RP_38_6
create GSS.TSMR128.2	RP_38_7
create RPCMSIM.RPMBHR.1	RP_39_28
create RPCMSIM.RPFDR.1	RP_39_30

create RPCMSIM.RPMMR.1	RP_39_31
create RPCMSIM.RPMBHR.1	RP_40_28
create RPCMSIM.RPFDR.1	RP_40_30
create RPCMSIM.RPMMR.1	RP_40_31
create RPCMSIM.RPMBHR.1	RP_41_28
create RPCMSIM.RPFDR.1	RP_41_30
create RPCMSIM.RPMMR.1	RP_41_31
create RPCMSIM.RPMBHR.1	RP_42_28
create RPCMSIM.RPFDR.1	RP_42_30
create RPCMSIM.RPMMR.1	RP_42_31
create RPCMSIM.RPMBHR.1	RP_43_28
create RPCMSIM.RPFDR.1	RP_43_30
create RPCMSIM.RPMMR.1	RP_43_31
create GSS.CLTR.2	RP_44_0
create GSS.CLTR.2	RP_44_1
create RPCMSIM.RPMBHR.1	RP_44_28
create GSS.CLTR.2	RP_44_2
create RPCMSIM.RPFDR.1	RP_44_30
create RPCMSIM.RPMMR.1	RP_44_31
create GSS.CLTR.2	RP_45_0
create GSS.CLTR.2	RP_45_1
create RPCMSIM.RPMBHR.1	RP_45_28
create GSS.CLTR.2	RP_45_2
create RPCMSIM.RPFDR.1	RP_45_30
create RPCMSIM.RPMMR.1	RP_45_31
create RPCMSIM.RPMBHR.1	RP_46_28
create RPCMSIM.RPFDR.1	RP_46_30
create RPCMSIM.RPMMR.1	RP_46_31
create RPCMSIM.RPMBHR.1	RP_47_28
create RPCMSIM.RPFDR.1	RP_47_30
create RPCMSIM.RPMMR.1	RP_47_31
create RPCMSIM.RPMBHR.1	RP_48_28
create RPCMSIM.RPFDR.1	RP_48_30
create RPCMSIM.RPMMR.1	RP_48_31
create RPCMSIM.RPMBHR.1	RP_49_28
create RPCMSIM.RPFDR.1	RP_49_30
create RPCMSIM.RPMMR.1	RP_49_31
create SS7ST.C7ST2CR.1	C7ST2C-0&&-3
create RPCMSIM.RPFDR.1	RP_50_31
create RPCMSIM.RPFDR.1	RP_51_31
create RPCMSIM.CSLSNTR.1	RP_53_0
create RPCMSIM.RPFDR.1	RP_53_19
create CSLM7R.CSLM7R.1	RP_53_1

```
create RPCMSIM.RPMBHR.1        RP_54_28
create RPCMSIM.RPFDR.1         RP_54_30
create RPCMSIM.RPMMR.1         RP_54_31
create RPCMSIM.RPMBHR.1        RP_55_28
create RPCMSIM.RPFDR.1         RP_55_30
create RPCMSIM.RPMMR.1         RP_55_31
create RPCMSIM.RPFDR.1         RP_56_31
create RPCMSIM.RPFDR.1         RP_57_31
create RPCMSIM.RPFDR.1         RP_58_31
create RPCMSIM.RPFDR.1         RP_59_31
create RPCMSIM.RPMBHR.1        RP_96_28
create RPCMSIM.RPFDR.1         RP_96_30
create RPCMSIM.RPMMR.1         RP_96_31
create RPCMSIM.RPMBHR.1        RP_97_28
create RPCMSIM.RPFDR.1         RP_97_30
create RPCMSIM.RPMMR.1         RP_97_31
#create RPCMSIM.RPIFDR.1       RP_98_29
#create RPCMSIM.RPFDR.1        RP_98_31
create RPCMSIM.RPIFDR.1        RP_99_29
create RPCMSIM.RPFDR.1         RP_99_31

# Creating EMRPs...
#================================================
create EMRPSIM.EMRP3.1         E1A_EMRP_0
create EMRPSIM.EMRPD1.1        E1A_EMRP_8

# Creating EMRP Software...
#================================================
create EMRPCMSIM.EMGFDR.1      E1A_EMRP_0_EMGFDR
create EMRPCMSIM.EMGFDR.1      E1A_EMRP_8_EMGFDR

# Connecting RPBHs to the CP...
#================================================
connect RPBH_0          CP          IRphbServer   0
connect RPBH_1          CP          IRphbServer   1
connect RPBH_2          CP          IRphbServer   2
connect RPBH_3          CP          IRphbServer   3

# Connecting RPs to their RPBHs...
#================================================
connect RP_1            RPBH_0      IRpbServer    1
connect RP_32           RPBH_1      IRpbSServer   32
connect RP_33           RPBH_1      IRpbSServer   33
```

```
connect RP_34                RPBH_1          IRpbSServer    34
connect RP_35                RPBH_1          IRpbSServer    35
connect RP_36                RPBH_1          IRpbSServer    36
connect RP_37                RPBH_1          IRpbSServer    37
connect RP_38                RPBH_1          IRpbSServer    38
connect RP_39                RPBH_1          IRpbSServer    39
connect RP_4                 RPBH_0          IRpbServer     4
connect RP_40                RPBH_1          IRpbSServer    40
connect RP_41                RPBH_1          IRpbSServer    41
connect RP_42                RPBH_1          IRpbSServer    42
connect RP_43                RPBH_1          IRpbSServer    43
connect RP_44                RPBH_1          IRpbSServer    44
connect RP_45                RPBH_1          IRpbSServer    45
connect RP_46                RPBH_1          IRpbSServer    46
connect RP_47                RPBH_1          IRpbSServer    47
connect RP_48                RPBH_1          IRpbSServer    48
connect RP_49                RPBH_1          IRpbSServer    49
connect RP_50                RPBH_1          IRpbSServer    50
connect RP_51                RPBH_1          IRpbSServer    51
connect RP_52                RPBH_1          IRpbSServer    52
connect RP_53                RPBH_1          IRpbSServer    53
connect RP_54                RPBH_1          IRpbSServer    54
connect RP_55                RPBH_1          IRpbSServer    55
connect RP_56                RPBH_1          IRpbSServer    56
connect RP_57                RPBH_1          IRpbSServer    57
connect RP_58                RPBH_1          IRpbSServer    58
connect RP_59                RPBH_1          IRpbSServer    59
connect RP_96                RPBH_3          IRpbSServer    96
connect RP_97                RPBH_3          IRpbSServer    97
#connect RP_98               RPBH_3          IRpbSServer    98
connect RPP_HW               RPBH_3          IRpbSServer    {98 RPPHW}
connect RP_99                RPBH_3          IRpbSServer    99


# Connecting RP software components to their RP...
#================================================
connect RP_32_0              GSS             IGSS           TSM-0
connect RP_32_0              RP_32           IRpCmServer    {}
connect RP_32_1              GSS             IGSS           TSM-1
connect RP_32_1              RP_32           IRpCmServer    {}
connect RP_32_2              GSS             IGSS           TSM-2
connect RP_32_2              RP_32           IRpCmServer    {}
connect RP_32_28             RP_32           IRpCmServer    28
```

connect RP_32_3	GSS	IGSS	TSM-3
connect RP_32_3	RP_32	IRpCmServer	{}
connect RP_32_30	RP_32	IRpCmServer	30
connect RP_32_31	RP_32	IRpCmServer	31
connect RP_32_4	GSS	IGSS	TSM-4
connect RP_32_4	RP_32	IRpCmServer	{}
connect RP_32_5	GSS	IGSS	TSM-5
connect RP_32_5	RP_32	IRpCmServer	{}
connect RP_32_6	GSS	IGSS	TSM-6
connect RP_32_6	RP_32	IRpCmServer	{}
connect RP_32_7	GSS	IGSS	TSM-7
connect RP_32_7	RP_32	IRpCmServer	{}
connect RP_33_28	RP_33	IRpCmServer	28
connect RP_33_30	RP_33	IRpCmServer	30
connect RP_33_31	RP_33	IRpCmServer	31
connect RP_34_0	GSS	IGSS	TSM-8
connect RP_34_0	RP_34	IRpCmServer	{}
connect RP_34_1	GSS	IGSS	TSM-9
connect RP_34_1	RP_34	IRpCmServer	{}
connect RP_34_2	GSS	IGSS	TSM-10
connect RP_34_2	RP_34	IRpCmServer	{}
connect RP_34_28	RP_34	IRpCmServer	28
connect RP_34_3	GSS	IGSS	TSM-11
connect RP_34_3	RP_34	IRpCmServer	{}
connect RP_34_30	RP_34	IRpCmServer	30
connect RP_34_31	RP_34	IRpCmServer	31
connect RP_34_4	GSS	IGSS	TSM-12
connect RP_34_4	RP_34	IRpCmServer	{}
connect RP_34_5	GSS	IGSS	TSM-13
connect RP_34_5	RP_34	IRpCmServer	{}
connect RP_34_6	GSS	IGSS	TSM-14
connect RP_34_6	RP_34	IRpCmServer	{}
connect RP_34_7	GSS	IGSS	TSM-15
connect RP_34_7	RP_34	IRpCmServer	{}
connect RP_35_28	RP_35	IRpCmServer	28
connect RP_35_30	RP_35	IRpCmServer	30
connect RP_35_31	RP_35	IRpCmServer	31
connect RP_36_0	GSS	IGSS	TSM-16
connect RP_36_0	RP_36	IRpCmServer	{}
connect RP_36_1	GSS	IGSS	TSM-17
connect RP_36_1	RP_36	IRpCmServer	{}
connect RP_36_2	GSS	IGSS	TSM-18
connect RP_36_2	RP_36	IRpCmServer	{}

connect RP_36_28	RP_36	IRpCmServer	28
connect RP_36_3	GSS	IGSS	TSM-19
connect RP_36_3	RP_36	IRpCmServer	{}
connect RP_36_30	RP_36	IRpCmServer	30
connect RP_36_31	RP_36	IRpCmServer	31
connect RP_36_4	GSS	IGSS	TSM-20
connect RP_36_4	RP_36	IRpCmServer	{}
connect RP_36_5	GSS	IGSS	TSM-21
connect RP_36_5	RP_36	IRpCmServer	{}
connect RP_36_6	GSS	IGSS	TSM-22
connect RP_36_6	RP_36	IRpCmServer	{}
connect RP_36_7	GSS	IGSS	TSM-23
connect RP_36_7	RP_36	IRpCmServer	{}
connect RP_37_28	RP_37	IRpCmServer	28
connect RP_37_30	RP_37	IRpCmServer	30
connect RP_37_31	RP_37	IRpCmServer	31
connect RP_38_0	GSS	IGSS	TSM-24
connect RP_38_0	RP_38	IRpCmServer	{}
connect RP_38_1	GSS	IGSS	TSM-25
connect RP_38_1	RP_38	IRpCmServer	{}
connect RP_38_2	GSS	IGSS	TSM-26
connect RP_38_2	RP_38	IRpCmServer	{}
connect RP_38_28	RP_38	IRpCmServer	28
connect RP_38_3	GSS	IGSS	TSM-27
connect RP_38_3	RP_38	IRpCmServer	{}
connect RP_38_30	RP_38	IRpCmServer	30
connect RP_38_31	RP_38	IRpCmServer	31
connect RP_38_4	GSS	IGSS	TSM-28
connect RP_38_4	RP_38	IRpCmServer	{}
connect RP_38_5	GSS	IGSS	TSM-29
connect RP_38_5	RP_38	IRpCmServer	{}
connect RP_38_6	GSS	IGSS	TSM-30
connect RP_38_6	RP_38	IRpCmServer	{}
connect RP_38_7	GSS	IGSS	TSM-31
connect RP_38_7	RP_38	IRpCmServer	{}
connect RP_39_28	RP_39	IRpCmServer	28
connect RP_39_30	RP_39	IRpCmServer	30
connect RP_39_31	RP_39	IRpCmServer	31
connect RP_40_28	RP_40	IRpCmServer	28
connect RP_40_30	RP_40	IRpCmServer	30
connect RP_40_31	RP_40	IRpCmServer	31
connect RP_41_28	RP_41	IRpCmServer	28
connect RP_41_30	RP_41	IRpCmServer	30

```
connect RP_41_31        RP_41       IRpCmServer  31
connect RP_42_28        RP_42       IRpCmServer  28
connect RP_42_30        RP_42       IRpCmServer  30
connect RP_42_31        RP_42       IRpCmServer  31
connect RP_43_28        RP_43       IRpCmServer  28
connect RP_43_30        RP_43       IRpCmServer  30
connect RP_43_31        RP_43       IRpCmServer  31
connect RP_44_0         GSS         IGSS         {CLT-0 0}
connect RP_44_0         RP_44       IRpCmServer  {}
connect RP_44_1         GSS         IGSS         {CLT-1 1}
connect RP_44_1         RP_44       IRpCmServer  {}
connect RP_44_2         GSS         IGSS         {CLT-2 2}
connect RP_44_2         RP_44       IRpCmServer  {}
connect RP_44_28        RP_44       IRpCmServer  28
connect RP_44_30        RP_44       IRpCmServer  30
connect RP_44_31        RP_44       IRpCmServer  31
connect RP_45_28        RP_45       IRpCmServer  28
connect RP_45_30        RP_45       IRpCmServer  30
connect RP_45_31        RP_45       IRpCmServer  31
connect RP_46_28        RP_46       IRpCmServer  28
connect RP_46_30        RP_46       IRpCmServer  30
connect RP_46_31        RP_46       IRpCmServer  31
connect RP_47_28        RP_47       IRpCmServer  28
connect RP_47_30        RP_47       IRpCmServer  30
connect RP_47_31        RP_47       IRpCmServer  31
connect RP_48_28        RP_48       IRpCmServer  28
connect RP_48_30        RP_48       IRpCmServer  30
connect RP_48_31        RP_48       IRpCmServer  31
connect RP_49_28        RP_49       IRpCmServer  28
connect RP_49_30        RP_49       IRpCmServer  30
connect RP_49_31        RP_49       IRpCmServer  31
connect RP_50_31        RP_50       IRpCmServer  31
connect RP_51_31        RP_51       IRpCmServer  31
connect RP_53_0         RP_53       IRpCmServer  0
connect RP_53_19        RP_53       IRpCmServer  19
connect RP_54_28        RP_54       IRpCmServer  28
connect RP_54_30        RP_54       IRpCmServer  30
connect RP_54_31        RP_54       IRpCmServer  31
connect RP_55_28        RP_55       IRpCmServer  28
connect RP_55_30        RP_55       IRpCmServer  30
connect RP_55_31        RP_55       IRpCmServer  31
connect RP_56_31        RP_56       IRpCmServer  31
connect RP_57_31        RP_57       IRpCmServer  31
```

```
connect RP_58_31            RP_58          IRpCmServer   31
connect RP_59_31            RP_59          IRpCmServer   31
connect RP_96_28            RP_96          IRpCmServer   28
connect RP_96_30            RP_96          IRpCmServer   30
connect RP_96_31            RP_96          IRpCmServer   31
connect RP_97_28            RP_97          IRpCmServer   28
connect RP_97_30            RP_97          IRpCmServer   30
connect RP_97_31            RP_97          IRpCmServer   31
#connect RP_98_29           RP_98          IRpCmServer   29
#connect RP_98_31           RP_98          IRpCmServer   31
connect RP_99_29            RP_99          IRpCmServer   29
connect RP_99_31            RP_99          IRpCmServer   31


# Adding ETC  Hardware
#=================================================

# Connecting RP software components to Group Switch...
#=================================================
connect C7ST2C-0&&-3        GSS            IDL2Connect   TSM-9-8
connect C7ST2C-0&&-3        RP_50          IRpCmServer   0
connect RP_53_1             RP_53          IRpCmServer   1
connect RP_53_1             GSS            IDL2Connect   TSM-9-14

# Connecting RPs to GSS...
#=================================================

# Connecting STRs to STCs or GSS....
#=================================================

# Connecting EMRPs to their STRs/RPBCs...
#=================================================
connect E1A_EMRP_0          E1A_STR_A      IEmrpbServer  {0 A}
connect E1A_EMRP_0          E1A_STR_B      IEmrpbServer  {0 B}
connect E1A_EMRP_8          E1A_STR_A      IEmrpbServer  {8 A}
connect E1A_EMRP_8          E1A_STR_B      IEmrpbServer  {8 B}

# Connecting EMRP Software to their EMRP...
#=================================================
connect E1A_EMRP_0_EMGFDR           E1A_EMRP_0
            IEmrpCmServer   {406 }
connect E1A_EMRP_8_EMGFDR           E1A_EMRP_8
            IEmrpCmServer   {406 }
```

```
# Configuring the CP...
#================================================
config CP {
set-processor-configuration -model 21225 -frequency 3
set-memory-configuration -drsdevice 2 -drsboards 2 -psdevice 1 -
pscmdevice 2
load-dump
/var/tmp/ejorgen/dump/R8A2A13_21225_SEA_ejorgen.21220_emudum
p.gz
}

config RPP_HW {
        rpproxy_setstate -bl 1
        rpproxy_startcs -trace -hp "rp1111/CP_sim"
}

config C7ST2C-0&&-3 {
c7st2c_setdevice -low 0 -high 3
}

#
# List of devices found on dump which are currently not supported
# or not added to the SEA Configuration Wizard.
# ================================================
#



# Unsupported RPs: (0)
# ---------------------------

# Unsupported RP software: (22)
# ----------------------------
# C7ST2CR         RP-51
# ETRALTR         RP-54 RP-55
# ETRBLTR         RP-48 RP-49 RP-54 RP-55
# ETRTGR          RP-54 RP-55
# ETRTTR          RP-48 RP-49
# FSIR            RP-98 RP-99
# RCLCCHR         RP-56 RP-57 RP-58 RP-59
# RCSCBR          RP-56 RP-57 RP-58 RP-59
# RGRLCR          RP-98 RP-99
```

```
# RGSERVR       RP-98 RP-99
# RHLAPDR       RP-56 RP-57 RP-58 RP-59
# RHSNTR        RP-56 RP-57 RP-58 RP-59
# RMPAGR        RP-56 RP-57 RP-58 RP-59
# RQRCQSR       RP-56 RP-57 RP-58 RP-59
# RQUNCR        RP-56 RP-57 RP-58 RP-59
# RTGBR         RP-98 RP-99
# RTGPHDVR      RP-98 RP-99
# RTTF1S1R      RP-46 RP-47
# RTTF1S2R      RP-46 RP-47
# RTTH1SR       RP-46 RP-47
# SRSR          RP-40 RP-41 RP-42 RP-43
# TERTR         RP-32 RP-33 RP-34 RP-35 RP-36 RP-37 RP-38
                RP-39 RP-40 RP-41 RP-42 RP-43 RP-44 RP-45
                RP-46 RP-47 RP-48 RP-49 RP-54 RP-55 RP-96
                RP-97

# Unsupported STRs: (0)
# -----------------------------

# Unsupported EMRPs: (0)
# -----------------------------

# Unsupported EMRP software: (7)
# -----------------------------
# EXAL0R        E1A_0
# RICSR         E1A_0
# RILCOR        E1A_8
# RILTR         E1A_0
# RITSR         E1A_0
# TEETR         E1A_0
# TWR           E1A_0

# Default Software Simulations Used For The Following RP/EMRP
Software
#================================================
# "C7ST2CR      9000/CAA 140 053/3A       R1B01"
# "RPMBHR       1/CAA 135 2518/RPMBH      R1B01"
# "TSMR         1/CAA 135 2511/TSM        R1A17"
```

11.3Synchronisation frames

##################################
This file is based on
/home/proj_y8/bsccadmin/public_html/abis_logs/bts-sync-1.processed.log
##################################

##################################
Block number : 0
PSEQ:4194303, mod PSEQ:9
aFNu:1901712, mod aFNu:7
aFNd:1901713, mod aFNd:8
TAV : 0
##################################
0x0000
0xd4ef
0xffff
0xffff
0xffff
0xffba
0x8490
0xffff
0xffba
0x8491
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 1
PSEQ:4194303, mod PSEQ:9
aFNu:1901716, mod aFNu:11
aFNd:1901718, mod aFNd:0
TAV : 0

```
##################################
0x0000
0xd4ef
0xffff
0xffff
0xffff
0xffba
0x8490
0xffff
0xffba
0x8491
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 2
PSEQ:4194303, mod PSEQ:9
aFNu:1901721, mod aFNu:3
aFNd:1901722, mod aFNd:4
TAV : 0
##################################
0x0000
0xd4ef
0xffff
0xffff
0xffff
0xffba
0x8499
0xffff
0xffba
0x849a
0xffff
0xffff
0xffff
0xffff
```

0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 3
PSEQ:0, mod PSEQ:0
aFNu:1901725, mod aFNu:7
aFNd:1901726, mod aFNd:8
TAV : 0
##################################
0x0000
0xd4ef
0xff80
0x8000
0xffff
0xffba
0x849d
0xffff
0xffba
0x849e
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 4
PSEQ:4, mod PSEQ:4
aFNu:1901729, mod aFNu:11
aFNd:1901731, mod aFNd:0
TAV : 0
##################################
0x0000

0xd4ef
0xff80
0x8004
0xffff
0xffba
0x84a1
0xffff
0xffba
0x84a3
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 5
PSEQ:8, mod PSEQ:8
aFNu:1901734, mod aFNu:3
aFNd:1901735, mod aFNd:4
TAV : 0
##################################
0x0000
0xd4ef
0xff80
0x8008
0xffff
0xffba
0x84a6
0xffff
0xffba
0x84a7
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

0xffff
0xffff
0xffff
0xffff

##################################
Block number : 6
PSEQ:13, mod PSEQ:0
aFNu:1901738, mod aFNu:7
aFNd:1901739, mod aFNd:8
TAV : 0
##################################
0x0000
0xd4ef
0xff80
0x800d
0xffff
0xffba
0x84aa
0xffff
0xffba
0x84ab
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 7
PSEQ:17, mod PSEQ:4
aFNu:1901742, mod aFNu:11
aFNd:1901744, mod aFNd:0
TAV : 0
##################################
0x0000
0xd4ef
0xff80

0x8011
0xffff
0xffba
0x84ae
0xffff
0xffba
0x84b0
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

##################################
Block number : 8
PSEQ:21, mod PSEQ:8
aFNu:1901747, mod aFNu:3
aFNd:1901748, mod aFNd:4
TAV : 0
##################################
0x0000
0xd4ef
0xff80
0x8015
0xffff
0xffba
0x84b3
0xffff
0xffba
0x84b4
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

0xffff
0xffff

##################################
Block number : 9
PSEQ:26, mod PSEQ:0
aFNu:1901751, mod aFNu:7
aFNd:1901752, mod aFNd:8
TAV : 0
##################################
0x0000
0xd4ef
0xff80
0x801a
0xffff
0xffba
0x84b7
0xffff
0xffba
0x84b8
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff
0xffff

11.4Slotlogtool response log

test1013.ejorgen.47> iostream_decoder -bsc log_3_0_0
decode an output stream from bsc
PCU-SYNC frame detected
Block number : 0
0000000000000000 0x0000
1111100001111111 0xf87f
1111111110000000 0xff80
1000000000000000 0x8000
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:0, mod PSEQ:0
TAV:0
UFE, Uplink Frame Error detected

PCU-SYNC frame detected
Block number : 1
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000000100 0x8004
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff

1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:4, mod PSEQ:4
TAV:0

PCU-SYNC frame detected
Block number : 2
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000001000 0x8008
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:8, mod PSEQ:8
TAV:0

PCU-SYNC frame detected
Block number : 3
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000001101 0x800d

1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:13, mod PSEQ:0
TAV:0

PCU-SYNC frame detected
Block number : 4
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000010001 0x8011
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:17, mod PSEQ:4
TAV:0

PCU-SYNC frame detected
Block number : 5
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000010101 0x8015
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:21, mod PSEQ:8
TAV:0

PCU-SYNC frame detected
Block number : 6
0000000000000000 0x0000
1111100010111111 0xf8bf
1111111110000000 0xff80
1000000000011010 0x801a
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff

1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
1111111111111111 0xffff
PSEQ:26, mod PSEQ:0
TAV:0

11.5GSL interface activation

send RPRSTARTGSL to RGRLCR
send: the signal was sent.
no monitor -long send MRSEIZEGSL<63245> from
RGRLCR:RGRLCR to RGRLCR:RP_DSPSUP
lpdchInd = 132
dspNumber = 4
macPid = 65759
no monitor -long send MRSEIZEGSLR<63246> from
RGRLCR:RP_DSPSUP to RGRLCR:RGRLCR
lpdchInd = 132
resultCode = 0
no monitor -long send MRACTIVATEGSL<63233> from
RGRLCR:RGRLCR to RGRLCR:RP_DSPSUP
lpdchInd = 132 (Note: Channel 132)
dspNumber = 4 (Note: DSP 4)

11.6GSL interface deactivation

send RPRSTOPGSL to RGRLCR
send: the signal was sent.
no monitor -long send MRRELEASEGSL<63242> from
RGRLCR:RGRLCR to RGRLCR:RP_DSPSUP
lpdchInd = 132
dspNumber = 4
no monitor -long send MRRELEASEGSLR<63243> from
RGRLCR:RP_DSPSUP to RGRLCR:RGRLCR
lpdchInd = 132

11.7DSP synchronisation signals

no monitor -long send MRTXDATAREQINIT<63248> from RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901752
no monitor   send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4 to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901757
no monitor -long send MRTXDATAREQ<63247> from RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484
txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484
txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484
txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29
txBuffer[19] = 1213
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17
releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23

chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901761
no monitor -long send MRTXDATAREQ<63247> from
RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484
txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484
txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484
txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29
txBuffer[19] = 1217
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17

releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23
chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901765
no monitor -long send MRTXDATAREQ<63247> from
RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484
txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484
txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484
txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29

txBuffer[19] = 1221
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17
releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23
chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901770
no monitor -long send MRTXDATAREQ<63247> from
RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484
txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484
txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484

txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29
txBuffer[19] = 1226
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17
releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23
chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901774
no monitor -long send MRTXDATAREQ<63247> from
RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484
txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484

txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484
txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29
txBuffer[19] = 1230
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17
releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23
chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRTXDATAREQINIT<63248> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
fnDl = 1901778
no monitor -long send MRTXDATAREQ<63247> from
RGRLCR:RP_MAC_000 to RGRLCR:RP_DSPSUP
txBuffer[0] = 553
txBuffer[1] = 54484
txBuffer[2] = 54484
txBuffer[3] = 54484

txBuffer[4] = 54484
txBuffer[5] = 54484
txBuffer[6] = 54484
txBuffer[7] = 54484
txBuffer[8] = 54484
txBuffer[9] = 54484
txBuffer[10] = 54484
txBuffer[11] = 54484
txBuffer[12] = 54484
txBuffer[13] = 54484
txBuffer[14] = 54484
txBuffer[15] = 54484
txBuffer[16] = 54484
txBuffer[17] = 0
txBuffer[18] = 29
txBuffer[19] = 1234
lpdchInd = 132
msInd = 65535
confType = 0
recInd = 17
releaseCause = 0
resType = 0
bsn = 0
msgSize = 0
drxPeriod = 23
chType = 0
timeStamp = 4705
psiId = 0
psiRepeatPeriod = 0
channelCodingScheme = 0
transmitType = 38
storedNextPoll = 1
imsiMod1000Mod704 = 4512
splitPgCycleCode = 164
nextMsg_p = DPPDATAIND
prevMsg_p = DPPDATAIND
no monitor send MRIRQIND<63241> from RGRLCR:RP_DSPINT_4
to RGRLCR:RP_DSPSUP
no monitor -long send MRGSLERRORIND<63240> from
RGRLCR:RP_DSPSUP to RGRLCR:RP_MAC_000
lpdchInd = 132
errorType = 1