RC9000-10

SW9911I-D

System Administrator's Guide

PN: 99203389

RC Computer

Keywords:
RC9000-10, RC8000, SSP, RC9310, FTS.

Abstract:
This guide is addressed to the system administrators of RC9000-10 systems. It contains information about installation and customization of system software, including the basic software for the integrated communication control units, as well as descriptions of administrative procedures in general.

Date:
Januar 1989

PN: 991-11-257

Table of Contents

1. Introduction

As the title suggests this guide is written for the RC9000-10 system administrators.

Before you begin

Before you attempt any of the installation and customization procedures outlined in this guide you must be familiar with the concepts of RC9000-10 basic software. If you feel more than a little unsure within this field, you are advised to consult Appendix A. *References*, which will suggest further reading.

This guide

This book is the system administrator's reference book. However, when you read it for the first time you are recommended to read the chapters in the following order:

When you have finished reading this chapter, you should continue with chapter 2. *Operator's Reference*, which is a useful presentation of the RC9000-10 system.

Then you should jump directly to chapters 10 and 11 about the System Support Processor, the SSP, since this module is probably completely unknown to you, but indeed most essential, particularly during system upstart.

Furthermore, Chapter 5 and 9 provide necessary descriptions about *Peripheral Devices* and *Integrated Communication Control Units*, respectively.

Finally you are strongly advised to spend some time thinking about the message in chapter 12. *The System Administrator's Three Commandments*. The time you spend on that chapter now, will be tenfold saved time later on.

The remaining chapters may be consulted as occasion requires.

Keeping Up-To-Date

You are strongly recommended to keep up-to-date with information about new releases of software and new documentation. The best way to

do this is by subscribing to the Support Handbook; contact your RC consultant.

Note
In case of inconsistency between software release Package Description and this guide, the most recent information will apply. In case of doubts, call RC Computer staff.

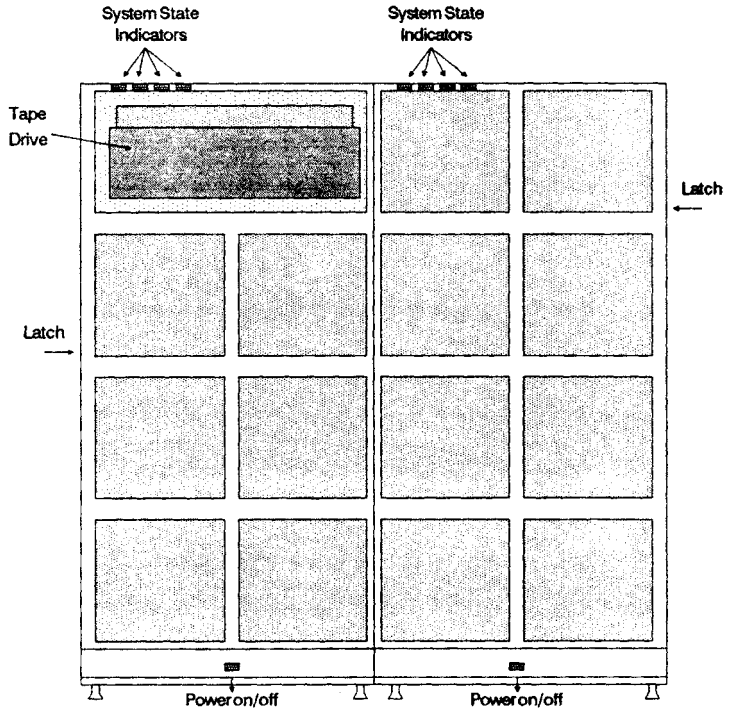
2. Operator's Reference

This chapter presents the modules of the RC9000 system. The drawings are only schematic and they are intended to serve as a frame of reference for succeeding chapters.

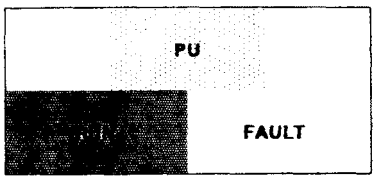
Included are also charts that explain the significance of the indicator lamps outside and inside the cabinet.

2.1 The Cabinet

The sketch below shows an RC9000C-L.



On the top of each of the two cabinets there are four system state indicators. One such indicator is shown below.



The following abbreviations are used:

PU	Processing Unit
CP	Communication Subrack
DCM	Disk Control Module
TCM	Tape Control Module

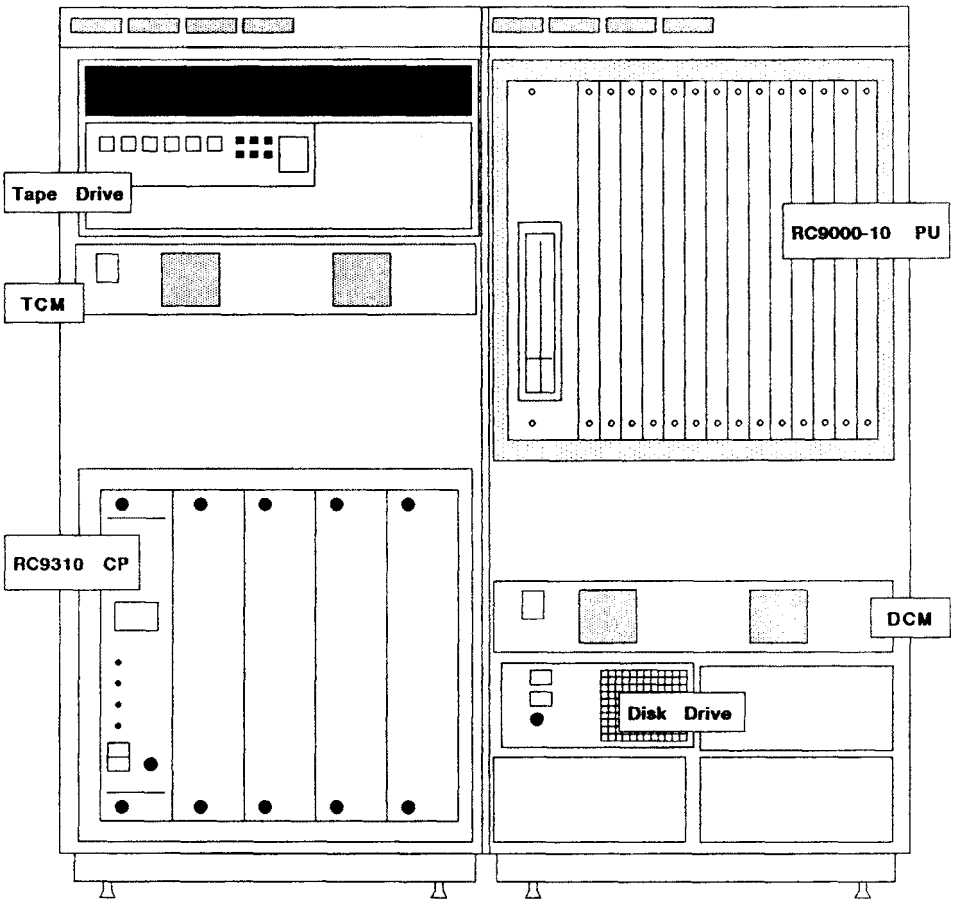
The indicator shows the state of the associated module, with (in this case) PU, RUN or FAULT. When the module indication (in this case PU) is lit, the module in question is powered on.

As a processing unit contains multiple boards, and a communication subrack may contain multiple communication control units, it is necessary to open the cabinet in order to identify the failing module.

The cabinet is opened by removing the front cover. Each front cover is held in place by two latches. The latches are located under the topmost 'tiles', at the rightmost and leftmost sides, respectively.

To remove the cover, press the latches upwards with your fingers.

The next sketch shows the RC9000C-L with the front covers removed.

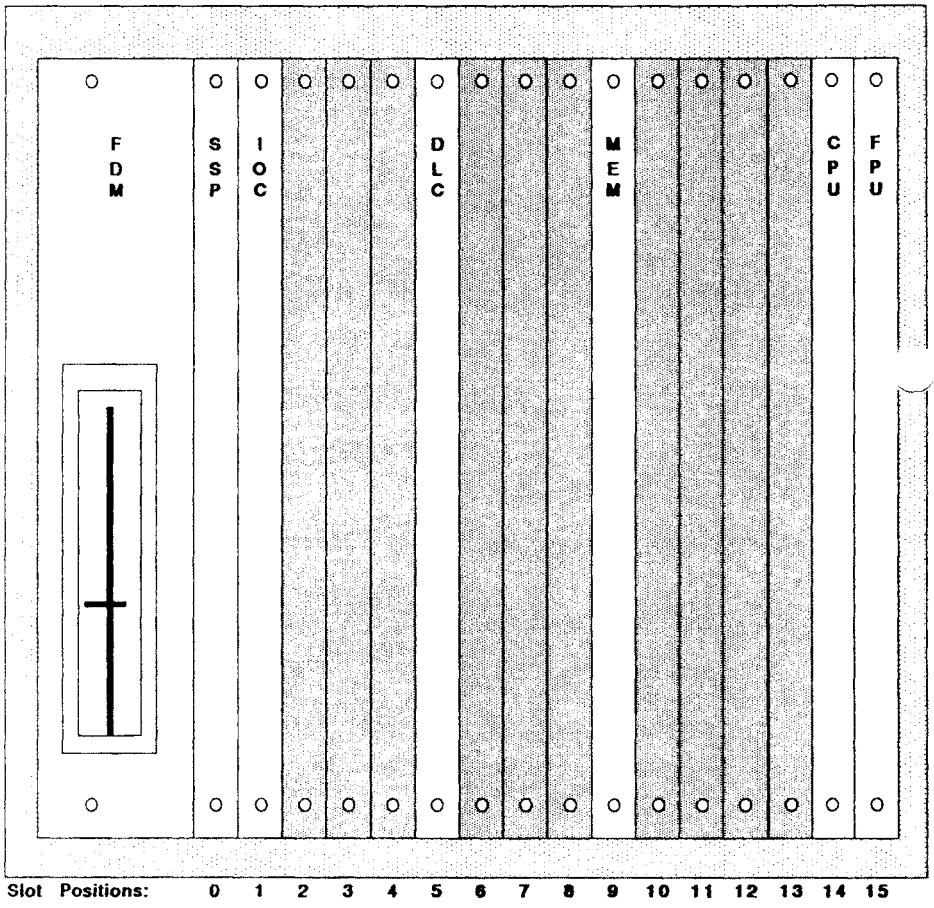


The cabinet to the left contains an RC9250 horizontal tape unit, an RC9220 tape control module (TCM) and an RC9310, mounted in a communication control unit mounting chassis.

The cabinet to the right contains an RC9000-10 processing unit, an RC9210 disk control module (DCM) and an RC9230 850 MByte disk drive.

2.2 The Processing Unit

The sketch below shows an RC9000-10 processing unit.



There are 16 slot positions in the processing unit. The numbering starts from the left side with 0 and ends with number 15 to the right.

To the left of the slots you see the diskette drive for the System Support Processor (here marked FDM, for Floppy Disk Module).

The processing unit contains the following boards:

SSP	The RC9140 System Support Processor
IOC	The RC9120 I/O Channel Controller
DLC	The RC9130 Dual LAN Controller
MEM	The RC9110 Memory Board

The RC8500 processor, consisting of two boards:

CPU	The Central Processing Unit
FPU	The Floating Point Unit

The board positions on the sketch are mandatory:

- The SSP must always occupy slot number 0.
- The slot numbers 1 through 4 may be used for IOC boards.
- The slot numbers 5 through 8 may be used for DLC boards.
- The slot numbers from 9 and upwards may be used for memory boards.
- The CPU/FPU boards must be placed from slot number 15 and downwards in such a way that the FPU boards always occupy slot positions with odd numbers (15, 13, ...) (The CPU and the FPU boards are interconnected by a small supplementary board, which in itself does not occupy a slot).

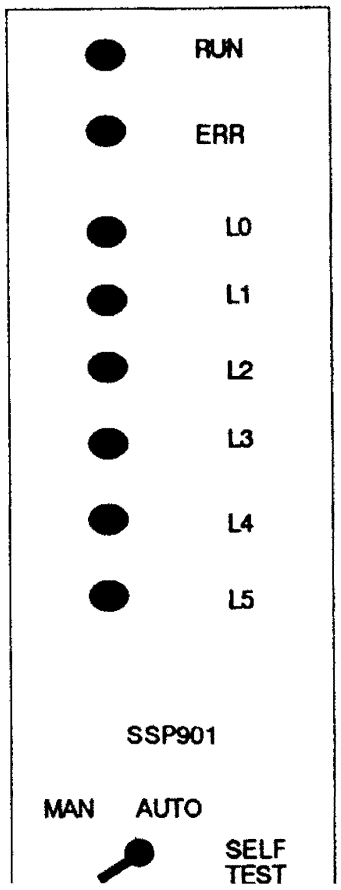
As mentioned, the board positions are mandatory; with one exception, however: memory boards can, if necessary, be interspersed in any empty slot. This will certainly be necessary in multi-processor configurations.

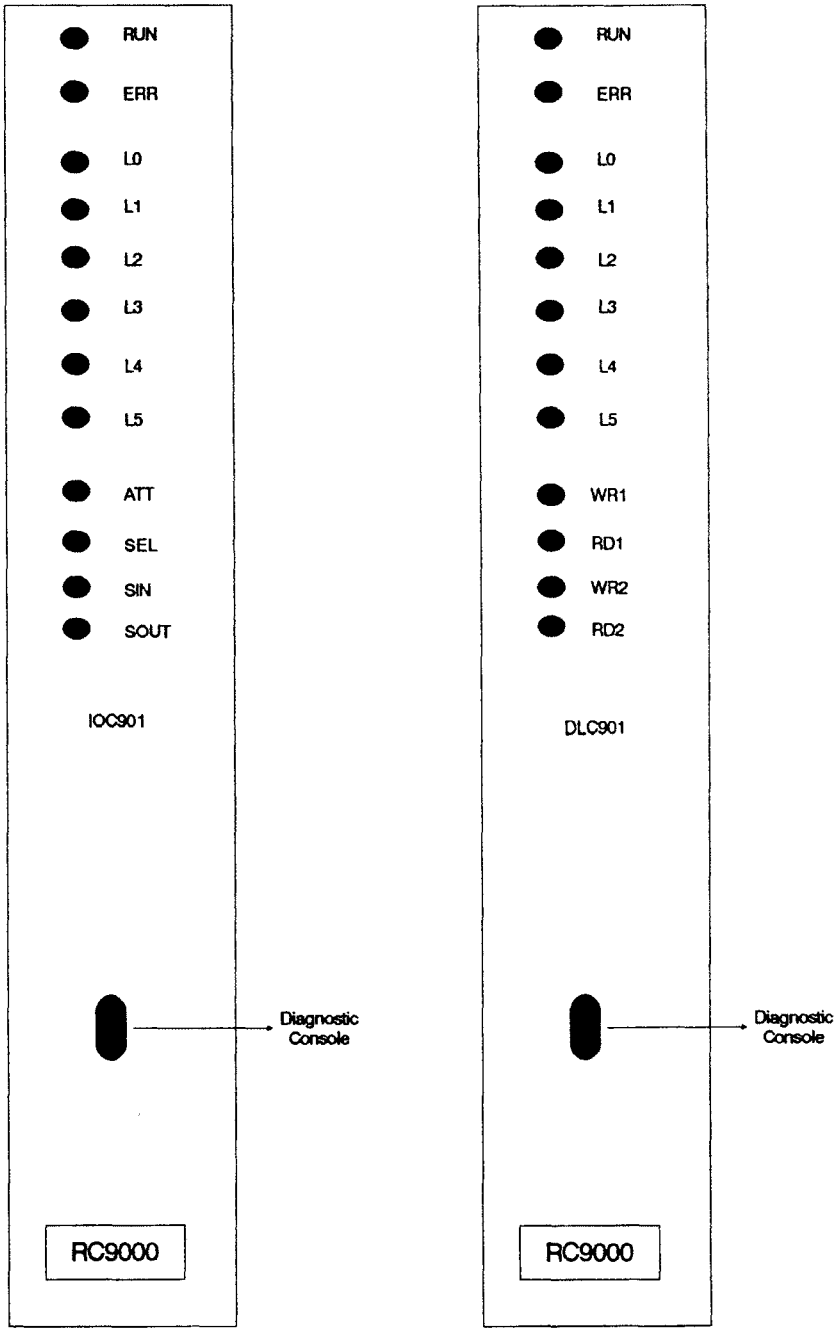
An *Error Indicator LED* is placed beneath each slot position. When the state indicator on the outside of the cabinet says PU FAULT, failing boards are identified by means of these indicators.

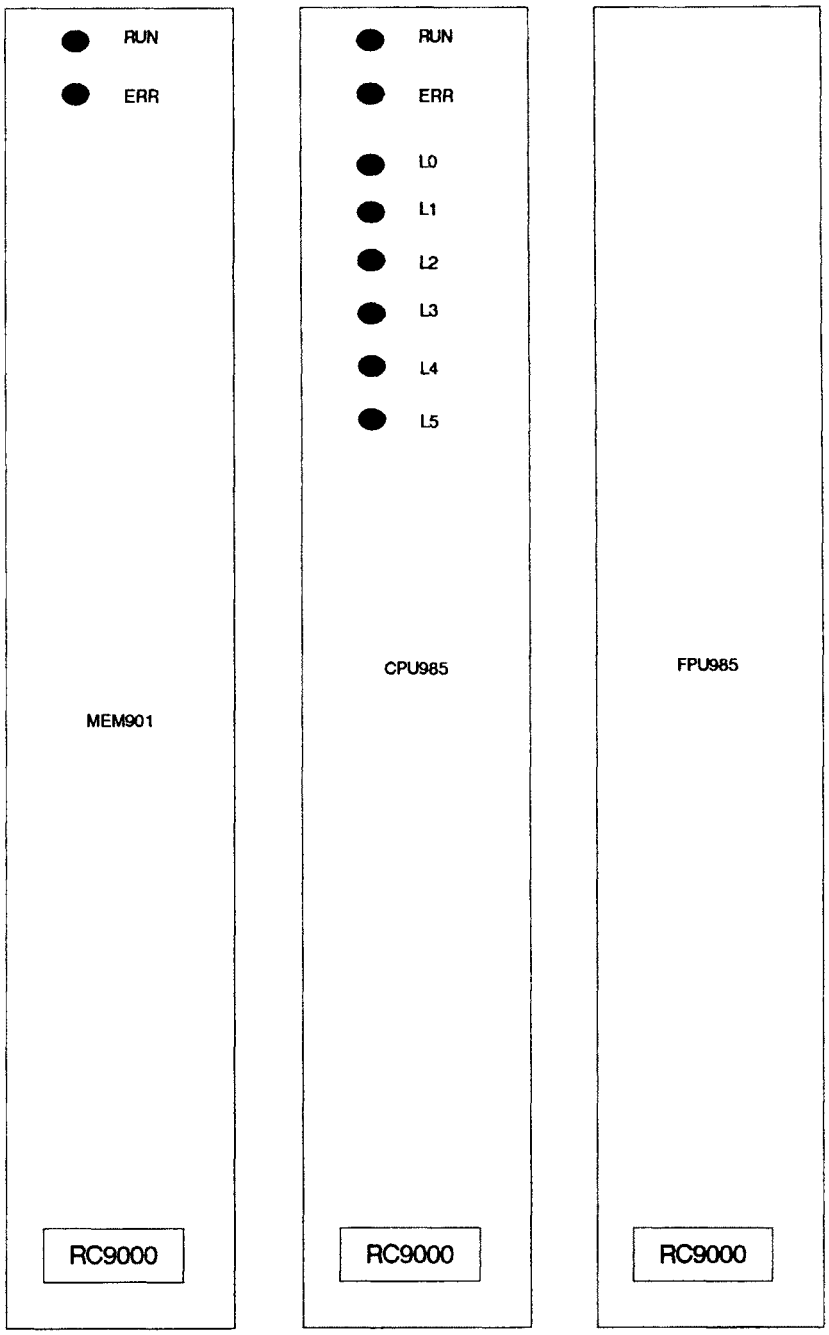
2.3 The Processing Unit Boards

This section shows the boards of the RC9000-10 as they appear when the front cover of the cabinet has been removed.

The following three pages contain sketches of the PU boards and charts that explain the significance of the lamps on the boards.


RUN
ERR
L0
L1
L2
L3
L4
L5
SSP901
MAN
AUTO
SELF
TEST


RUN
ERR
L0
L1
L2
L3
L4
L5
ATT
SEL
SIN
SOUT
IOC901
Diagnostic Console
RC9000
RUN
ERR
L0
L1
L2
L3
L4
L5
WR1
RD1
WR2
RD2
DLC901
Diagnostic Console
RC9000


RUN
ERR
MEM901
RC9000
RUN
ERR
L0
L1
L2
L3
L4
L5
CPU985
RC9000
FPU985
RC9000

RC9000-10 PU LAMPS

<table>
<tr><th></th><th>SSP</th><th>IOC</th><th>DLC</th><th>CPU</th><th></th></tr>
<tr><td>L0</td><td colspan="4">— TESTING —</td><td>L0</td></tr>
<tr><td>L1</td><td colspan="3">— KERNEL ERROR —</td><td>MONITOR ERROR</td><td>L1</td></tr>
<tr><td>L2</td><td rowspan="2">STATE INDICATOR</td><td rowspan="4">LOAD INDICATOR</td><td rowspan="4">UNUSED</td><td>UNUSED</td><td>L2</td></tr>
<tr><td>L3</td><td>EXECUTING INSTRUCT'S.</td><td>L3</td></tr>
<tr><td>L4</td><td rowspan="2">UNUSED</td><td>ACTIVE</td><td>L4</td></tr>
<tr><td>L5</td><td>WAITING FOR MONITOR</td><td>L5</td></tr>
</table>

SSP STATE INDICATOR

	L3 off	L3 on
L2 off	initialization state	operating state
L2 on	SSP Controlled State	power down

The charts above describe how to decode the information provided by the lamps labelled L0 to L5 on each board.

When L0 is lit (meaning that the boards is being tested), the other lamps have no significance.

After the test phase, the lamps L1 to L5 provide the information shown in the first chart.

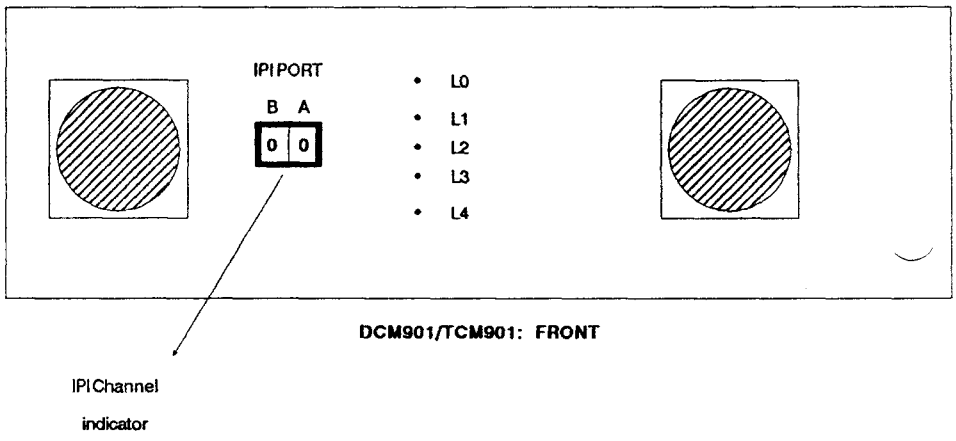
In the case of the IOC, the lamps L2 to L5 constitute a *load indicator.* The four lamps actually form a bar chart that indicates the load on the IOC board.

The lamps L2 and L3 on the SSP indicate the state of the SSP, as explained by the second chart.

The remaining lamps on the IOC/DLC are hardware dependent and are not of interest to the general user.

2.4 Disk And Tape Control Modules

The sketches below show the disk and tape control modules. The chart explains the significance of the 5 indicator lamps on the front of the control units.



DCM And TCM Lamps

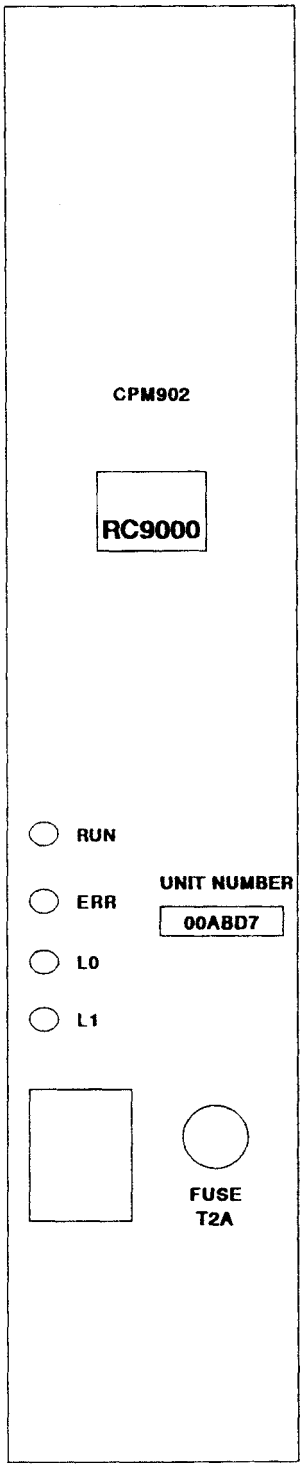
	DCM901	TCM901
L0	*Activity on device 0*	*Activity on device 0*
L1	*Activity on device 1*	*Activity on device 1*
L2	*Activity on device 2*	*Corrected Read Error*
L3	*Activity on device 3*	*Read/Write Retry*
L4	*Read/Write Data I/O*	*Read/Write Data I/O*

The two control modules have a similar appearance. The *IPI Channel Indicator* tells which device number the control unit has as a slave on the I/O channel. The units have two interfaces towards the channel: A and B.

The chart of the lamps explains the significance of the lamps numbered L0 to L4.

2.5 The RC9310 Communication Control Unit

The sketch shows an RC9310 unit. The chart overleaf explains the signifinance of the lamps RUN, ERR and L0.



The label beneath the UNIT NUMBER inscription will contain a six digit hexadecimal number. This number is used for making the Ethernet address and the FTS user name of the unit. The unit number is also present in the unit's PROM. The number on the label and the number in the PROM must always be the same - cf. chapter 9 for details and significance.

Meaning of RC9310 Lamps	RUN	ERR	LO
Selftest in progress			
Stopped after selftest error			
Load/Start in progress			
Fault in load/start phase			
Normal operation			
Fault during operation			
A Shaded Box Means That The Corresponding Lamp Is Lit			

3. System Installation and Customization

This chapter describes how to install the SW9890I basic system software and the SW8010I System Utility package, how to initialize the disks, and how to perform a basic customization of the system.

Note
that a new system delivered to you will already contain the basic system software, and you will not need to perform the installations and customizations as described in this chapter.

Nevertheless, the procedures are included in this guide, since you may wish to make minor adjustments or - for example - initialize a new disk.

The SW9890I SSP diskette
contains the basic software, i.e. micro programs for the central processing unit and the floating point unit, programs for the System Support Processor (SSP), the LAN Controller, the I/O Controller, the Monitor and 's' operating system.

The SW9890I magnetic tape
contains the File Transfer Service (FTS) and the *lanstat* program.

The SW8010I tape
contains the System Utility package. This package must be installed **before** you install the **SW9890I magnetic tape**.

If your system consists of more than one PU, you will need the corresponding number of diskettes, whereas you will only need one tape.

If you have not read Chapter 2, *Operator's Reference*, and Chapters 10 and 11 about the SSP, it will be useful to go through these chapters before you begin to initialize the system.

3.1 SW9890I Diskette, Installation and Customization

In this section the following conventions will be used:

--> means **select**
[send] means **execute the menu by pressing [send]**
[DEL] means **abort the menu by pressing [DEL]**

The character keys stated refer to an RC45 terminal. Please, refer to Appendix B. if you need more detailed information.

Installation

Insert the SW9890I diskette in the diskette drive, and set the SSP switches:

* **Selftest switch**		**> AUTO**
* **Bootmedium switch**		**> AUTO**
* **PU test**		**> AUTO**
* **PU load**	**MAN <**	
* **SWT5**	**MAN <**	
* **SWT6**		**> AUTO**
* **SWT7**		**> AUTO**
* **SWT8**		**> AUTO**

Power on the SSP terminal.

Power up the system by means of the ON/OFF switch at the foot of the cabinet or by means of the special external power supply.

The SSP will now perform various tests and validations and then stop in manual PU load state. The terminal will display the `Console select menu`.

Customization

The system customization is performed by means of the SSP menu system, cf. chapters 10 and 11.

Check that the SSP shares your own interpretation of the module configuration by:

--> S - SSP console
--> D - Display utilities
--> M - Display module status

If the module configuration is correct:

[DEL] [DEL] which will take you back to the main menu.

--> C - Customization utilities
--> A - Set module configuration to actual configuration

--> P - Set PU name

In this menu you insert the appropriate values for:

PU Name
The input field can hold up to 10 characters, but only 8 characters is recommended. The default value is the SSP serial number. The PU name is used to identify the system in various connections, e.g. to identify messages on the system terminal and in the log. **[send]**.
The SSP will now display the message `PU name set`.

Now you are back in the `Customization utilities`.

--> 1 - Parameters for model 10
--> B - Basic system parameters
The menu will display the default values, which are also described in the Package Description. There is probably no need to change these. If you do, the SSP will check whether there are sufficient resources for the customization in question, and you will be informed when you execute the menu.

Please refer to Ref. [11] and [12] for relevant information.

If you change the values: **[send]**.
Otherwise: **[DEL]**

You are now in the `Parameters for model 10` menu.

--> 1 - Parameters for LAN controller no. 1
--> D - Set device handler parameters

If you change the values: **[send]**.
Otherwise: **[DEL]**

--> I - Set IMC parameters
If you change the values: **[send]**.
Otherwise: **[DEL]**

In both of the last two menus you can probably use the default parameters. You will find details in the Package Description and in section 5.2.

Each LAN controller must be customized individually.

Press **[DEL]** until your are back in the `Customization utilities` menu.

--> T - Timezone and daylight saving

Time zone:
A numeric value in the range of -720 - 720, which indicates the number of minutes west of UTC, Universal Time Coordination. Default is Middle European Time (MET = -60). This information is used when displaying messages on the system terminal and is passed to the operating system. **[send]** or **[DEL]**

Daylight Saving Time:
Indicates whether daylight saving time is in effect. Legal values are YES or NO. Default is NO. Note that the value will not change automatically, so the system administrator should change this value whenever daylight saving time become effective or inoperative. **[send]** or **[DEL]**

--> C - Real time clock

Check the correct time and **[send]** or **[DEL]**.

Return to the `Customization Utilities` menu by means of **[DEL]**.

--> W - Write Customization data to diskette

and **wait** till the message `'customization file written'` appears. **[DEL]**.

Back-up your customization data !

--> E - Exchange diskette

Remove the diskette and insert the back-up diskette.

--> C - Customization utilities
--> W - Write customization data to diskette

Remove your back-up diskette and insert the first diskette.

Set the SSP switch PU load > AUTO

Reset the SSP with a pointed object.

The SSP will now display the console select menu, and pick the operating system console as default. The rest of the initialization and customization will take place under the control of the operating system.

You can always change the above-mentioned customization, and if you do so, it will take effect after system reload.

3.2 Disk Initialization

Note that the disks included in a system delivery have already been initialized. You should therefore only initialize a disk in case you install a new disk.

The disk initialization consists of

- creating a link to the physical disk
- partitioning the disk into a number of logical disks: one disk describing part and a number of data disks
- naming these disks and
- establishing a main catalog on the system disk,

all of which is done by means of s-commands.

Note that the subsequent initialization is decribed be means of examples, which - if they are followed - should be employed with care.

See dppendix E.5 for the full syntax of the s-commands and ref. [4] for detailed explanations of the commands.

Creating a link to the physical disk

If you want to create a link from disk number 2 connected to controller (DCM) number 0 on the first I/O channel (iocmain1) to the external process with device number 60, the call should be made like this

```
createlink iocmainl disk unit02.60 0 2
```

```
disk 0 2 linked to 60
```

Partitioning the physical disk

The disk must be partitioned into areas (logical disks) that constitute a multiplum of the number of segments per track. This number varies according to the type of disk.

In the examples below we have used an RC9230 which has 46 segments per track, 15 tracks per cylinder and 1379 cylinders. The disk has a total of 1379 * 15 * 46 = 951,510 segments.

The following command will partition the disk into one disk describing part (61) with 46 segments and four logical data disks (62 - 65) each with 237866 segments.

```
initkit 60 46 61 237866 62 237866 63 237866 64 237866 65

linkall 60
```

```
logical disk linked on 61
logical disk linked on 62
logical disk linked on 63
logical disk linked on 64
logical disk linked on 65
```

Making catalogs on the logical disks

The following commands will establish a catalog on each data disk:

```
kitlabel  62  disk   catdisk   131  138
kitlabel  63  disk1  catdisk1  131  138
kitlabel  64  disk2  catdisk2  131  138
kitlabel  65  disk3  catdisk3  131  138
```

New main catalog

If you wish to create a new main catalog, the following example will establish a main catalog of 510 segments:

```
maincat catalog 510
```

```
maincat size:510 partitions:1 keys:510
```

Testing

Now you should test that the above procedures have been executed correctly by means of `kitoff`, `unlink` and `linkall`:

```
kitoff 62
kitoff 63
kitoff 64
kitoff 65

unlink 61 62 63 64 65

linkall unit02
```

```
logical disk linked on 61
disk   mounted on 62
disk1  mounted on 63
disk2  mounted on 64
disk3  mounted on 65
```

Updating the startarea file

If you already have a startarea file you should remember to update it. In this case:

```
createlink iocmain1 disk unit02.60 0 2
linkall unit02
```

3.3 The System Utility Package SW8010I

Confer with the SW8010I Package Description, before you begin installing the tape. Appendix E.3 provides a useful list of the legal modekinds.

Mount the tape with the System Utility software.

Create a link to the tape station connnected to controller (TCM) no. 7 on iocmain1 by the following command:

```
createlink iocmain1 tape to.10 7 0
```

Load the tape by the following s-command:

```
call 10 systemtape
binin mtl6 systemtape  2  4
```

3.4 The SW9890I FTS Tape

Note that the tape must be installed **after** installation of the System Utility package. Please consult the SW9890I Package Description.

3.5 Creation of a Startarea File.

Now, you should create a startarea file, which is executed automatically during autoload.

Note that programs started by the startarea file uses the SSP console as terminal, so some operations can only be performed from the SSP console.

The file must contain the commands that should be executed at autoload time.

The file can contain both s-commands and FP-commands. If, however, your file contains both kinds of commands, all s-commands must preceed all FP-commands, and lines with s-commands must start with an "*". The last s-command before an FP-command must be '* unstack'.

Lines commencing with ';' are comments.

The following jobfile *startarea* is an example of a startjob that will start up a system with one disk drive, one tape station and one printer. Furthermore, it will generate core dumps in three generations and start up FTS, TAS, PRIMO, and user application xxxx.

Note that after having created this file, the **SWT5 switch** on the SSP should be set to **AUTO** and it should stay in this position from now on.

```
; jobfile startarea for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; making links to peripheral devices (disk, tape printer, etc.)
; save coredump
; save log for snoop

; create a link to the physical disk
*createlink iocmain1 disk unit01.50 0 1
*linkall unit01
*lock

; create a link to the tape station
*createlink iocmain1 tape mt00.10 7 0

*all initproc size 100000       ; create process size 100000 hw
*i 4 startarea                  ; take fp-input from this file
*run unstack                    ; start process
;
;
; dump core etc. for error analysis saved in 3 generations
clear user coredump3            ; remove old 3rd generation
rename coredump2.coredump3      ; 3rd generation
rename coredump1.coredump2      ; 2nd generation
coredump1=set 900               ; 1rst eneration
scope user coredump1
;
; logareal for errorsnoop, saved in 3 generations
clear user rclogarea3           ; remove old 3rd generation
rename rclogarea2.rclogarea3    ; 3rd generation
rename rclogarea.rclogarea2     ; 2nd generation
rclogarea=set 108               ; 1rst generation
scope user rclogarea

replace startinput              ; start various processes
```

```
; jobfile startinput for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start all processes and set dump
*prepare coredump1              ; set new dump area
*read makelinks                 ; links to printer etc.
*read startserver               ; fts-server
*read startuser                 ; fts-user
*read starttas                  ; tas
*read startprimo                ; primo
*read startappl1                ; application no 1 (user application)
*read startappl2                ; application no 2 (user application)
*unstack                        ; unstack s-stack
```

```
; jobfile startserver for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start fts-server
*new ftsserver
*base -8388607 8388605
*function 1 2 3 4 5
```

```
; jobfile startuser for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start fts-user
*new ftsuser
*base -8388607 8388605
*function 1 2 3 4 5
*size 100000
*buf  8
*area 8
*perm disc  1000 10
*perm disc1 1000 10
*perm disc2 5000 50
*i 4 startuser
*run
*unstack

bftsuser incno.4                ; 4 incarnations
finis
```

```
; jobfile starttas for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start of Terminal Access System
*new tas
*base -8388607 8388605
*buf 10
*area 5
*size 35000
*perm disc 0 2
*function 0
*prog fp
*i 4.starttas
*run

*new menu
*base -8388607 8388605
*buf 55
*area 27
*size 100000
*perm disc 0 0
*function 0
*prog tasterm
*run
*unstack

tascat  ; catalog.cattxt
```

```
; jobfile startprimo for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start of primo
*new primo
*size 19942
*temp disc 1000 10
*buf 19 area 13
*base -8388607 8388605
*function 1 2 3 4 5
*prog bprimo
*run

*unstack
```

```
; jobfile startappl1 for rc9000 "einstein"
; last revision made by Albert 1988.12.12
; start of user application "xxxx"
*new xxxx
*base xx xx
*function x x x
*size xxxx
*buf  xx
*area xx
*temp disc xx xx
*perm xxxx xx xx
*perm xxxx xx xx
*i 4 startappl1                          ; take fp-input from this file
*run
*unstack

message start af bruger applikation xxxx    ; operator message
xxxx xxxx.xxxx xxxx.xxxx                    ; call of program
message bruger applikation xxxx stoppet     ; operator message

finis
```

The system customization is now complete, and you may now install the other software you want to use.

This is done according to the package description of the software package in question.

You are referred to chapters 6, 7, and 8, which describe the customization of FTS, TAS, SOS and PRIMO, respectively, and chapter 9 which describes the customization of the RC9310 Communications Control Unit.

4. Day-to-Day Administration

This chapter is a practical description of the startup and shutdown of the system. It is supposed that the system has been initialized and customized as described in chapter 3. *System Installation And Customization.*

4.1 Start-Up Procedure

The subsequent routines can only be performed if:

- the configuration of the system is unchanged
- the basic software is present on the disk

1) Power on the the SSP terminal.

2) Set all SSP switches to AUTO position.

3) Turn on the power to the system. Shortly after the external lamps on the cabinet will indicate that the system is powered on.

4) A few moments later the `Console select menu` will appear on the screen.

5) After a couple of minuts the following will be displayed on the console:

```
monitor release: <versionid>
monitor version: <yy> <mm> <dd> <hh> <mm> <ss>
date of options: <yy> <mm> <dd> <hh> <mm> <ss>

first physical disk linked to <devno>
logical disk linked to <devno>
disc1 mounted on <devno>
```

6) The startarea file (cf section 3.5) will be read and executed. According to the design of the startarea file in question and the hardware configuration, a number of messages, specific for each installation, will be displayed. The system will now be ready for use.

If you encounter errors during these procedures, please call RC Technical Staff.

4.2 Daily Close-Down

This section describes how to perform an orderly shut-down of the operating system s and the processes under s, TAS, SOS, PRIMO and FTS.

The description will include how to remove processes and operating systems in a controlled way, and finally how to perform an eventual system power-off.

4.2.1 Close-down of TAS

The operator can prevent further log-in by the command:

```
att tas
stop login <group>
```

If you state a number in the parameter <group>, only users belonging to groups with lower numbers can log-in. If no parameter is stated, all users will be prevented from log-in.

The command:

```
att tas
stop system <min>
```

will stop TAS after the number of minutes stated in <min>. Each minute a shut-down message will be displayed on all terminals in use, until the stated period of time has expired. If <min> is stated as 0, TAS will stop immediately.

The command:

```
att tas
stop system check
```

in stead of <min> causes TAS to stop when the last user logs out. When this has happened a stop message will be displayed at the system console.

When TAS has stopped, all terminals linked to applications will be removed, and TAS processes will also be removed. This will not affect other processes in the system; the remaining processes must be closed down as described below.

4.2.2 Close Down of SOS and PRIMO

PRIMO

PRIMO is closed down by removing the process with the s-command:

```
att s
proc primo remove
```

SOS

When closing down SOS the system should first be drained by using the lock-command

```
att sos
lock <operator key>
```

The <operator key> is the operator password which can be defined during customization, and will be required to allow operator intervention.

The lock command will make SOS refuse all attempts to create jobs or to connect terminals to multiterminal jobs. When the last job quits, SOS will write a message on the system terminal, telling the operator that the system is empty.

Now the system can be closed by issuing the halt-command:

```
att sos
halt <operator key>
```

Halt will cause an immediate stop, without removing active jobs, and this is why it is important to drain the system first.

Now SOS can be removed by the s-command:

```
att s
proc sos remove
```

Finally, write the errorlog and discstat files out in a file on backing storage.

4.2.3 Closedown of FTS

You should stop all ftsservers (one per LAN):

```
att ftsserver
stop
```

The commands are repeated for each ftsserver.

4.2.4 Closedown of the system

Close down the system with the following s-commands:

Remove all processes with `cleanup`.

Close all catalogs by means of `kitoff`.

Close down all the logical disks with `unlink <no>`.

Remove all links to the physical disks with `removelink <no>`.

You may now power off the system.

4.3 Back-Up And Restore

File system backup and restore is described in ref. [9], *Save, incsave; Load, incload.* However, the program calls and relevant examples have been included in Appendix E.

5. Peripheral Devices

The descriptions of peripheral devices which are maintained by Monitor are called *external processes*. This is because the basic mechanisms used for interaction between programs (internal processes) and devices (external processes) are the same as those used for interaction between internal processes, in other words: to a running program in a process a peripheral device looks just like any other process.

Messages sent to external processes are intercepted by Monitor and redirected to *device handlers*. Device handlers are the software modules which run on the controllers in the PU and are responsible for controlling device functions. Device handlers do not exercise direct control over devices, but work in cooperation with additional controller software.

Device handlers for disk and tape devices run on the I/O channel controller and cooperate with disk and tape control modules attached to the channel. Device handlers for LAN devices run on the LAN controller and (in most cases) cooperate with communication control units connected via the LAN.

Monitor has a table of device descriptions, the size of which may be customized in the SSP menu `Model 10 Basis System Parameters (F04)`. The indices into this table are called *device numbers*.

When an external process is created, an unused device description is allocated to it and a *link* set up to a device handler on a controller. Note that whenever an external process exists it has a link to a device handler. For most practical purposes 'external process' and 'link' are therefore synonymous terms, and often it is most convenient just to speak of a link.

External processes may be referred to by device number or by name. If you know the number of a device, you can assign a name to it by means of the s-command `call`, or from a program by a monitor call.

A limited set of external processes are always created when Monitor is started, most of them with fixed device numbers and names:

```
no name      description

2  console1  system console attached to SSP
20 sspmain   main process for the SSP (as controller)
21 lanmain1  main process for 1st LAN controller
22 lanmain2  main process for 2nd LAN controller
23 lanmain3  main process for 3rd LAN controller
24 lanmain4  main process for 4th LAN controller
28 iocmain1  main process for 1st I/O channel controller
29 iocmain2  main process for 2nd I/O channel controller
30 iocmain3  main process for 3rd I/O channel controller
31 iocmain4  main process for 4th I/O channel controller
```

Furthermore, a number of external processes representing the system disk is created; one process for the physical disk and one for each of the logical disks. The names and numbers for the system disk processes are not fixed. See also section 3.2, *Disk Initialization*.

There is a main process for each controller. It handles supervisory functions such as creation and deletion of links to individual devices. The main processes are only created for those processor modules which are present in the PU. So if, for example, your system only has one LAN controller, only `lanmain1` is created and similarly for I/O channel controllers.

All links beyond those mentioned above must be created explicitly, either as permanent links which are typically created by calls in the startarea file (cf. section 3.5, *Creation of a startarea file*), or as temporary links which are created when needed. The subsections below give details on how this is done for channel and LAN devices, respectively. Refs. [1] and [2] provide more detailed descriptions of these external processes.

In most cases the device number of an external process is not important, as references are made by name. However, some programs, in particular BOSS, have the habit of referring to certain devices by number. For these devices, the appropriate numbers must therefore be specified when the links are made. It is a common convention that magnetic tape stations have device number 10 (and 11, if there is more than one), and that a system printer has device number 5.

When a specific device number is not requested for a link, Monitor will select the unused device description with the highest number. When a specific number is required, it is therefore most practical to choose a small number to avoid hitting an already occupied device description.

5.1 Channel Devices, Disk And Tape

Links to channel devices are manipulated by means of s-commands.

As the maximum configuration of control modules that can be attached to a channel is fixed, no customization is necessary for the channel controller.

Normally programs do not access disks directly via their external processes, but use files which are accessed via area processes. The

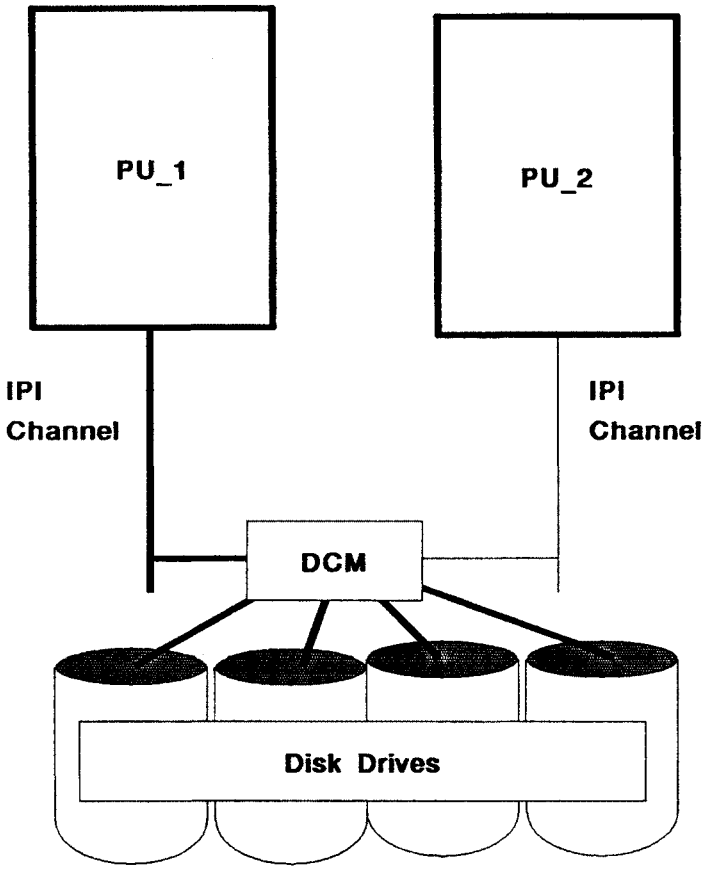
mapping to external processes is then performed implicitly by Monitor. However, to make file access possible, external processes must be created for both physical and logical disks and the catalogs on the logical disks must be included in the backing storage system.

This requires two s-commands in the startarea file for each physical disk: `createlink`, which creates the link to the physical disk, and `linkall`, which does the rest.

Tapes are usually read and written by means of utility programs which assume that an external process exists and require the name or number of the external process as a parameter. It is therefore common practice to create a link to each tape station in the startarea file. (See also section 3.5).

5.1.1 Dual ported control modules

In system installations with more than one RC9000-10 PU, disk and tape control modules may be attached to two channels, allowing two PUs to access the same devices. Such a dual-ported configuration may be used for purposes of device sharing as well as backup access.



Dual Porting Of Disk Control Module

To support the administration of configurations with dual porting, two states of a device (disk or tape station) are distinguished by Monitor, the I/O channel controller, and control modules: *reserved* and *neutral.*

Reserved: The control module will only process commands received on the channel for which the device is reserved. Disks with catalogs and tape stations should always be reserved.

Neutral: The control module will process commands from both channels. In this way an application which is distributed on the two PUs may access the device via both channels. Note that the application is responsible for the integrity of the data, as each Monitor has no means to control the commands sent by the other PU on the other channel.

The `createlink` s-command
The device state is controlled by a parameter to `createlink`. For a disk this parameter can take three values: `diskneutral`, `disk` or `diskforce`.

In the normal case, when the value `disk` is specified, the command will be rejected if the device is already reserved. However, if its state is `neutral`, the link will be created and the device reserved for the PU that creates it.

The `diskforce` specification is stronger still. It will cause the device to become reserved for the PU where the call is made, even if it was already reserved for the other PU. This possiblity is provided as a means to break a reservation made by a PU which later fails. Suppose PU B is installed as backup for PU A. A jobfile, similar to a startarea file, should then be prepared to be executed on PU B to restart the application in the event that PU A fails. For each relevant disk that would normally be reserved by PU A, such a jobfile should contain a `createlink` command specifying `diskforce`.

The above-mentioned explanation of the parameter values for a disk, also applies for a tape station.

5.2 LAN Devices

When an RC9000-10 PU is attached to a local area network by means of a LAN controller it will usually make certain services, viz. the CSP host function and an FTS server, visible on the LAN. With respect to these services the PU is known by a *host name.*

Normally the host name is the PU name which can be customized in the SSP menu `Set PU Name (F00)`. It is possible to specify a host name different from the PU name. This can be done for each LAN controller in the SSP menu `Model 10 LAN controller parameters (F06)`. In general this is not recommended. However, if a PU is attached to the same LAN through two or more controllers, it is necessary to use different names.

Links to LAN devices may be created and removed by means of the utility programs `makelink` and `deletelink`. However, this is not

necessary in all cases, as some programs create the links they need by sending requests to the main process for the controller in question. Links for CSP terminals are created in a special way (see below).

The remaining part of this section contains information of relevance to system administration for each type of LAN device. All customization for these devices pertains to the LAN controller and is therefore done in the SSP menu `Model 10 LAN Controller Parameters (F06)`.

5.2.1 CSP devices

CSP devices are terminals, printers, and other devices which are connected to asynchronous V.24 ports. These devices are individually connected to device handlers, i.e. there is an external process for each device.

The LAN controller distinguishes three types of CSP devices:

- CSP terminals,
- CSP printers,
- consoles/nologin lines.

Every CSP device has a 10-character *device name* which uniquely identifies it on the LAN. Device names are formed by combining a part which identifies the control unit with a number or letter which identifies the individual device. See chapter 9 for a description of device names for RC9310 and RC9330 devices.

Links for CSP terminals are created automatically when the terminals become connected to the RC9000-PU. The name of an external process created in this fashion is the terminal's device name prefixed by the digit '0'.

Links to CSP printers and consoles/nologin lines must be created by the standard mechanisms, usually by calling `makelink` from a startarea file. When the link is created, the name of the device must be specified as a parameter which is passed to the device handler. The name of the external process is defined independently. The device handler will then wait for the device with the specified name to become connected. This will happen when the device control unit which controls the device is started (cf. chapter 9). At this time the link and the device connection are joined, and the link may be used for sending or receiving data to or from the device.

Unless a link with the correct device name has been created, the LAN controller will reject a connection from the device.

Apart from the way links are created, i.e. with respect to their interface to internal processes, nologin lines work just like CSP terminals.

The number of device handlers for each type of CSP device must be customized. Three additional parameters which concern CSP devices may also be customized:

- The interrupt character for CSP terminals, i.e. the character used to generate an attention to an internal process. The character may be

specified by the decimal value of its ASCII code. The default is 27 (ESC).

- The number of input buffers to be shared among CSP terminals operating in canonical mode. These buffers are used for queued input, i.e. data that has been typed ahead of prompting.

- The deselection timer for CSP printers, i.e. how many seconds the printer may be deselected (local), e.g. to change paper, before the condition is treated as a fault.

5.2.2 3270 devices

The 3270 input and output device handlers provide the interface which is required by the FORMAT8000 terminal access procedures. These device handlers always operate as pairs: input and output. A device handler pair expects to be reserved by an internal application process which uses the FORMAT8000 procedures and to be activated with the name whereby the application shall be known to 3270 CUs. The device handler pair will then automatically maintain connections to all 3270 CUs on the LAN, including front-end units which communicate with remote CUs.

The number of 3270 device handler pairs that shall be running on the LAN controller and the number of 3270 CUs to which they shall make connections when they have been activated are specified as customization parameters.

You may also customize the 3270 input timeout value. Input messages that have been received by the LAN controller from a 3270 terminal, but not yet read by the application process, are monitored with a timer. In case of a timeout, the input message is discarded and the message `'Host timeout, data lost'` is shown in the status line of the terminal. The best value for this timer depends on the application's response time profile. A timeout should occur when and only when the application process is not providing normal service.

5.2.3 IMC port device

The IMC port device is a pseudo device which supports process to process communication across the LAN. It allows programs running on the RC9000 PU to communicate with programs running on other computer systems attached to the LAN using the general IMC (Inter Module Communication) functions. The port handler is used by FTS and can also be used by user-written programs.

The number of IMC port device handlers and a few additional parameters for the port handlers may be customized. In the common situation when the port handler is only used by FTS (no user-written programs) with default values of the parameters for the FTS processes, the port handler parameters should be left with their default values.

It is not necessary to create IMC port links in the startarea file for FTS. The FTS processes create the links they need themselves.

5.2.4 The lanstat device

The lanstat device is a pseudo-device which is used by the lanstat utility program to obtain status information about the LAN controller, LAN device processes, LAN device handlers and traffic on the LAN. See Appendix E.1 for a description of the *lanstat* command.

An example, which also sheds light on several of the points discussed above, is the following display produced by the `lanstat dev` command.

```
lanstat dev
                                                                  lan 1
device type          free      devno     name          base       incno
CSP console             1
                                 85     moeller        (0,7)          0
IMC port                8
                                 89     ftsport1  (-8388607,8388605)   0
3270 input              1
                                 84     gin1      (-8388607,8388605)   0
3270 output             1
                                 83     gout1     (-8388607,8388605)   0
lanstat                 0
                                 82     wrk000067 (-8388607,8388605)   0
CSP printer             0
                                 87     henrik         (0,7)          0
                                 86     vejhe          (0,7)          1
CSP terminal           10
                                 88     0p72187d  (-8388607,8388605)  10

 54 free LAN/IOC externals
```

It is recommended that a lanstat link be created for each LAN controller in the startarea file (cf. section 3.5). The name shall be `lanstat<i>`, where `<i>` (1..4) is the number of the LAN controller. If this is not done, `lanstat` will attempt to create the link itself, but this is sometimes inconvenient.

5.2.5 Describing printers to PRIMO and BOSS

CSP printers
The normal way of connecting a printer to an RC9000-10 system is as a CSP printer. For each such printer a permanent link should be created by means of `makelink` (see above).

In most systems, printers are controlled by PRIMO or BOSS. In order to describe a printer to PRIMO or BOSS it is necessary to give a little more information than just the name of the printer process. This information is therefore stored together with the name of the process in a file descriptor, and the printer is then referred to by the name of the file descriptor.

The file descriptor for a CSP printer is created as follows:

```
<fname> = set <lpmode> <pname> d.0 0 0 0 0
```

<fname> is the file descriptor name which is used to refer to the printer

<lpmode> is one of the following

```
2048.14 slow, no newline conversion (mode 0)
2050.14 fast, no newline conversion (mode 2)
2064.14 slow, newline conversion (mode 16)
1066.14 fast, newline conversion (mode 18)
```

`fast` means the CSP printer only sends status reports when there is an error status. `slow` means a status is sent for every data block which is transmitted across the LAN and allows tighter recovery control. `fast` is generally recommended. If the actual printer has a large buffer, tight control is illusory anyway.

If the printer is customized to perform a newline function (both CR and LF) when it receives an LF character, no newline conversion should be specified. If the printer cannot perform this conversion, select newline conversion. The device handler will then insert the extra CR characters.

<pname> is the name of the CSP printer process.

Example

```
lp7 = set 2066.14 printer7 d.0 0 0 0 0
```

When this file descriptor exists the printer may be referred to as `lp7`, e.g.:

```
filexfer pip lp7
```

The example above uses PRIMO. If BOSS is used the name of the file descriptor may be used to refer to the CSP printer as a remote printer (note that BOSS requires the name to be at most six characters long).

BOSS can also use a CSP printer as a standard printer. In this case the device number of the printer must be specified in the customization of BOSS, and the printer process must therefore always be created with the same device number (`makelink`). The printer mode for BOSS standard printers can also be customized. The effect of printer modes for CSP printers is indicated in parentheses above under <lpmode>.

5.2.6 Printing via FTS

By making use of FTS, PRIMO can also use printers which are attached to other systems on the LAN. These systems must have FTS servers interworking with local spoolers. When PRIMO submits a file for printing by a spooler on another system, PRIMO does not have control of the spool queue and printing process. The spool queue must be monitored on the systems where the spooler runs. This may for example

be a Partner or RC900 system, or it may be PRIMO on another RC9000-10.

The feature requires a file descriptor for the server plus an additional file descriptor for each printer attached to the server. The server descriptor is necessary to hold information which PRIMO needs, but which will not fit into the printer entry.

The server file descriptor is created as follows:

```
<sname> = set ip <server> d.0 <lanno> 0 0 0
```

and each printer descriptor as follows:

```
<pname> = set lp <printer> d.0 0 0 12.<cc> 0
```

`<server>` is the FTS server name.

`<lanno>` is the number of the LAN controller which provides access to the appropriate LAN, normally 1.

`<printer>` is the printer name known to FTS server.

`<sname>` and `<pname>` are file descriptor names.

`<cc>` must be a number.

`<sname>` must be `<cc>` characters long. `<pname>` must be longer, but with the first `<cc>` characters equal to `<sname>`.

Example

```
rom = set ip romeo d.0 1 0 0 0

romp7 = set lp lp5 d.0 0 0 12.3. 0
```

When these entries exist, the printer known to the server `romeo` as `lp5` may be referred to as `romp7`, e.g.:

```
filexfer pip romp7
```

6. File Transfer Service

6.1 Introduction

The File Transfer Service (FTS) for RcLAN, which is included in the basic system software, is a general facility for transferring files between any two stations attached to the LAN. FTS can transfer a file between two similar systems or even between two computers or workstations which have radically different filing systems.

A file transfer always takes place between two parties called an FTS *server* and an FTS *user*.

A general description of FTS is given in [3], *FTS User's Guide*, part of the documentation for the Monitor. The description introduces a number of concepts, including FTS server and FTS user, and explains the parameters which must be supplied to FTS to specify file transfers. To understand the remaining part of this chapter the reader must be familiar with this general description.

On RC9000-10 systems the FTS server function is provided by the ftsserver program which must run permanently in an internal process. If the RC9000-10 system is used as load server for RC9310 and/or RC9330 control units ftsserver must be running, even if FTS is otherwise not used.

The FTS utility programs *wr* and *rr* work in conjunction with a program called ftsuser, which, just like ftsserver, must run permanently in an internal process. This process will accept requests for file transfers to or from FTS servers on other computers attached to the LAN in the form of messages from other internal processes. The programs wr and rr generate such messages, but they may also be generated by other processes. This is the mechamism whereby PRIMO may serve requests to print files on printers attached to other systems.

LAN numbering

If a system is attached to multiple LANs using multiple LAN controllers, the LAN controller number to be used to reach other servers must be made known to users of FTS. This in ormation is required both when wr and rr are used and when a main catalog entry is created describing a printer to be reached by PRIMO via FTS.

Server name

Users accessing the RC9000-10 FTS server function via the LAN need to know the server name with which the ftsserver announces itself on the LAN. The server name of the RC9000-10 system is identical to the host name of the LAN controller, usually the PU name.

Receiving files to be printed

The FTS server can receive files to be printed on a printer accessible to PRIMO on the RC9000-10 system. The FTS server passes the printer name as well as the file contents to be printed to PRIMO which controls the actual printing process. Thus PRIMO must be installed and running if this feature is to be used, and the printer name to be specified in the call of *wr* is the name of the catalog entry which describes the printer to PRIMO.

For example, if a printer is described by a catalog entry named 'lp5' on the RC9000-10 system with name 'einstein', then the command

```
wr s.einstein p.lp5 letter
```

may be given on another RC9000-10 system attached to the LAN to transfer the file named `letter` to `einstein` to be printed on printer `lp5`.

6.2 FTS Management

6.2.1 Ftsserver

The ftsserver program must run permanently in an internal process, and it can run in several incarnations within its process. The number of incarnations determines the number of FTS users that can be serviced simultaneously. The recommended name for the process is ftsserver. The name of the program file may change with new releases and will be given in the package description.

If your system has multiple LAN controllers, one ftsserver process must be started for each controller. In this case, you should add the LAN number to the process name (ftsserver1, ftsserver2, etc.).

Process requirements

Function bits 4 and 5 must be set to allow the process to create and remove links to IMC port handlers. See the Package Description for other process requirements.

Disk resource requirements

When the ftsserver receives a file which does not overwrite an existing file, it must have disk resources to store the file. The ftsserver process

should therefore be started with a reasonable amount of disk resources. The system administrator must decide the amount.

Resources used for a print file are reused after PRIMO reports back that the file has been printed, but resources for user files are consumed. This is also the case when an existing file is overwritten with a longer file.

LAN controller resource requirements
Ftsserver uses an IMC port process with a link to a port handler on the LAN controller. The port will be opened with one connection end-point for each incarnation of the ftsserver. The LAN controller must be customized with a port handler for this purpose. This is done in the SSP menu **`Model 10 LAN controller parameters.`**

Program call
Assuming the program file name is `bftserver`, the program call is as follows:

`bftsserver {l.<lanno>/bufsize.<size>/ , incno.<no>/ftscat.<catname>} 1-*`

`<lanno>` the number of the LAN and LAN controller. Default is 1.

`<size>` is the size of the data buffers which are used to move file data. The default is 4096 bytes. If less than 128 is specified, the buffer size will be set to 128.

`<no>` is the number of incarnations and must be at least 2. Default is 5.

`<catname>` is the name of the file which will be used as FTS user catalog. The default name is *ftsusercat*.

Starting ftsserver
In order to create the ftsserver process and start the program you are recommended to create a file named *startserver* with the appropriate commands and then include this file in the *startarea* file, cf. section 3.4, Creation of a startarea file.

Status commands
The following commands will make the ftsserver process display status information:

`disp` gives information about transfers in progress.

`last` gives information about the last events that have occurred, from a buffer which is overwritten cyclically. Events are file transfers and errors experienced by the ftsserver process. One line is displayed for each file transfer with the following information: transfer start time, R/W for read or write, file

name, file base, modification time for a file which is read, and transfer end time.

`stop` stops the process after current transfers.

Error messages
Error messages are not written on the system console, but only in the buffer, from which they can be retrieved be means of the `last` command. All messages, including logged file transfers are time stamped. The following error situations may be reported:

`sense error`
The LAN controller does not respond. This message is written periodically if the LAN controller is not operational.

`link rejected <rno>`
An attempt by the ftsserver to create a link to a port handler failed. The reason indicated by <rno> may be:
1 No unused external process
2 No free port handler

`reducing incs <rno>`
The number of incarnations is reduced because of lack of resources. The kind of resources is indicated by <rno> and is most likely to be:
1 Number of end-points per port handler

`link created <n> incs`
A link to a port handler has been successfully set up and initialized supporting <n> incarnations of the server.

`link removed`
The link to the port handler was abnormally removed because the LAN controller went out of operation.

`no buffer`
An incarnation of the ftsserver was stopped because the number of buffers given to the process was too small.

6.2.2 Ftsuser

Like the ftsserver, the ftsuser program must run permamently in an internal process. It can also run in several incarnations within the process. The number of incarnations determines the number of FTS users, i.e. the number of processes on the local RC9000-10 system that can be served simultaneously. The name of the process must be ftsuser. The name of the program file may change with new releases and will be given in the package description.

One ftsuser process may be used for communication on multiple LANs, if required, as the link to the LAN controller is established for each transfer.

Process requirements
Function bits 4 and 5 must be set to allow the process to create and remove links to IMC port handlers. See the Package Description for other process requirements.

LAN controller resource requirements
Each incarnation of the ftsuser process uses an IMC port process with a link to a port handler on the LAN controller. The link is created for each file transfer and removed after the transfer. The LAN controller should be customized with a port handler for each incarnation of the ftsuser process; this is done in the SSP menu `Model 10 LAN controller parameters`.

Program call
Assuming the program file name is bftsuser the program call is as follows:

```
bftsuser {bufsize.<size>/incno.<no>}1-*
```

<size> is the size of the data buffers which are used to move file data. The default is 4096 bytes. If less than 128 is specified, the buffer size will be set to 128.

<no> is the number of incarnations. Default is 2.

All information needed by the ftsuser program is obtained per file transfer from the requesting internal process.

Starting ftsuser
The ftsuser process is started in a fashion similar to the ftsserver. However, ftsuser does not require disk resources, only entries for current input and current output.

6.3 FTS User Catalog

Before starting a file transfer, the FTS server will look up the user in the FTS user catalog to establish the proper disk, base for file name search, and access rights. The FTS user catalog is a file containing an entry per user. A user is defined by a user name and a user number, so there may be several entries with the same user name, but different user numbers. The user number may thus be used as a password. By default the name of the FTS user catalog file is `ftsusercat`. A different name may be used if so desired.

A catalog containing a few predefined entries is included in the distribution. One of these entries with the user name `fts` which allows printing, but not reading and writing of files, is used by the FTS server, when the FTS `user` parameter is missing in a transfer request from an FTS user. This means that files can be spooled for printing by users who are not registered on the RC9000-10 system.

The FTS user catalog is maintained by means of the utility program `ftscatup`; the program call and an explanation of the commands is included in Appendix E.1 along with other utility programs.

6.4 FTS Utility Programs

FTS includes three utility programs:

`wr` which transfers files from the user to the server

`rr` which tranfers files from the server to the user and

`ftscatup` used to maintain the FTS user catalog.

The program calls and explanations of the available commands are found in Appendix E.2.

7. TAS Management

7.1 Installation and Start-Up

This section describes how to perform an initial installation to be used for test purposes and the like.

When the Terminal Access System has been installed according to the package description for SW8110, an initialization file and a catalog file must be generated to be used for starting up the system.

The files *stdtasinit* and *stdcattxt* are parts of the package.

stdtasinit contains standard values for the initialization parameters.

stdtasinit must be copied into the file *tasinit*, and the contents must be edited to suit the actual system. At first installation, *tasinit* can normally be used without editing.

stdcattxt contains descriptions of a number of standard user, terminal and terminal type entries; *operator* is a super user, *newterm* is a user to be used for installing new terminals, cf. 7.3.1.

stdcattxt must be copied into *cattxt*. In case there are new users, new terminals and new terminal types, they can be inserted by editing the text, cf. section 7.3 and [5].

7.1.1 Startup of TAS, newtas

With *newtas* TAS is started up with the generation of a new TAS catalog from *cattxt. newtas* has the following default contents:

```
*new tas                  ; Definition of the tas process
*base  -8388607 8388605   ; Max bases
*buf 10                   ; Number of buffers
*area 6                   ; Number of area processes
*size 35000               ; Process size
*perm disc 1200 8         ; Space for: Catalogfiles + testfiles + spoolfile
*function 0
*prog fp
*i 4.newtas               ; Read fp command in newtas
*run                      ; Start tas process
*
*new  menu                ; Definition of the menu process
*base  -8388607 8388605   ; Max bases
*buf 55                   ; Number of buffers
*area 27                  ; Number of area process
*size 64000               ; Process size
*perm disc 0 0            ; No disk resources at startup
*function 0
*prog tasterm             ; Start tasterm program in process
*run                      ; Start menu process
*unstack
; Use the tas process for creating test- and spoolfiles
base abs 8388605 8388605  ; sysbases
tascattest=set 20         ; ctname
tastermtest=set 100       ; ttname
tasspool=set 200          ; spoolname
permanent tascattest.3 tastermtest.3 tasspool.3
base std
; Start tascat program under fp in tas process
tascat init.tasinit catalog.cattxt
```

TAS is started from the main console by:

```
att s
read newtas
```

7.1.2 Start-up of TAS, starttas

In a routine start-up situation the file *starttas* is used to start up TAS, since a TAS catalog has already been created. Section 7.2.1 describes how to customize *starttas*. The following file is the default *starttas* file delivered with the system.

```
*new tas                    ; Definition of the tas process
*base  -8388607 8388605     ; Max bases
*buf 10                     ; Number of buffers
*area 5                     ; Number of area processes
*size 35000                 ; Process size
*perm disc 0 2              ; Entries for c and v
*function 0
*prog fp
*i 4.starttas               ; Read fp command in starttas
*run                        ; Start tas process
*
*new  menu                  ; Definition of the menu process
*base  -8388607 8388605     ; Max bases
*buf 55                     ; Number of buffers
*area 27                    ; Number of area process
*size 64000                 ; Process size
*perm disc 0 0              ; No disk resources at startup
*function 0
*prog tasterm               ; Start tasterm program in process
*run                        ; Start menu process
*unstack
tascat
; Start the tascat program under fp in the tas process
```

TAS can be started from the main console by the command

```
att s
read starttas
```

but the normal procedure would be to call *starttas* from the system start file, i.e. the startarea file, cf. section 3.5.

7.2 Customization of TAS

Section 7.1 described how to install TAS under s and make the system run. However, in order to optimize the efficiency of the system, you will have to customize TAS according to your specific system.

The customization of TAS consists of calculating the resource claims of the TAS processes *tas* and *menu* and the parameters of the initialization file *tasinit*.

7.2.1 The TAS processes

TAS consists of two processes, *tas* and *menu*. The program *tascat*, which among others things contains the catalog and the operator functions, must run in the process called *tas*; and the program *tasterm*, which among other things, contains the pool and MCL functions, must run in the process called *menu*.

Both programs must be started from s, and remain in the primary storage for as long as TAS is running.

Thus, TAS could be started with the following resources, and the resulting values should be written in *starttas* and *newtas*. Please see section 7.2.3 for explanation of the parameters used in the following calculations.

The tas process (the tascat program)

The *tas* process has the following claims:

```
message buffers:        5 + maxoperators + maxterminals/4
area processes:         5 + (1; if new catalog)
size (halfwords):       min. 30000 + 1500*maxoperators
                        + 8*(2*maxusers + maxterminals +
                        maxsessions)
```

disk claims (with a new catalog):

```
segments:               3 + (max. user entries/4) + (max.
                        terminal entries/14) + (max.
                        terminaltype entries/4)
entries:                2 + (3; if new catalog)
```

The menu process (the tasterm program)

The *menu* process has the following claims:

```
message buffers:        3 + 2*cpools + clinks +
                        2*maxsessions + 1.5*maxsystem +
                        maxterminals/4
area processes:         mclprograms + cpools + maxsessions
                        + 2
size:                   (halfwords): 24500 + corebufs*520 +
                        maxterminals*68 + maxtypes*58 +
                        mclprograms*24 + (maxsystem +
                        clinks)*636 + (maxsystem/2)*8 +
                        maxsessions*1102 + cpools*466 +
                        "number of segments in spoolfile"*2
```

The *menu* process has no backing storage claims.

7.2.2 Calculation of Initialization Data

The efficiency of TAS depends on the relation between the resources given to the system, and the load on the system; that is, the relation between the terminals in use and the activity on these. (Terminals that are not logged in put no load on TAS).

The load depends on terminals that are logged in to TAS' *menu* part and terminals included in pools created from applications (format8000 etc.).

The efficiency will show itself in the user waiting time and the load on the machine, especially disks.

The size of TAS will depend on the initialization parameters `maxuser`, `max terminals`, `maxsessions`, `cpools` and `clinks`. At a given size, the efficiency will depend upon the parameters `corebuf`, `spseg` and, partly, `tbufsize`.

The size of the terminal I/O buffer, `tbufsize`, should be set to the size of the terminal link buffer, in order to minimize the amount of data partition at transmission, cf. [5], ch. 8.

The more core buffers (`corebufs`) the system has at its disposal, the less transportations of data between core-buffers and spool-area on disk - and since each of these transportations, apart from putting a load on the disk, also takes time, the number of core buffers is a decisive factor in the answering times of the system.

A reasonable estimate of the number of core buffers that should be allocated in a system with medium size load, is:

```
  2 for each terminal that can log-in
+ 2 for each used MCL-program
+ about half the number of spool segments reserved for
  format8000 applications etc.
```

7.2.3. Initialization data, the tasinit file

When TAS is started, initialization data for both processes (*tas* and *menu*) are read from a text file. The name of the file can be stated as a parameter at start-up of the *tascat* program or it can keep its default name *tasinit*.

Each line in the file describes an initialization parameter in the following format:

```
<parameter name> <data> ;[<comment>]
```

The format of the line is the same as the one which applies to the field in the catalog text file (cf. section 7.3.1, Inserting, Listing and Deleting Entries), so it is possible to have empty lines, or lines only containing a comment. The lines in the text are numbered, with the first line as number 1.

The following list is a going through of the parameters, with the significance, the possible values and the default values described for each parameter.

`catdoc`
The name of the document on which the catalog files shall reside.
Values: text string, max. 11 characters.
Default value: disc

`cattest`
A test mask, governing which test data shall be printed from *tascat*. This mask should normally not be changed, cf. [5].
Values: integer, 0 - 1024
Default value: 412

`clinks`
The max. number of links that can be created by means of the `create link` message to TAS.
Values: integer, 0 - *
Default value: 0

`corebufs`
States the number of spool buffers the system has at its disposal.
Values: integer, 3 - *
Default value: 25

`cpools`
States the number of pools that can be created by means of the `create pool` message to TAS.
Values: integer, 0 - *
Default value: 0

`ctcname`
The name of the file into which testdata from *tascat* is written. This name must be present before TAS is started, and located on the bases stated by **sysbases**.
Values: text string, max. 11 characters
Default value: tascattest

`hostid`
The text that is written as headline of the sign-on message, when a user is logged in from a new terminal.
Values: text string, max. 80 characters
Default value: Velkommen til <host>

`initversion`
Date for identification of the initialization file version.
Values: integer, 0 - 999999 (yymmdd)
Default value: 0

`login`
States whether terminals can be logged in directly after start-up of the system.
Values: boolean; start = terminals can be logged from all groups, stop = terminals can only be logged in when the command 'start login' has been issued, or the command 'stop login <group>' (cf. [5])
Default value: start

logtime
The number of minutes passing after first printing of timetext and until the terminal (the user) is logged out.
Values: integer, 1 - 30
Default value: 5

maxoperators
States how many simultaneous operator log-ins there can be. Of these one will always be reserved for the main console.
Values: integer, 1 - 5
Default value: 2

maxsessions
States the number of simultaneous sessions possible in TAS.
Values: integer, 1 - *
Default value: 20

maxsystem
States the number of terminals that can use the system menu simultaneously.
Values: integer, 1 - *
Default value: 5

maxterminals
States the number of terminals that can be logged in.
Values: integer, 1 - *
Default value: 10

maxtypes
States the number of different terminal types that can be used simultaneously under TAS.
Values: integer, 1 - *
Default value: 2

maxusers
States the number of users that can be logged in simultaneously.
Values: integer, 1 - *
Default value: 10

mclbases
The upper and lower limits for the files that are part of the MCL-program database. The base should be laid out in such a way that it can only be used by TAS.
Values: integer pair
Default value: 8388605 8388605

mclprograms
States the number of different MCL-programs that can be used simultaneously under TAS.
Values: integer, 1 - *
Default value: 5

reserve
States whether the terminals should be logged in to TAS automatically when connection is established with the host.
Values: boolean; on = the sign-on screen is displayed at establishment of the link to a terminal, off = log-in to TAS is first made after an attention to the menu process. This last value should only be used during initial testing, since parts of the TAS

access control can be by-passed in this state.
Default value: on

signon A line in the sign-on text that is printed after the date etc. Multiple sign-on lines can be stated (max. 200 characters all in all).
Values: text string, max. 80 characters.
No default value

spoolname The name of the file used for spool area. The file must be on sysbases, and be present at start-up of TAS. The number of segments in the file must be used as the maximum number of segments in the spool area - see spseg below.
Values: text string, max. 11 characters.
Default value: tasspool

spseg Used for calculating the number of spool segments that are reserved in the spool area by creation of a link to a pool. By creation of a link by means of the MCL ATTENTION sentence (single tty link) 1+spseg spool segments are reserved, otherwise 2*spseg per link.
Values: integer, 1 - 2047
Default value: 3

stoptext The text that is printed on all terminals when TAS is stopped. The text is printed every minute until the system stops.
Values: text string, max. 80 characters.
Default value: Afmeld!

sysbases The upper and lower limits that are used by TAS for catalog files, spool area and testoutput files. For this interval the same comment applies as for mclbases.
Values: integer pair
Default value: 8388605 8388605

tbufsize The size of the I/O buffer used for each terminal.
Values: integer, 70 - 500
Default value: 170

termblock Defines the number of times illegal login from a terminal can be tried before the terminal is locked. If given value = 0, no lock is made.
Values: integer, 0 - 4095
Default value: 0

termcat The name of the file belonging to the catalog and holding the terminal entries.
Values: text string, max. 11 characters.
Default value: tastermcat

termtest Test mask for managing of the test generation in 'tasterm'. Should normally not be altered.
Values: integer, 0 - 4095

Default value: 1365

`timecheck` States if the timecheck facility should be activated at system start-up.
Values: boolean; on = active, off = passive.
Default value: on

`timeout` The amount of input timeout periods that must expire between the last activity in the actual session and the moment that the session is removed.
Values: integer, 1 - *
Default value: 30

`timetext` The text that is printed on a terminal before it is logged out by the automatic time control (timecheck). The text is printed every minute until the terminal is logged out.
Values: text string, max. 80 characters.
Default value: Afmeld!

`trap` States whether trace of the procedure stack etc. should be printed on the main console if runtime errors occur (traps) in 'tascat'.
Values: boolean; -1 = no printing, 0 = printing of procedure stack at trap.
Default value: -1

`ttname` The name of the file where 'tasterm' prints test. The file must be present at start-up of TAS, and located on sysbases.
Values: text string, max. 11 characters.
Default value: tastermtest

`typecat` The name of the file belonging to the catalog and holding the terminal type entries.
Values: text string, max. 11 characters.
Default value: tastypecat

`userblock` Defines the number of times log-in with false password can be tried before the user is locked. If value = 0, no lock will happen.
Values: integer, 0 - 4095
Default value: 0

`usercat` The name of the file belonging to the catalog and holding the user entries.
Values: text string, max. 11 characters.
Default value: tasusercat

7.3 Inserting Terminals and Users

The following description presupposes that a TAS catalog has already been generated, as described in 7.1.1.

The catalog is divided into three types of entries: users, terminals and terminaltypes. This section deals only with terminals and users.

7.3.1 Inserting Terminals

The easiest way to install new terminals under TAS is by using the `newterm` user id:

When you log in from a terminal unknown to the system, the following text will appear on the terminal:

```
terminal ikke i katalog
bruger id :
```

If you state `newterm` as user id, a menu called `'Installering af ny terminal'` will be displayed. If you fill in this menu the terminal will automatically be inserted in the catalog.

7.3.2 Making users under TAS

Inserting, listing and deleting entries

Inserting Entries

Each entry consists of a number of fields, containing a specific kind of information. Each field has a unique name to identify it. Each entry has a key field used for identifying it, and this key field must be unique (it is e.g. the key fields that are searched when a user tries to log in, stating her user-id.)

The description of an entry in the catalog is in text format, and when you wish to insert a new user, you must create a text file in the same format.

Each field consists of the name of the field, followed by eventual data for the field. Each field can consist of one and only one line.

The user entries consist of the following fields:

`user`	The key field, stating the user-id.
`password`	Password for the user, in text form (max. 48 characters). If empty, no password is given the user in question.

`monday`
`tuesday`
`wednesday`
`thursday`
`friday`
`saturday`
`sunday`
	The hours in which the user is allowed to log-in to TAS. The interval must be stated as two integers between 0 and 24. A different interval can be stated for each day in the week. If the field is not stated, log-in is denied on the day in question.

`block`	Number that states how many times log-in is attempted without the right password. The value is

counted by TAS. The field must be set back to 0 to remove an eventual denial of access for the user in question.

`sessions`	Number that states how many different sessions the user can start up from one or more terminals.
`privilege`	Numbers stating which privileges the user has (cf. ref. 5).
`mclname`	The name of the MCL-program that is started when a new session is created. The program must be present in compiled form in TAS' MCL-program database.
`base`	Base interval (2 numbers) stating on which bases an eventual user MCL-program shall be searched for. The bases are used if a MCL-program is not present in TAS' MCL-program database.
`groups`	A list of the terminal groups the user is allowed to choose terminals among.
`mcltext`	The text that is set in the MCL variable T when a new session is started. The text can be at most 80 characters long.
`freetext`	Text of up to 30 characters, used e.g. for describing a user (full name etc.)

Example

This is an example of a user entry:

```
user        kkk
password    pip
monday      0 24
tuesday     8 17
thursday    8 17
friday      8 17
sessions    2
mclname     stdmcl
base        100 199
groups      2 3 4 5
mcltext     56730 Test kkk 01
freetext    Example of user entry
```

As you can see this entry has the following points:

Monday the user has access all day, while he has no access at all wednesday, saturday and sunday.

He can at maximum start 2 sessions simultaneously.

The MCL program stdmcl is stated as start menu.

He has no privileges (since this field is completely omitted).

The utility program *settascat* is used for inserting the entry into the catalog. The syntax of call is:

```
settascat [<cattxt file>]
```

where `<cattxt file>` contains the description of the entry as explained above. If no file is named, the program will read from current input (e.g. the keyboard).

The program *settascat* is used for both inserting new entries and updating existing entries. If an already existing entry is stated, only the fields given in in-put (normally `<cattxt file>`) are changed. If an unknown entry is stated, it will be created as a new entry in the catalog (provided that there is resources for it, stated when the catalog was created in the first place).

Listing Entries

Catalog entries can be listed by means of the program *listtascat*. An example:

```
listtascat user.kkk
```

will list the contents of the entry given above as example.

Deleting Entries

The program *deltascat* is used for deleting entries, as in this example:

```
deltascat user.kkk
```

which will delete the entry in the above example again.

8. SOS and PRIMO Management

8.1 Installing SOS and PRIMO under s

SOS and PRIMO are installed under s by means of the files *startsos* and *startprimo*. These files are to be called from the *startarea* file during start-up, cf. section 3.5.

Note that the installation of PRIMO requires the presence of the ALGOL compiler *(SW9805, Compiler Collection)*.

8.2 Customization of SOS

In the SOS package delivered there is an auxiliary file called *soshelp*, which contains the file *sostrim*.

sostrim contains default values for the parameters to consider when customizing SOS. The customization is done by editing *sostrim*, e.g.:

```
xtrim = edit sostrim
EDIT BEGIN
1./optionid:=/, r/0/880901/,
1./minsize:=/, r/12800/20000/,
1./timeslice:=/, r/3/5/,
1./testsegments:=/, r/42/168/,
f
EDIT END
```

Here, `optionid` is changed from 0 to 880901, `minsize` from 12800 to 20000, `timeslice` from 3 to 5 and `testsegments` from 42 to 168.

The rest of the parameter values in *sostrim* will remain the default values.

sostrim further contains commands for generating the trimmed program version together with some utility and test programs.

Parameters for customizing SOS

The *sostrim* file contains the following parameters:

optionid The date of the option version as shown at startup of SOS. Should be changed every time SOS is trimmed.
Values: integer, 0 - 8388607 (yymmdd)
Default value: 0

minusers The minimum number of jobs that may be enrolled to SOS simultaneously.
Values: integer, 1 -
Default value: 1

comdusers Defines the minimum number of terminals that will be able to communicate with SOS without having created any job.
Values: integer, 1-
Default value: 2

minbufs Defines the minimal set of message buffers in the pool of resources that may be allocated to jobs.
Values: integer, 1-
Default value: 4

minareas Defines the minimal set of area processes in the pool of resources that may be allocated to jobs.
Values: integer, 1-
Default value: 7

minsize The minimal free size in primary store for user process.
The value is rounded up to the closest multiplum of 8K halfwords.
Default value: 12K

bufl Defines the size of the I/O buffer used for communicating data between terminals and jobs.
Values: must be even
Default value: 104

timeslice The time maximum allocated to jobs in primary store in seconds, before being swopped out.
Values: integer, 1-
Default value: 3

cpulimit Defines the maximum of time slices used in the CPU before a job is removed from the system.
Values: integer, 0-
Default values: 25

classloss
classgain
priogain These three parameters concern the scheduling strategy of SOS.
Values: integer, 1 - 2048
Default values: 1

`testsegments`	Defines the size of the testoutput area. Values: integer, 0- Default value: 42 (if =0, no testoutput is generated)
`oprkey`	Defines a text to be used as operator password. This text must be non-empty. Default value: opr
`swopdoc`	Defines the document on which the swop area shall be placed. If empty, the swop area is placed on the system disc. Values: text string, max. 11 characters. Default value: " " ; empty
`testdoc`	Defines the document on which to place the testoutput area. If empty, the test output area is placed on the system disc. Values: text string, max. 11 characters. Default value: " " ; empty

SOS calculates the available job resources on basis of its start-up resources. If this number is smaller than the minimum defined in *sostrim*, the system will stop after the initialization with an error message.

8.3 Making Catalogs and Users under SOS

Any job executed under SOS must be described in the user catalog. The user catalog contains information about resource demands, scope (file access), passwords, startup commands and, in case of a multiterminal job, descriptions of terminal users who are allowed to log into this job.

The user catalog consists of a set of job (process) descriptions.

The user catalog is created and updated by the program *upsoscat*.

upsoscat may list the actual contents of a user catalog in such a way that the listing may be used as input for generating a new catalog. As users may change their passwords, it is not convenient to generate a changed catalog from an edited version of the original catalog text. Instead a new catalog may be generated without destroying actual passwords, by using an edited version of an actual catalog listing.

8.3.1 The Contents of the User Catalog

The user catalog must contain the following information per process:

`process name`	max. 8 characters
`buffers / area`	the demand of the process on buffers and areas
`bases`	the standard, user and max (project) base of the project

`password`	max. 11 characters
`minsize`	the minimum size acceptable for the process
`maxsize`	the maximum size used for the job (even though SOS may have room for more)
`FP-commands`	max. 59 characters are executed when the job is created. Can be used to startup an application.
`bs-claims`	device-name (max. 11 characters) plus entries and segments for key0, key1, key2 and key3 (describing the permanence of files created) Max. 12 units
`terminals`	external identification (max. 11 characters) local identification (max. 3 characters) password (max. 11 characters) input buffering (max. number of input buffers spooled by TAS) timercount (max. number of timeout periods expired before TAS returns an answer)

The process name must identify the job unambiguously.

An arbitrary number of terminals can be registered under a process. The external identification and the local identification must be unambiguous for terminals under the same process.

Parts of the descriptions may be omitted. The *upsoscat* program will then generate default values. The default values that are non-empty are these:

Default values

```
buffers         4
areas           7
maxsize         8388608
bs-claims       disc key0 6 0 key1 0 0 key2 0 0
                key3 0 0
terminal        input
buffering       1
terminal        timer
count           40
```

At catalog generation it is checked that process names, buffers and areas are given values different from 0.

8.3.2 Call of 'upsoscat'

The program *upsoscat* is called like this:

```
[<newcat>=] upsoscat [<input>] ,
[oldcat(<cat>/no)] [list.(<outfile>/no)]
```

Explanation of parameters

<newcat>	the name of the user catalog
<input>	the name of the file containing the input. If this is omitted, input is taken from the lines following the program call.
<cat>	the name of the user catalog to be updated
<outfile>	the name of the file in which the contents of the catalog is to be printed

If the parameter `oldcat` is omitted or `oldcat.no` is stated, a new user catalog is created in <newcat>; otherwise <newcat> will contain an updated version of <cat>.

If `list` is omitted or `list.no` is stated, the new contents of the catalog are not printed; otherwise the contents will be printed in the <outfile> in a manner making it possible to use it as input for *upsoscat*.

8.3.3 Creation of the user catalog

For each process to be created, the data described in section 8.2.1 must be stated.

The syntax for input is this:

```
[maxprocess  <number>]

process <proc-name>
[buf <buf>]

[area <area>]

stdbase  <number> <number>
userbase <number> <number>
maxbase  <number> <number>

[password " [<key>]  " ]

[minsize <number>]

[maxsize <number>]

[fp "  <text>   " ]

                                    0-12
(key0     <entr>     <segm>)
(key1     <entr>     <segm>)
(key2     <entr>     <segm>)
(key3     <entr>     <segm>)
```

```
{term <name> "[<local id>] " "[<key>] " ,
                                    0-*
[<bufs> <timeouts>] ] }

[ end ]
```

Explanation of parameters:

<proc-name>	max. 8 characters
<name>	max. 11 characters
<buf>, <area>, <entr> and <segm>	non-negative integers
<text>	max. 59 characters, all characters except " allowed
<local id>	max. 3 characters
<key>	max. 11 characters

Note that the `term` option must not be followed by any other option within a process description -i.e. when used, the terminal should be the last part of the process description.

On the basis of <number> after `maxprocess` it is calculated how many processes <max> there must be room for in the user catalog. <max> is the smallest number which is a multiple of 50, and which is bigger than or equals <number>.

If `maxprocess` is omitted, <max> is set to 50.

There can be described <max> processes in the user catalog.

8.3.4 Explanation of the User Catalog

The contents of the user catalog has the following significance:

<proc-name> and <password>: The terminal user must state the name of the process and the possibly defined password to be able to create the process.

If multiple terminals should be able to login to an already created process, the <term>-option must be non empty. The terminal name (<name>), and the local identification as well as a possible password (<key>) must be given at login.

Under the stating of bs-claims, the keys, key0...key3, states the possible degrees of permanency of files created by the user. key0 allows only temporary files, while key3 allows for files on project and user scope.

The <end> parameter need not be stated if input is defined in the call of *upsoscat*, i.e. if input is a file. If input is given directly from a keyboard, it will be necessary to end with <end>.

8.4 Changing the User Catalog

There are three ways of changing the user catalog:

correction, i.e. making changes in an already existing entry;
creation, i.e. inserting processses in an already existing user catalog and deletion, i.e. removing entries from an existing catalog.

To perform one of these changes, you must call *upsoscat*, only giving one of the following parameters instead of `process`:

`cprocess` for correction,
`iprocess` for inserting and
`dprocess` for deleting.

The syntax for the input is roughly the same as for creating a new catalog.

When correcting, you will only need to state the information that you want to change, by giving the new value. The existing, non-mentioned, values will persist.

When you insert a process you will of course have to state all wished-for parameters, exactly as when creating a new catalog.

When deleting, you need only state the name of the process to be deleted.

Terminals must always be changed last, exactly as when creating a catalog. A terminal can be created with `term`, while it can be deleted with `dterm`, followed by the terminal name and the local identification.

8.5 Customization Of PRIMO

In order to utilize the resources in an adequate way it PRIMO should be adapted to fit the special needs of your installation. You should therefore consider the trim parameters in the subsequent section.

PRIMO is customized by means of the file *primotrim*, which contains a set of standard variables and commands for generating the trimmed version of the program.

PRIMO trimming parameters

options
At start-up a message showing the version and the data of PRIMO will be displayed together with this constant. At each trimming this constant should be changed to show the date of the trimming (e.g. 880901). The standard value is 0, indicating that standard trimming is used.

prcount
The number of printer coroutines in use at the same time. PRIMO uses one printer coroutine to serve each printer known to PRIMO. A printer is known from the device is signed up by the operator, or a transport is

defined to the printer (whatever happens first), and until the last transport concerning the printer is executed, and the printer is signed off by the operator. Restrictions: Min. 0.

prbufsize
The size in halfwords of the buffers used for printer output. Restrictions: Min. 104, max. 512, even.

prltpage
The value defines whether printer lists as a default should be headed and terminated by a page identifying the transport.

prltpage = 0	No leading and trailing page are printed.
prltpage <> 0	Leading and trailing page are printed on all printer lists.

prlinepage
The number of lines per printer page. The value is used in some operator commands concerning printer pages.

If `N NL` characters are found between two `FF` characters, PRIMO defines the number of pages in this part of the file to be

```
N // prlineprpage + 1.
```

Restrictions: Min. 1.

fprcount
Defines the number of IBM 3270 printer coroutines. Comsumption of IBM 3270 printer coroutines: see the description of IBM 3270 printer coroutines. Restrictions: Min. 0, max. 49.

fprbufsize
The size in halfwords of the buffer used for IBM 3270 printer output. Restrictions: Min. 104, max. 512, even. (should at least accomodate a full print line inclusive header and trailer information (+10 halfwords)).

ftscount
The number of FTS printer coroutines. One coroutine may serve one printer at the time.

oprcount
The number of operator coroutines. Defines the maximal number of operators having a conversion with PRIMO at the same time.

A conversation is started when an operator after having pressed the attention button addresses PRIMO, and is terminated, when PRIMO has written the reply on the operator command. Restrictions: Min. 1.

trsegm
The size in backing storage segments of the transport description area. The area holds descriptions of all transports known by PRIMO. A transport is known from the transport is defined, and until a certain time defined in the trimming (see trsaveminut) has passed after completion of the transport. It is possible for an application program to

release the transport description earlier, if an explicit release operation is sent to PRIMO. Restrictions: Min. 0.

testsegments
The size in backing storage segments of the testoutput area. If this number is zero, no testoutput is generated. Performance is higher if testoutput is suspended, but the possibilities for recovering system errors will be minimal. Restrictions: Min. 0.

waitops
The number of pending wait and get state of transport operations. Restrictions: Min. 0.

oprdetails

oprdetails bit 23
Defines whether details about the termination state of the transports should be displayed to the operator.

oprdetails bit 23 = 0:	no details are displayed.
oprdetails bit 23 = 1:	details are displayed.

oprdetails bit 22
Defines whether operator requests should be routed to the main operator when no operator has signed up.

oprdetails bit 22 = 0:	do not route to main operator
oprdetails bit 23 = 1:	route to main operator

Note that bit 23 is the least significant.

taccept
Defines whether transports to non-existing device hosts should be accepted.

taccept = 0: do not accept
taccept < > 0: accept

9. Integrated Communication Control Units: RC9310 and RC9330

Although any product that conforms to the RcLAN protocols may be attached to the Local Area Network and serve as communication control unit, the two units called RC9310 and RC9330 are so closely integrated into the RC9000 system that the general handling of these units, including their customization, will be described here. In this chapter the term *control unit* therefore refers to the RC9310 and RC9330 communication control units.

Since these units do not have any private storage devices or load media, the executable files and the customization files for the units are distributed on RC9000 distribution media and are accessible via an RC9000 PU designated as the *load server.*

The control units have a PROM-based initial program which allows them to behave as FTS users on the LAN and use the FTS server function on the load server to extract the files they need. Note that this implies that the FTS server must be running on the load server before any control units can be loaded.

Most RC9000 systems are delivered with at least one RC9310 control unit. This chapter therefore provides complete system administrator information about this unit and its basic software, SW9310I. Similar information for optional SW packages and for RC9330 and its packages is distributed in the form of appendices to be used in conjunction with the present guide.

9.1 Files For Control Units

9.1.1 FTS User Naming

The key to the establishment of contact between control unit and load server and to identification of the appropriate files on the load server is the FTS user name used by the control unit.

This name is obtained by concatenating a two-character prefix, *cu*, for RC9310, or *cp*, for RC9330, and a hexadecimal number which identifies the unit, the *unit number.* RC9310 has a six digit unit number, while RC9330 uses only four digits. As an example, the RC9310 unit with the

unit number 000ABC will have the FTS user name *cu000ABC*, and the RC9330 unit with the unit number 00AB will have the FTS user name *cp00AB*. Note that the names must appear in exactly this form: the prefix in lower case, and hexadecimal digits in upper case.

Every control unit has its *unit number* stored in a PROM and written on a label on the front of the unit, so that it can be read by the system administrator.

When a control unit boots after a power reset it will scan the LAN for an FTS server which will accept its FTS user name. An appropriate FTS user must therefore be established on the PU selected as load server. Note that the control unit cannot and need not know the identity of the load server.

9.1.2 Directory Organization

When RC9000-10 is used as load server, a control unit is established as FTS user by inserting an entry into the FTS user catalog. The entry must contain the FTS user name associated with the unit as described above, a base which allows the control unit to access the appropriate files, and it must have both read and write access rights.

We recommend that files installed from SW packages for control units be stored with name base `-9319..-9310` for RC9310, and with name base `-9339..-9330` for RC9330. If these bases are impractical on a given system, some other intervals which are not used for other purposes should be used. In this way *"directories"* are established whereby these files are easily separated from other files on a system.

Furthermore, we recommend that for each individual control unit a point base be assigned as a subdirectory within the directory for the relevant kind of control unit and specified for the unit's entry in the FTS user catalog. Thus, the first RC9310 control unit should be given base `-9310..-9310`, the second `-9311..-9311`, and so forth (if you have more than ten units, the directory must be widened.)

The files which are installed from SW packages are shared by all control units of the relevant kind and will be visible from all their subdirectories if the recommendations above are followed. The subdirectories are needed in order to separate customization files created for individual units by the system administrator, as well as the log files which are written by the control units. The log files are the reason why write permission is needed in the FTS user catalog entries.

9.1.3 Program Files And Customization Files

Generally the files contained in the SW packages are software modules to be loaded down to the control units and prototype files for customization files.

Operational parameters for the control units such as communication protocol, use of modem signals, address mappings, etc., are specified in customization files. Customization files are text files which are normally created by copying and editing the prototype files. In most cases the

prototype files contain so much information in the form of comments about the meaning of the parameters that customization files can be edited without the use of a hardcopy guide. In many cases parameters have default values which are assumed if a different value is not specified in a customization file. The prototype files generally show how the default values of the various parameters are specified.

When a new release of a previously installed software package is installed, the old program and prototype files will be overwritten, but customization files, set up in subdirectories, will not.

9.2 Installing Or Replacing A Control Unit

When a control unit has been physically installed, the corresponding FTS user must be established on the load server. On an RC9000-10 system this is done by means of the *ftscatup* utility program. As an example, if an RC9310 unit with unit number 000ABC is being installed as the first RC9310 unit, the following command can be used:

```
ftscatup i.cu000ABC.0.-9310.-9310.disc.w.r
```

Note that lower and upper base are assigned identically in accordance with the recommendation given in section 9.1. Note also the FTS user number is set to 0 (unused), and that files will be written on the disk named `disc`.

The example assumes that files from SW packages for RC9310 units will be installed with base `-9319..-9310` (or some other base surrounding the point base), and that customization files which are to be used only for this unit will be given the point base (`-9310..-9310`).

Installation of an RC9330 unit takes place in the same way. Only the FTS user name and the base are different, as explained in section 9.1

If a failure occurs in a control unit and it is removed and replaced by an identical unit, care must be taken to make sure that the replacement unit will access the correct files. The key here is the FTS user name, which depends on the unit number as described in section 9.1. The FTS user name which will be used by the replacement unit must be determined and an entry inserted into the FTS user catalog, identical to the entry for the removed unit except for the FTS user name. As an example, assume the unit from the example above were to be replaced by a unit with number 000ABD. The appropriate entry would be inserted as follows:

```
ftscatup i.cu000ABD.0.-9310.-9310.disc.w.r
```

If the replacement is permanent, the catalog entry for the removed unit should also be deleted. Continuing with the example, this would be done by the command:

```
ftcatup d.cu000ABC.0
```

Note that unless a station name has been defined for the control unit (cf. subsection 9.3.5) the device names will change when the control unit is replaced.

9.3 RC9310, General Description

The RC9310 unit supports the following interfaces:

* 1 Centronics parallel printer interface,
* 8 V.24 asynchronous ports, for locally attached terminals, serial printers or host lines,
* 1 synchronous V.24/X.21 interface for host connection,
* 1 asynchronous V.24 interface for a diagnostic console, and
* a LAN interface, for connecting the unit to the LAN.

The basic software package for RC9310 (SW9310I) supports the use of the parallel printer port and the asynchronous V.24 ports.

9.3.1 Channel Numbering

The printer port and the 8 asynchronous channels are identified in customization files by channel numbers.

When your system is installed, cables for terminals are already connected to the control units, and you will be informed about the channel number associatyed with each cable.

The channel number is important when you customize the unit. Therefore, it would be wise to mark each cable with the associated channel number.

Ecah RC9310 unit has 8 channels for terminals, numbered 1-4 and 9-12, and a channel for a parallel printer.

9.3.2 Terminal Functions

The primary facility offered by RC9310 is attachment of terminals and printers locally, i.e. directly to the asynchronous ports and the printer port.

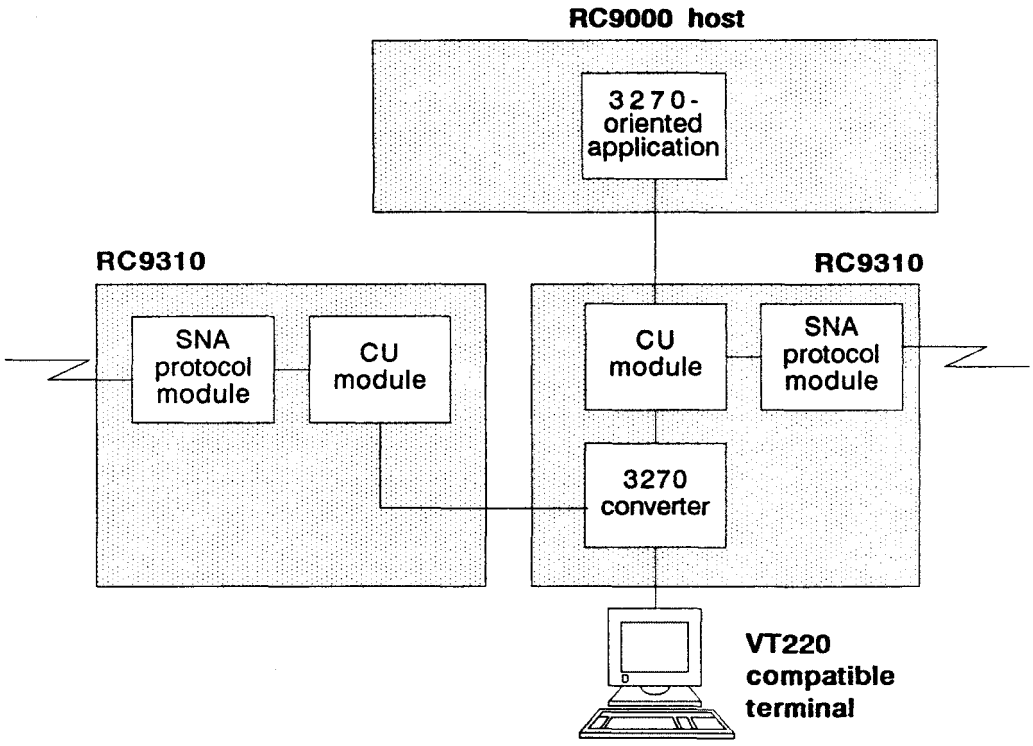
Typically the terminals and printers will be connected to the RC9000 system where the application programs run. However, it it possible to have a number of host systems attached to the LAN, and terminals are in general allowed to select freely among these.

Terminals may communicate with application programs in two different ways:

* as *character oriented* terminals, or
* as *format oriented* terminals.

Character-oriented terminals are VT100/VT200 type terminals, like the RC45, communicating according to the *CSP* protocol. Terminals working in this mode are connected directly across the LAN to the host where the application runs. The basic software package, SW9310, enables the character-oriented terminal functions, the parallel printer and the use of channels as asynchronous gateways. When the basic software package is installed, it becomes possible to install packages that implement the format oriented facilities.

Format-oriented terminals are IBM 3270 compatible terminals. The optional software package SW9311 contains a conversion facility which lets VT200-type terminals appear as 3270 terminals towards applications that use a format oriented interface. Terminals which in this way are made to appear as 3270 devices are not connected directly to a host, but indirectly via a cluster control (CU) module. In addition to local hosts, CU modules may communicate with remote IBM-compatible hosts. A terminal connected to a CU module may choose among both local and remote hosts. Normally the CU module for a terminal attached to an RC9310 unit will run on the same unit, but it is also possible to connect a terminal device to a CU module on another unit, in order to reach remote hosts connected to that unit. The possibilities are illustrated by the figure below.



What happens when a terminal attached to an RC9310 control unit is activated?

There are three possibilities, depending on how the customization files have been set up:

* The terminal may be connected to a preselected host automatically.

* A customized menu may be displayed from where the user may choose a host or even a specific application by entering a single character. Some of the choices may imply that the 3270 conversion facility is invoked. This will be done automatically. See example in Appendix D.

* A default menu is shown. The menu contains the names of all CSP hosts visible on the LAN to which the RC9310 unit is attached, and includes the 3270 conversion facility as a special choice if it is available.

In order to set up a session between a terminal and a host, RC9310 needs a *session specification*. In the first two cases above the session specifications are defined by the system administrator in customization files. In the third case, it must be typed explicitly by the user.

A session specification for a *character-oriented session* has one of these forms:

```
<hostname>
<hostname>,<application-index>
```

The `<application-index>` is used when the host is RC900 or RC9000 model 30/40, cf. the relevant host system documentation.

The session specification for a *format-oriented session* has one of these forms:

```
3270term
3270term,<CU-no>
3270term,<CU-no>,<host/application-name>
3270term,,<host/application-name>
```

Here `3270term` is a keyword which tells RC9310 to invoke the 3270 converter. `<CU-no>` is the number (`1..99`) of the CU module through which the terminal shall be connected. If `<CU-no>` is omitted, the CU module on the RC9310 unit itself is chosen. `<host /application-name>` is the name whereby the application or host is known to the CU module. If this name is omitted, and there is more than one choice, the CU module will show a menu. Note: session specifications must always be enclosed in apostrophes.

9.3.3 Printer And Nologin Channels

As opposed to terminal channels which may freely be connected to all available hosts on the LAN, channels customized as printer or nologin channels are always connected to and controlled by a specific host.

An asynchronous port should be set up as a nologin line if it is used for a console terminal on which programs may need to write even if no user has activated the terminal (not logged in). It can also be used for other devices with an asynchronous V.24 interface or for a communication line to be controlled by the host. Nologin channels communicate across the LAN by the same protocol as CSP terminals and therefore have the same interface to application programs.

The printer port can only work as a printer channel. Each of the asynchronous ports may also be customized to support printers. Printers use a protocol across the LAN which allows a host programs - typically a printer spooler - to control the Centronics printer interface very directly.

When you have customized a printer or nologin channel to connect to a specific host, you must also make sure that the host will accept the connection. On RC9000-10, you do this by making sure that an external process is linked to a device handler for the channel, cf. section 5.2.1.

9.3.4 Asynchronous (CSP) Gateway

An asynchronous channel may be configured to appear as a CSP host, i.e. it will appear as a host, under a customizable name, on the LAN, and CSP terminals may connect to the channel in the same way as they would connect to any other host.

The channel may be used as an asynchronous *gateway* to another computer system, either via a direct V.24 link, or via a modem.

Each gateway channel may serve only one terminal at a time. However, if several gateways provide the same service, they may be made to appear under the same host name on the LAN, and they will thus appear as a host link with several channels.

The gateway function offers a locally generated menu (i.e. a menu generated in the RC9310 unit) that gives the user control over the communication - e.g. capability to break the connection even if the terminal is "hung" on the remote system.

The system administrator must inform the terminal users about the control seqeunce that invokes the menu (cf. subsection 9.4.7 on how the customize this sequence).

9.3.5 Station Name And LAN Device Name

When a channel is used for a CSP device, whether terminal, nologin line or printer, that device will be identified in communication with hosts on the LAN by a *device name*. The device name is obtained by concatenating the station name of the RC9310 unit and the channel number.

For example, a terminal attached to channel 4 on the unit with the name *cu-02-* will have device name *cu-02-04*.

The station name may be specified by means of the NODENAME parameter in customization file `syscst` (cf. section 9.4). By default, i.e. if you do not specify it, the station name will be equal to the FTS user name of the control unit (cf. subsection 9.1.1). This has the undesirable consequence that the name will change if the unit is replaced. causing device names to change also. We therefore recommend that you specifiy a name for each RC9310 unit and suggest the names `cu-01-`, `cu-02-`, etc. This will give device names of the form shown in the example above.

9.3.6 Log Files

It is possible to attach a diagnostic console to the RC9310 unit. During load and start information will be written on the console. The same information will be written in a log file on the load server, in the subdirectory belonging to the unit, i.e. with the name base determined by the unit's entry in the FTS user catalog.

When setting up parameter files it may be useful to scan this file to see if any errors are found in the files. Do not be disturbed that most of the information appears unreadable. Do not try to read the file until the RC9310 unit has finished loading.

9.4 RC9310 Customization

The basic software for RC9310 is contained in the package SW9310I, which is always delivered with the physical unit. This section covers the customization pertaining to this package only.

Customization files belonging to optional software packages are not covered here, but in the guides which are distributed as part of these packages.

Customization File Syntax

A few syntactical conventions must be observed in the customization files:

The part of a line that follows a semicolon (;) is ignored, and may be used to introduce comments. If the semicolon is in the first position of the line, the whole line is ignored.

Names of parameters can appear in lower or upper case - they are not case sensitive. In the descriptions below they are kept in uppercase, in the boldface monospace font.

In the parameter descriptions the values are represented by metasymbols, e.g. `<keyword>`, `<no>` or `<string>`. `<keyword>` always represents a set of predefined value names. Like parameter names, the keywords may be typed in upper or lower case. `<no>` always represents a decimal number within a specified range. Finally `<name>`, `<string>` or `<session specification>` represents a general character string, with a specified maximum length. Such a string may be composed of control

characters given as ASCII decimals and strings of literal characters enclosed in apostrophes, separated by commas.

Default Values

Reasonable default values exist in all cases where this is possible. The default values are in effect unless a different value is specified in a customization file. If you don't create any customization files for an RC9310 unit, it will be set up as follows:

* The printer port will support a printer, which which will be connected to (owned by) the load server.

* For each V.24 port, a terminal with these properties will be expected:

  VT200 8-bit character set, linespeed 9600 bits/second, no parity, 1 stopbit/character.

  On activation of the terminal, the default menu will appear, allowing the terminal user to select among all available hosts on the LAN.

Customization Files

The following subsections describe all generally relevant parameters for the basic customization files. The meaning of each parameter, as well as the correct syntax and the legal values, are described. The prototype files also contain examples and comments on some more exotic parameters.

The following customization files pertain to the basic software package:

`configcst` — This file names the software modules that will be activated.

`syscst` — Basic parameters for LAN communication. The station name is specified in this file; otherwise the parameters should generally not be changed. Note: the form of the station name should be as in the following example:
`NODENAME=cu-01-`
See the configuration guide for RcLAN (distributed in the RC9000 support handbook) for additional information.

`itccst` — This file specifies the usage of the channels.

`menucst` — This file may (optionally) specify a *customized menu*. If the system administrator wishes to change the texts of the *default menu* (e.g. to make them appear in a language other than English) they must also be specified in this file.

`csptermcst` This file contains the texts that the CSP terminal program displays on user terminals (status messages, error messages). It should not be necessary to create the file - unless you have special requirements (e.g. messages in a language other than English).

`cspgwcst` This file contains parameters for the CSP gateway program which allows CSP terminals on the LAN to use the asynchronous ports as communication lines. The gateway will appear in the same way as a CSP host in the menus, and may be chosen in the same way.

The names of the prototype files end with `ed`, e.g. `csptermed`, meaning a file that should be edited into `csptermcst`. In this way, an installation of a new release will not overwrite customized files.

The distribution tape also contains a number of prototype files with the file name extension `dk` instead of `ed`. These files contain displayable messages in Danish. If you want messages displayed in Danish, use the `*dk` files to create the `*cst` files.

9.4.1 RC9310 Software Configuration File: `configcst`

This file determines the overall setup of the unit. It lists the software modules that should be activated when the RC9310 unit becomes operational. The basic modules, i.e. the CSP terminal and printer functions, which will almost always be needed are activated by default.

In the basic software package the following additional software modules are available:

`cspgw` The CSP gateway module. The module makes it possible for CSP terminals on the LAN to access asynchronous ports on RC9310 as communication lines.

`monitor` This module does not provide user visible services. It is a maintenance program, useful for RC Technical Staff for service purposes.

9.4.2 RC9310 Channel Usage File: `itccst`

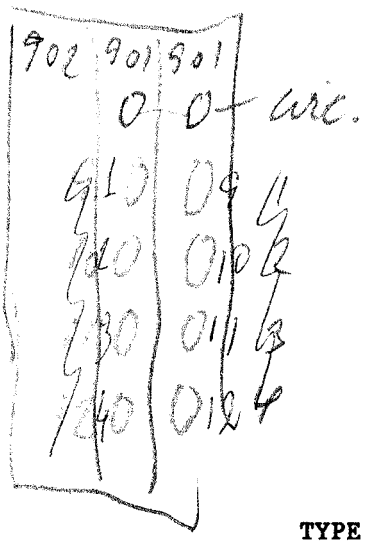
This file specifies the usage of the channels, i.e. the printer port and the 8 asynchronous ports.

The parameters for a channel begin with the keyword `CHANNEL` and a number that associates the channel with a physical port (see subsection 9.3.1 above for the relationship between channels and ports).

Then follow the parameters that specify the properties of the channel. Each parameter is identified by a keyword, followed by a setting of the parameter. The setting may be either another keyword, describing a

certain property (e.g. PARITY EVEN), or it may be a value, usually a number (e.g. LINESPEED 9600).

The following parameters are possible:



CHANNEL <no>

The start of an entry describing channel number <no>.

Legal values of <no> are:

0	The Centronics printer channel. This channel is per default for a parallel printer, and does not have to be customized further, except for the name of the controlling host which must be set with the HOSTSEL parameter.
1 - 4	Asynchronous V.24 channels
9 - 12	Asynchronous V.24 channels

Channel numbers 5 through 8 and 13 through 16 are reserved for future use.

TYPE <keyword>

The entry describes the device type. Legal <keywords> are:

TERMINAL	Terminal which will be connected to a host/application when activated by a user.
NOLOGIN	Nologin connection owned by a CSP host (the channel will automatically and exclusively be connected to the host named by HOSTSEL - see below).
HOST	The channel is made available as a CSP host for asynchronous communication. This is the CSP gateway function. Additional parameters for the channel must be specified in the cspgwcst file.
PRINTER	Serial printer connection.

HOSTSEL <name>

Identifies the host on the LAN to which the channel will automatically be connected. Normally <name> is just the name of the host. In the case of a TERMINAL channel <name> may be any session specification, cf. subsection 9.3.2.

This parameter is not used for HOST channels. For NOLOGIN and PRINTER channels the load server is selected as default. For TERMINAL channels there is no automatic connection by default.

CHARSET **<keyword>**

This parameter specifies the character set supported by the device on the channel. The following <keywords> are valid:

TTY, VT100, VT200, 7BIT, 8BIT.

7BIT/TTY and **VT100** are used for devices with a 7-bit wide character set. 7BIT and TTY are completely equivalent. The difference between 7BIT/TTY and VT100 is that if VT100 is specified, VT100 control sequences will be used for locally generated menus. If the device does not support these sequences, 7BIT or TTY must be specified.

8BIT and **VT200** are used for devices with an 8-bit wide character set. VT200 implies that the device supports VT200 control sequences, and they will appear in locally generated menus.

The appropriate specification for an RC45 terminal is VT200. However, the terminal itself should be set up as VT100. If such a terminal is connected to an RC900, the RC900 will send a sequence which sets the terminal in VT200 mode.

Note that for a HOST, PRINTER or NOLOGIN channel, the CHARSET parameter only determines the character size - 7 or 8 bits.

COMT **<keyword>**

This parameter specifies the time to wait for an XON character after XOFF has been received. <keyword> states the time in seconds. Legal values are

1S, 5S, 20S, and the special value INF (which means *infinite* wait for XON).

LINESPEED **<no>**

This parameter specifies the linespeed of the channel. <no> states the speed in bits/second. Legal values are:

300, 600, 1200, 1800, 2400, 4800, 9600 and 19200.

Note that a speed of 19200 *is not supported on* channels nos. 1 and 9.

STOPBITS **<no>**

This parameter specifies the number of stopbits used on the channel.

`PARITY` `<keyword>`

This parameter specifies the type of parity check generated and checked on a V.24 channel. Legal values of <keyword> are NONE, EVEN and ODD.

`MODEMT` `<no>`

This parameter specifies how long the Data Carrier Detected signal may be inactive before the channel is considered to be offline.

Legal values of <no> are 1 ... 60000 which indicates the time in milliseconds, and the special value INF which specifies that the signal shall be ignored altogether.

`TERMTYPE` `<string>`

This parameter specifies the terminal type. It is not used by RC8000 or RC9000-10 hosts, but UNIX hosts use the type identification to select appropriate control sequences. Consult the system administrator documentation for the relevant host(s).

If a parameter is not explicitly mentioned in the file, or no legal value is specified, a default value will be assumed and used.

The default values are:

TYPE	TERMINAL
CHARSET	VT200
COMT	INF
LINESPEED	9600
STOPBITS	1
PARITY	NONE
MODEMT	200

Example:

An `itccst` file could look like this:

```
CHANNEL 1       ; terminal with choice of hosts
      CHARSET VT100
      LINESPEED 4800
      TERMTYPE 'vt100'

CHANNEL 4       ; fixed connection to host 'othello'
      TYPE NOLOGIN
      HOSTSEL 'othello'
```

```
CHANNEL 9       ; fixed connection to load server
      TYPE PRINTER
      CHARSET 7BIT
      STOPBITS 2
      PARITY EVEN
      MODEMT INF
```

9.4.3 RC9310 Terminal Menu File: `menucst`

This file contains customization parameters for the menus the user of an RC9310 attached terminal will see. The relationship between automatic connection to a host as specified by `HOSTSEL` in the `itccst` file, the default menu and the customized menu was outlined in subsection 9.3.2 above.

There are two prototype files for `menucst`: `menued` and `menudk`. The `menued` file has displayable texts in English, the `menudk` file has these messages in Danish. If no `menucst` file is created, a default menu will be established anyway, with the same texts as specified in the `menued` file.

The default menu is designed to be self explanatory, and it should not be necessary to to change the default texts, unless you want a different language.

On the other hand, if you want to set up a customized menu, allowing users to choose a specific session by entering a single character, you will have to create a `menucst` file, starting from either `menued` or `menudk`.

The Customized Menu

A customized menu has this screen layout:

KEY <key>

This parameter defines the single key that will be used for selecting the session. <key> must be alphanumeric, that is: a/.../z, A/.../Z, 0/.../9.

TEXT <string>

This parameter specifies the text string that will be displayed on the terminal in association with the selector key defined by KEY. <string> up to and 40 characters.

SESSION <session specification>

The parameter must be a session specification, as described in subsection 9.3.2, at most 40 characters.

Example:

A menucst could look like this:

```
KEY   A
TEXT          'Reservation System'
SESSION       'othello'

KEY   B
TEXT          'Ledgers'
SESSION       'iago'
```

This file associates the letter A with the host named othello, described on the screen as Reservation System, the letter B with the host iago, described as Ledgers.

9.4.4 RC9310 CSP Gateway File: cspgwcst

This file contains parameters for the CSP gateway program. For every channel specified in the itccst file as a HOST channel at least the CHANNEL and the NAME parameters must be set. If a host channel is not mentioned in this file, it will not be activated.

The parameters fall into two groups: *general parameters*, which specify the CSP gateway menu that may be invoked by terminals using the gateway, and which is common for all channels used as CSP gateways, and *channel parameters*, which specify the characteristics of each channel.

The prototype files for cspgwcst (cspgwed, or cspgwdk for Danish language messages) contain reasonable settings of the general parameters, i.e. the menu texts and the keys used for selection.

The channel parameters are:

CHANNEL <no>

Specifies the channel. <no> must be in the range 1 ... 16.

NAME <string>

Specifies the name by which the gateway will be visible on the LAN (just as a CSP host is visible by its hostname) - max. 8 characters.

If the same name is specified for multiple channels, these channels will appear as *one* host facility with several lines. A terminal will be connected to the first available channel.

PROMPT <string>

Specifies an informative message which will be displayed on a terminal when it connects to the channel.

LOGOFF <string>

Specifies a text string to be sent to the host when a connection is being closed. It is supposed to be the proper logoff command sequence for the host.

ECHO <keyword>

Specifies whether the terminal should perform local echo or not. Legal values of <keyword> are YES and NO.

LOCAL <no>

Specifies the control character which from a terminal connected to the channel will be used to invoke the gateway function menu. <no> must be in the range 1 ... 31, where 1 means the character combination CTRL+A, 2 means the combination CTRL+B, and so forth.

BSLSH <no>

Specifies the "backslash" character, i.e. a character that will enable the character specified in LOCAL to be sent to the host *without* invoking the gateway menu. Legal values of <no> must be in the range 1 ... 255, it specifies the character in the ASCII character set by decimal value.

If you customize several channels to appear as one gateway with several lines you should set up the LOCAL and BSLSH parameters in the same way for these channels, in order not to confuse users.

The following default values are defined:

```
PROMPT          13,10,'CSP Gateway',13,10
ECHO            NO
LOCAL           4 (i.e. CTRL+D)
```

Note that there does not exist a default value for BSLSH, i.e. the backslash function will not be active unless you define a value for the character yourself.

Example:

A `cspgwcst` file could look like this:

```
CHANNEL         1
NAME            'world'
LOGOFF          'exit'
LOCAL           1
```

This file makes channel number 1 appear as a CSP host with the name `world`, defines the character string `exit` to be sent to the connected host on logoff (in this example supposedly a UNIX host), and defines the sequence CTRL+A to invoke the gateway menu.

10. The System Support Processor, SSP

The SSP is a self-contained module with private power supply, local memory and a micro processor. It is equipped with a diskette drive and a V24 port for the system console.

The SSP plays an important role during the boot procedure of the PU, where it is responsible for testing, loading and initialization the other modules in the PU. During normal operation the SSP monitors the PU and provides the system console interface.

Typical system administrative tasks to be performed via the SSP are system software installation and customization, cf. Chapter 3, *System Installation and Customization*.

This chapter is a description of the functionality of the SSP. Chapter 11 describes how to operate the SSP terminal.

10.1 The SSP States

The **PU** can operate in either

the SSP-controlled state or **the normal operating state.**

During start-up, the PU will be **SSP-controlled**, i.e. the SSP will control all the modules in the PU. The SSP can at any time reset the state of the PU if it detects an error.

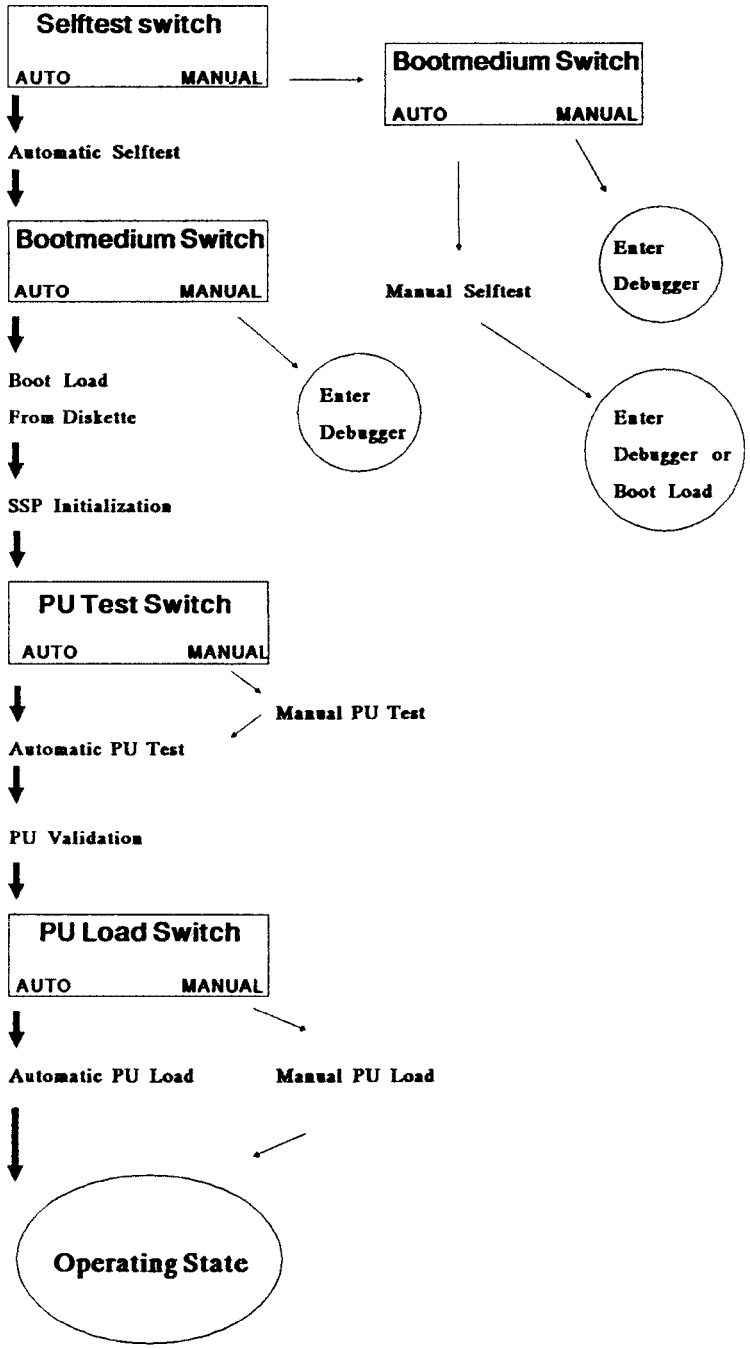
Normal operating state means that the PU will be working under the primary control of the operating system.

The **SSP** can work in:

- Self test state	automatic/manual
- PU test state	automatic/manual
- Validation state	
- PU load state	automatic/manual
- Operating state	
- Error handling state	
- PU power down state	

The following sketch is an overview of the SSP states and their interrelations.

The bold arrows indicate the normal sequence of events during a start-up.



The SSP states and their interrelations.

10.1.1 Selftest state

During the selftest the SSP verifies the function of all hardware in the SSP module, except the system bus interface.

The selftest program resides in a PROM and is normally executed after a reset.

The selftest program will activate the lamps on the SSP front panel, and this makes it possible for the operator to detect possible errors, even in case of faulty terminal communication.

During the automatic selftest program a number of fairly short tests are executed. If any defects appear, the mode will be changed to manual.

10.1.2 PU test state

In this state the SSP functions as a test master; all other PU modules are test slaves. The PU test verifies the function of the system bus, the system bus interface and the function of all other modules in the crate.

When the test is finished the system bus is reset and the SSP enters validation state. If the test program detects any errors, the system state error lamp is turned on.

10.1.3 Validation state

In this state the SSP validates whether the configuration of the PU is legal and a load possible.

A legal configuration must meet the following prerequisites:

Memory modules
At least one memory module must be present and functional. The number of memory modules is only limited by the number of slot positions; however, a maximum of 16 MHalfwords will be used.

LAN controllers
At least one LAN controller must be present and functional.

Input/Output controllers
At least one I/O channel controller must be present and functional.

CPU modules
At least one CPU985 and one FPU985 must be present and functional. The maximum number of CPU modules in a subrack is four.

If the PU subrack contains a legal configuration, the autoload switch will be read. If the switch is in the autoload position, the SSP enters the PU autoload state. If the switch is set in manual position, the SSP enters manual PU load state, and waits for operator intervention.

If the PU contains an illegal configuration, the SSP will enter the error handling state.

10.1.4 PU load state

During PU load the SSP downloads programs for all modules and starts the operating system.

10.1.5 Operating state

In the operating state the PU runs under the control of the operating system, and the sole role of the SSP is to make the following facilities available:

- Operating system console
- Real-time clock
- Healthiness monitoring
- Display of messages on the terminal

The SSP will leave the operating state in case of an error reported from the healthiness monitoring or from the operating system, in which cases the SSP will enter the error handling state.

The operating system can request a shutdown, which will cause the power to the rest of the PU to be turned off, the SSP then enters the powerdown state and waits for the power to the SSP to be turned off.

10.1.6 Error handling state

In this state action upon faults in the PU is decided. It is used as a waiting state, when the operator does not want to enter the operating state. The SSP enters the error handling state due to the following types of errors:

External hardware error can be power failure or temperature failure. These types of failure cause the power to the rest of the PU to be turned off and the SSP to enter the power-down state.

PU hardware error can be a parity error, data format error or timeout on the system bus or a watchdog timeout, or other hardware errors on modules in the PU. This type of error will cause the SSP to enter the PU test state, where an automatic test is executed to identify the failing part of the PU. After the test, a new validation is performed excluding the faulty modules.

Load error can be caused by a missing or incompatible program file. The SSP will stay in the error handling state.

Validation error. The PU cannot be loaded because the functional modules do not constitute a legal configuration. The SSP stays in the error handling state.

If the SSP enters the error handling state from the operating state upon request from the operating system, the operator must specify whether the SSP should enter the automatic/manual PU test state, stay in the error handling state or enter the power down state.

If the SSP stays in the error handling state, the operator can choose to enter the power-down state, the manual/automatic PU test state or reload.

10.1.7 Power-down state

In this state the power to the rest of the PU is turned off and the lamps for all not functioning modules are turned on. The power off can be caused by external hardware error or an operator command. In both cases the SSP enters this state through the error handling state.

The operator can instruct the SSP to leave the power down state.

10.2 The SSP Menu System

The SSP menu system is the operator's interface to the SSP programs.

There are three different types of menus:

- Selection menus, where the operator selects a function that he wants executed.
- Formfill menus, where the operator fills in the forms with adequate parameters, in order to execute the menu afterwards.
- Display menus, which list information.

The operation of the SSP terminal is dealt with in Chapter 11, *The SSP Terminal.*

11. The SSP Terminal

The SSP terminal is the operator interface to the SSP. This chapter describes the SSP message system and how to operate the SSP terminal.

11.1 The SSP Menu Screens

The SSP menus constitute a hierarchy of many menus. Before you begin any actual customization you should explore this hierarchy from the top (**SSP main menu**) to the bottom.

The following figure is an example of a menu screen when working in the SSP menu system:

```
        CUSTOMIZATION UTILITIES                                    CO3

        P - Set PU name
        M - Module customization
        A - Set module customization to actual configuration
        E - Set serial number used for ethernet address calculations

        O - Modem customization

        1 - Parameters for model 10
        3 - Parameters for model 30/40

        R - Read customization data from diskette
        D - Read another PU's customization data from diskette
        W - Write customization data to diskette

        C - Real time clock
        T - Timezone and daylight saving

        Select function ==> W
  Customization file written
-------------------------------------------------------------------
  [joanie     PUTA0003I 1141] Automatic PU test started
  [joanie      PUA0014I 1141] PU test finished
  [joanie     PULA0002I 1141] Automatic PU load started
  [joanie      PUA0015I 1142] PU load successfully finished
```

The top area is used for operator communication to and from the active console.

The last line in the operator communication area is used for 'menu messages'; in the above example "`Customization file written`".

The bottom area is used for messages from the SSP programs and the operating system. The message system is described in section 11.3.

11.2 The Consoles

Note that the character keys referred to throughout this chapter correspond to an RC45 terminal. If you need more detailed information about the character keys, please refer to Appendix B.

The SSP terminal has various consoles:

`ssp` The SSP console. This console handles the menu system, which is thoroughly described in section 11.3, and makes the SSP functions available to the operator.

`os` The operating system consol. This console gives access to the operating system that runs on the main CPU. This console is only available when the SSP is in the operating state.

The active console can be changed by typing `[Ctrl a]`. When you type this character the operator communication area will be erased and a list of the available consoles will be displayed. A new console is selected by typing the first character of its name and `[CR]`.

```
C O N S O L E   S E L E C T   M E N U

    0 - os
    S - ssp



Select function ==>
```

11.2.2 Software release version number display

When you activate `[DEL]` in the `SSP main menu`, the last line in the operator communication area will display the version number and the date of the SSP software release you are using.

11.3 The SSP Message System

11.3.1 Message display area

As mentioned in the beginning of this chapter, the bottom area of the screen is reserved for messages both from the SSP programs and the operating system.

The message display area consists of 4 lines. Input from the operator is not possible here.

One message may extend over several lines. When a message is displayed, the message display area will scroll so that only the most recent messages are displayed.

11.3.2 Extended message display area

The message area will, however, be extended to 22 lines if you return to the console select menu (by typing `[Ctrl a]`) and push the `[DEL]` key.

11.3.3 Message format

The messages have the following format:

`[puname xxxxnnnns hhmm] <log message>` or

`[puname bbb(nn)s hhmm] <log message>`

puname is the name of the PU, which can be up to 10 alphanumeric characters.

xxxxnnnns is the message identifier consisting of three subfields:

xxxx is the subsystem originating the message.

nnnn is a log message code number consisting of four decimal digits.

s indicates the severity class of the message and consists of one alphanumeric character. There are four levels of severity:

i (I) Informative message. A system status change has occurred and does not require user intervention.

w (W) Warning. A problem has been detected, but the operating system is still functional. User intervention is required as soon as it is convenient.

`s (S)` Severe error. Partial loss of the system has been detected. Operator intervention is required.

`bbb(nn)s` is the other possible message identifier, where `bbb` indicates the board type (e.g. IOC, DLC) and nn the slot number.

`hhmm` indicates the time: hour and minute, local time.

The lower case characters indicate that the message is reported from the operating system. The upper case characters indicate that the message is reported from the SSP.

`Log message` is the actual message. One message may extend over more than 1 line. When the message is displayed the message will scroll, so that the most recent message is displayed.

11.3.4 Message logging

The SSP logs all messages originating from the SSP. The messages will be logged in two files: one with the current log messages and one with the previous log messages in relation to the latest reset. The files can each hold 128 messages.

In the unlikely event of the messages exceeding this maximum, the logging will be suspended and the messages will only appear on the terminal.

The SSP displays messages both from the SSP and the operating system, but logs only messages from the SSP.

When the SSP has entered the operating state and the system is under the control of the operating system, the SSP log files will be transferred to the operating system and the logging will then take place here.

11.3.5 Log message printouts

In the `Log utilities menu` you can specify whether you want a log printout on a slave printer connected to the SSP terminal. The printer will write all messages that appear on the screen.

11.4 Operating the Terminal

This section describes how to edit the operator communication area.

The editing takes place in the input field on the screen, the size of which depends on the menu in question.

The following pages will list the characters that can be activated during the editing and the resulting effect.

Appendix B. gives a detailed description of the character codes.

11.4.1 SSP character encoding

`[BS]`
The character to the left of the cursor is erased, and the cursor is moved one position to the left.

`[ß]`
The character under the cursor is erased.

`[Ctrl x]`
All characters in the input buffer are erased. The cursor is moved to the extreme left of the input field.

`[left arrow]` The cursor is moved one position to the left.

`[right arrow]` The cursor is moved one position to the right.

`[â]`
The terminal mode is toggled.

The following characters can be used to terminate the input:

`[CR]`
`[up arrow]`
`[down arrow]`
`[home]`
`[SEND]` or `[Ctrl d]`
`[DEL]`

When typing any other character, the terminal reaction will depend upon the mode:

insert mode:	all characters to the right of the cursor are moved one position to the right. The rightmost characters are deleted from the screen and from the buffer. The typed character is placed in the cursors position and the cursor is moved one position to the right.
replace mode:	The characters under the cursor are replaced by the typed character. The cursor is moved one position to the right.

As mentioned in section 10.2, there are three different types of menus, and the operation of the menus vary accordingly.

11.4.2 Selection menus

When the menu is displayed the input field will contain an appropriate default selection, which can be changed by the operator and the menu can be executed with one of the following characters:

`[CR]` or `[SEND]`	The selected option in the input field is executed. If the input field contains an invalid selection, an error message will be displayed.

[up arrow]	The default selection is changed to the selection above. If the default was the topmost option the default will change to the option at the bottom.
[down arrow]	Similar to [up arrow]
[home]	The default selection is changed to the topmost selection.
[DEL]	The menu is aborted.

11.4.3 Formfill menus

The input fields will contain default values. When you have filled in an input field it will be checked to see if it contains a legal value. If it contains an illegal value an error message will appear. The menu can be ended with one of the following characters:

[SEND]	The requested function is executed with the given parameter.
[up arrow]	The active line is changed to the line above and the menu is reshown. If the active line was the topmost line, the active line is changed to the bottom line.
[down arrow]	Similar to [up arrow].
[CR]	Like [down arrow], except when the active line is the bottom line, in which case the function is executed with the given parameter.
[home]	The active line is changed to the topmost line and the menu is reshown.
[DEL]	The menu is aborted.

11.4.4 Display menus

A display menu can be terminated by one of the following characters:

[CR] or [SEND]	The values in the display menu are updated and the menu is reshown.
[DEL]	The menu is aborted.

12. System Administrator's Three Commandments

The prime duty of the System Administrator is to maintain the system security.

The system security will be maintained if you observe the instructions given in the three Commandments below:

1) Always structure the allocation of bases. Section 9.1.2 gives a good example of this RC9000-10 catalog structure.

2) You should design a back-up plan according to the need for system backup. The plan should be changed as the use of the system changes. (Cf. ref. [9]).

3) You should keep a diary of your system, and make sure that every small change in the system, e.g. new hardware/software, customizations of operating systems and other software, startarea files etc., is entered in your diary.

If you observe the above mentioned Commandments it will always be possible to make your system operational within reasonable time.

If you do not, you are certainly skating on thin ice.

Appendix C. Routine Maintenance

C.1 The Front Covers

The cabinet front covers may be cleaned with a firmly wrung soft cloth when required. Use plain water and, if necessary, mild soap.

C.2 The Air Filters

It is essential that the air filters placed on the inside of the front covers be cleaned regularly, e.g. once a month.

Omission may cause superheating of the system.

The cleaning is best done with a vacuum cleaner and second best by shaking off the dust out of the window.

Appendix A. References

Part numbers in references are subject to change as new editions are issued and are listed as an identification aid only. To order, use package number.

1 PN 991 11261
Channel Device Processes, Reference Manual
Part of: SW9890I-D, Monitor Manual Set

2 PN 991 11262
LAN Device Processes, Reference Manual
Part of: SW9890I-D, Monitor Manual Set

3 PN 991 11284
FTS User's Guide
Part of: SW9890I-D, Monitor Manual Set

4 PN 991 11260
Operating System s, Reference Manual (To appear)
Part of: SW9890I-D, Monitor Manual Set

5 PN 991 11268
TAS Operatør Manual
Part of: SW8110-D, TAS Manual Set

6 PN 991 11267
TAS Bruger Og MCL Manual
Part of: SW8110-D, TAS Manual Set

7 PN 991 11264
System Utility, Part 2
Part of: SW8010I-D, System Utility Manual Set

8 PN 991 11265
System Utility, Maintenance Programs
Part of: SW8010I-D, System Utility Manual Set

9 PN 991 11310
Save, incsave; Load, incload
Part of: SW8010I-D, System Utility Manual Set

10 PN 991 11266
Screen Editor, Reference Manual
Available as: SW8020-D, Screen Editor

11 PN 991 11255
RC9000-10 System Software
Part of: SW9910I-D, *System Overview* (To appear)

12 PN 991 11260
Monitor, Reference Manual (To appear)
Part of SW9890I-D, Monitor Manual Set

13 PN 991 11263
System Utility, Part 1
Part of: SW8010I-D, System Utility Manual Set

Appendix B
The SSP Terminal Character Codes

This appendix lists all characters which are assigned a special meaning on the SSP terminal.

Note that some of the characters have multiple encodings.

RC45 key	ASHII	Hex. Code	Function
[Ctrl a]	SOH	01h	Console select
[Back Space]	BS	08h	Delete previous character
[ϕ]	CSI 3 ü	9bh 33h 7eh	Delete current character
[Ctrl x]	CAN	18h	Delete buffer
[left arrow]	CSI D	9bh 44h	Move cursor left
[right arrow]	CSI C	9bh 43h	Move cursor right
[â]	CSI 2 ü	9bh 32h 7eh	Toggle terminal mode
[CR]	CR	0dh	New line
[up arrow]	CSI A	9bh 41h	Up arrow
[Ctrl z]	SUB	1ah	Up arrow
[down arrow]	CSI B	9bh 42h	Down arrow
[Crtl j]	LF	0ah	Down arrow
[Home]	CSI H	9bh 48	Home
[Send]	CSI 7 ü	9bh 37h 7eh	Execute
[Ctrl d]	EOT	04h	Execute
[Ctrl c]	ETX	03h	Abort
[DEL]	DEL	7fh	Abort
[PA1]	CSI 1 ü	9bh 31h 7eh	Abort
[ESC]	ESC	1bh	Attention

These encodings correspond to the usual VT100 settings except for [Home] and [Send] which are programmed automatically during upstart. The functions performed by these keys can, however, be simulated by other keys. The terminal may use the 7-bit version ESC Æ (1bh 5bh) in stead of the 8-bit CSI (9bh).

Appendix D.
Information For RC9000-10 Terminal Users

This chapter contains useful information for RC9000-10 terminal users. The first section is specific for users of RC9310 attached terminals, while the remaining sections are valid for all terminal users, whether the terminal is a CSP terminal or an 3270 terminal.

D.1 Terminals Attached To RC9310 Control Units

A terminal attached to an RC9310 unit must be connected to a host system across the LAN before communication between the terminal user and the host can begin. The connection is set up under control of a program running on the RC9310 unit. To attract the attention of this program it is enough to switch the terminal on and strike any key. Depending on the customization of the RC9310 unit, one of three things may then happen:

1 A **default menu** will be displayed, allowing the user to choose among hosts active on the LAN. The menu will look like this:

```
Hosts on network:

figaro              romeo
isolde              susanna
joanie              tristan
mike                zonker
paladin             3270term


Enter name:
```

The user may now select a host by typing the full name, followed by <CR>, or by typing an unambiguous abbreviation, followed by <CR>. The host named `figaro` may thus be selected by either `figaro` <CR> or by `f` <CR>.

`3270term` indicates that the 3270 conversion facility is available. The user may select a session with an application that uses a format-oriented interface by choosing the `3270term` "host".

The user may type a full 3270-session specification, or parts of one. The formats and consequences are described in subsection 9.3.2. The system administrator must inform users about available choices.

2 A **customized menu** will be displayed, allowing the user to select a session by striking a single key:

```
        RC9310 Session Establishment   Rel. 1.2

 Key                Application

 1                  Reservartions
 2                  Payroll
 3                  Inventory


          Press key to select (or /hostname)
```

The user may select e.g. the *Reservations* application by striking `1 <CR>`.

It is possible to return to the default menu by striking `/ <CR>`, or to select a host directly by typing `/ <hostname/session specification> <CR>` (provided the user knows the name of an available host or 3270 session).

The session invoked by the key selector may be a CSP or a 3270 session. As the user must be informed of the type of session in order to operate the terminal accordingly, the system administrator must inform users of the nature of the sessions.

3 Finally, the terminal may be set up to connect automatically to a host upon activation.

When the terminal has been connected to a host, or to a (3270-) session, communication with the control unit is transparent, and remains so until the connection is broken.

Normally the connection will be broken when the user log out out from an access system such as TAS, or from the operating system, or, in the case of a 3270 session, when this session ends.

When this happens, communication will again be with the control unit, and a new connection can be established.

D.2 Communicating With An RC9000-10 Host From A CSP Terminal

When a CSP terminal is connected to an RC9000-10 host, and the host has resources to create a terminal process, it will display the message `connected to <hostname>`. Otherwise a message which indicates lack of resources will be shown.

If the host system uses the Terminal Access System (TAS), the terminal will then automatically begin communication with TAS. Otherwise the user must generate an attention by typing an interrupt character (normally ESC) to begin interaction with an internal process. The interrupt character is echoed as `att`, and the operator must then type the name of the desired internal process terminated by CR. If the terminal is reserved by a process, this dialog is omitted, and an attention is just sent to the reserving process.

The CSP host function keeps track of the internal process which is currently interacting with the terminal. The relevant interactions are: output from process to terminal, input or attention from terminal to process, or reservation of the terminal. When the current process changes, i.e. before output from a new process is displayed of input to a new process is accepted, the CSP host function writes `from <process>` or `to <process>` on a separate line, so that the operator will know the source or destination of the data.

To send an attention to the current process it is unnecessary to type the process name, i.e. if the `att` prompt is answered with an empty line, an attention is sent to the procss which recently interacted with the terminal.

The interrupt character may be customized on the LAN Controller, cf. subsection 5.2.1.

The RC9000-10 CSP host function supports three commands which were also found on older RC8000 front-ends:

`CTRL+ø st`	show *hostname*
`CTRL+ø re`	remove terminal connection
`CTRL+ø di`	display available hosts

These commands are intercepted by the LAN Controller, and there is no interaction with internal processes.

When one of these special commands or the name of a process to receive an attention message is expected, the terminal is in a special mode, where normal processing of input/output to/from internal processes is suspended. In order to obey standard timeout rules towards internal processes, the CSP host function imposes a time limit of 15 seconds on the typing of the command following `CTRL+ø` or the process name.

The internal process may set the terminal mode to be *conversational* or *canonical*. In particular, TAS may be instructed to set the terminal mode.

When the mode or some other terminal attribute, such as echo, is being changed on the instruction of an internal process, input already typed will be flushed, even it has been echoed (canonical mode). The interrupt character may also be temporarily lost at such times.

In conversational mode, input in only accepted when it is specifically requested by an internal process. In canonical mode, which is provided because it will work better for terminals connected via a PAD and a packet switched network, input is always accepted and buffered by the LAN Controller. However, if the input buffers of the LAN Controller are overwhelmed, input may be discarded even when it has already been echoed. When this happens, a message will be produced by the LAN Controller to inform the user. If the user always waits for the prompt from the program (usually the bell) before typing input, the situation will not arise.

Backspace may be used to edit the current line, and the whole line may be regretted by ENQ (CTRL+E).

The RC9000-10 host breaks the connection when the last internal process which is a user of the CSP terminal process is removed or dissociates itself from the terminal. Normally this is when the user logs out of TAS.

D.3 Communicating With An RC9000-10 Host From A 3270 Terminal

The terminal must be connected via a control unit providing the 3270 CU function. The control unit maintains a menu of communication lines and/or applications in the LAN environment to which each terminal device may be connected.

An RC9000-10 application process may, via links to a 3270 device handler pair on the LAN Controller, make itself reachable from 3270 CUs attached to the LAN. The application will be shown in the menus and may be selected by terminal users. Communication will then be transparent between the application and the terminal device.

When the operator has pressed an attention key, thereby causing the 3270 CU to send an input message to the host, this message will be buffered on the LAN Controller and delivered to the application when it issues an input operation to read the message. In this situation the LAN Controller activates a timer, and if the timer expires before the message has been read, the LAN Controller will discard the input message and show the message `Host timeout, data lost` in the terminal status line. The length of the timer may be customized. Normally it should be set to that the message will only be shown when a serious error has occurred on the RC9000-10. The best value for the timer depends on application response times.

To remove the timeout message, generate a Reset attention from the keyboard. If the application is still running, resend the input message by means of the same attention key which was used in the first place. Note,

that a relatively short timeout my be used to detect that an application is running slowly.

D.4 Terminals On A CSP Gateway

When using a CSP gateway, the terminal user must be aware of some

Appendix E
Auxiliary Programs Reference

This appendix contains lists of the most frequently used auxiliary programs.

Section E.1 contains a complete survey of all available utility and maintenance programs (including FTS utility programs), divided into groups according to their functionality. Section E.2 is a selected alphabetical list of frequently used programs, including examples of use and program calls. Section E.3 lists the most commonly used mode-kinds, section E.4 the Contents Keys, E.5 the s-commands, and E.6 is a list of the Screen Editor commands.

This appendix is intended as a quick reference; for complete descriptions, please use the references given below.

E.1 Utility and Maintenance Commands, Abstracts

This section divides the System Utility and Maintenance Programs according to their functionality. You may therefore find that some of the programs are included under more than one heading.

* indicates that the program is included in the alphabetical list in E.2.

@ indicates that BOSS is required.

(u) indicates that the program is included in *System Utility Programs* (ref [7] and [13]).

(m) indicates that the program is included in *System Utility Maintenance Programs* (ref. [8]).

(sl) indicates that the program is part of *System Utility; Save, incsave; Load, incload* (ref. [9]).

(fts) indicates that the program is an FTS utility program. Ref. [3], *FTS, User's Guide*.

E.1.1 Catalog Handling Programs

E.1.1.1 Creating entries

assign (u) creates a temporary entry or changes an entry so that the tail of the two specified entries become identical.

entry (u) * creates a temporary catalog entry or changes a catalog entry according to the parameters in the call. The program is a supplement to the program *set* and is used when one wants to set some of the elements in the entry tail by copying from the tails of other entries.

set (u) * creates a new catalog entry with scope temp or changes an already existing entry (with scope temp) according to the parameters.

setmt (u) creates catalog entries of scope temp describing files on magnetic tape according to the parameters.

E.1.1.2 Changing entries

backfile (u) * subtracts one from the file number (unless it is 0) in the tails of the entries specified and signals reach of file 0.

basemove (m) * moves files from one name base to another.

changeentry (u)* changes an existing catalog entry according to the parameters in the call. The program is a supplement to the programs *set* and *entry* and is used when one wants to change some of the elements in the entry tail by copying from the tails of other catalog entries.

entry (u) * creates or changes a temporary catalog entry according to the parameters in the call. The program is a supplement to the program *set* and is used when one wants to set some of the elements in the entry tail by copying from the tails of other entries.

nextfile (u) * adds one to the file number in the tail of the catalog entries specified.

permanent (u) changes the permanent key of the specified entry to the specified integer.

rename (u) * changes the names of catalog entries as specified.

scope (u) * changes the scope of catalog entries as specified in the call of the program.

set (u) * creates a new catalog entry with scope temp or changes an already existing entry (with scope temp) according to the parameters.

E.1.1.3 Looking up entries

lookup (u) * finds and lists catalog entries with specified name.

procsurvey (u) * lists types of procedures and their parameters, as well as the procedure date.

search (u) * finds and lists all catalog entries with a given scope.

translated (u) * prints the data of translation which is found in all ALGOL/FORTRAN programs.

E.1.1.4 Changing catalog base

base (u) * changes the catalog base of the job process, thereby changing the name base, and displays the process bases of the job process.

E.1.1.5 Surveying catalogs

cat (u) * works as catsort, except that no sorting of entries prior to output takes place, i.e. the entries specified in the call are listed in the order in which they are found in the catalog.

catsort (u) * lists on current output selected parts of the main catalog (or any subcatalog) sorted according to the parameters. At last also total number of entries and segments output are listed.

search (u) * finds and lists all catalog entries with a given scope.

E.1.1.6 Removing entries

clean (m) * removes all catalog entries with catalog base wi hin the specified limits.

clear (u) * removes catalog entries with name and scope as specified.

clearmt (u) removes catalog entries according to the parameters.

delete (u) * finds and removes all catalog entries with a given scope, possibly filtered by filters given. The filters work on the entry name and on the document name as for the program *search*. Use **with care!**

E.1.2 Data Handling Programs

E.1.2.1 Creating data in a file

binin (u) * The program can input files generated by the program *binout*. The programs *binin* and *binout* are primarily used when binary files are stored on magnetic tape.

binout (u) * The program can output catalog entries and contents of files in a format (a binout file) which may be input by the program *binin* or the program *initialize catalog*.

char (u) outputs the specified character the specified number of times.

edit (u) * is a line oriented program for editing text files.

head (u) * prints a number of form feeds and a page head containing the name of the job and the date.

label (u) * outputs a BOSS label on file 0 of the specified magnetic tape.

message (u) * may be used to make messages during the job flow.

E.1.2.2 Changing data in a file

correct (u) * The program corrects specified words on the backing storage according to the parameters. The program may also be used to print specified bits as integers.

edit (u) * is a line oriented program for editing text files.

rubout (u) rubs out the contents of the specified backing storage files. If demanded the catalog entry is removed after the cleaning.

E.1.2.3 Moving data between files

binin (u) * The program can input files generated by the program *binout*. The programs *binin* and *binout* are primarily used when binary files are stored on magnetic tape.

binout * The program can output catalog entries and contents of files in a format (a binout file) which may be input by the program *binin* or the program *initialize catalog*.

compresslib (u) compresses into one single area a number of external ALGOL/FORTRAN procedures or code procedures to occupy a minimum number of segments.

copy (u) * copies one or several files into another file and calculates the number of characters copied and the sum of their ISO values. Blind characters are not copied. The program can be used instead of *edit* if only a simple copying is wanted. Furthermore the program may be used for check reading of text files.

edit (u) * is a line oriented program for editing of text files.

move (u) * performs blockwise copying of files on backing storage or magnetic tape.

E.1.2.4 Verification of data in a file

binin (u) * The program can input files generated by the program *binout*. The program *binin* and *binout* are primarily used when binary files are stored on magnetic tape.

edit (u) * is a line oriented program for editing of text files.

print (u) * prints from a backing storage area or directly from the core store with specified formats. The program is primarily intended for printing of dumped core areas.

translated (u) * prints the data of translation which is found in all ALGOL/FORTRAN programs.

E.1.2.5 Backup and restore

incload (sl) * is used to (a) relocate and reload catalog entries and backing storage areas from magnetic tape backups created by *incsave*, (b) relocate and read files from such tapes as a

simple test of the tape format, and (c) produce documentation of such backup from the save catalog resident in the file system.

incsave (sl) * transfers catalog entries and bs entries, i.e. files, to magnetic tape for backup purpose in an incremental procedure based on the concept of levels.

load (sl) * The program restores files from a backup made by *save*.

save (sl) * The program performs backup of files and file systems.

E.1.3 Job Control Programs

E.1.3.1 Job flow control

claimtest (u) checks the claims of the calling process according to the call parameters and leaves the ok bit true if the claims specified are present, false otherwise.

bossjob (u) @ sends a newjob message to BOSS demanding the specified file enrolled as job file in an off line job. In this way a job running under another operating system may create a BOSS job. The actual job continues with the next FP command.

corelock (u) @ sends a corelock message to the parent (the operating system) demanding that the job should stay in the core the specified number of seconds.

coreopen (u) @ sends a coreopen message to the parent (the operating system) signalling the end of a corelock period (cf. *corelock*). The program is only used on process control installations.

end (u) * returns current input to previous current input at the positions where it was left.

finis (u) * terminates the job.

init (u) forces a reinitialization of FP.

if (u) * makes the execution of the next FP command conditioned by the values of one (or several) mode bits. The condition may reflect the success of the latest program end (or it may correspond to the mode bits as set by a call of the program *mode*).

job (u) @ makes it possible to use tapes containing a BOSS job request directly under s.

mode (u) * changes the FP mode bits specified in the call and may thereby change the working cycle of FP.

newjob (u) sends a newjob message to the parent (operating system) demanding the specified file enrolled as job file in a new off line job i.e. in this way a new job is created. The actual job continues with the next FP command.

online (u) @ turns the job into the conversational mode where the current input to the job is typed on the terminal at run time. A conversational job is very resource demanding and the user must have a special option in the user catalog.

repeat (u) * makes it possible to repeat (a specified number of times) a series of FP commands placed in brackets.

replace (u) * sends a replace message to the parent (the operating system) defining a file as replacement for the current job file. After termination of the job BOSS will create a new job with the same name and the specified file as job file. A replace message from an on line is not accepted by BOSS.

skip (u) * bypasses parts of current input as specified in the parameter list.

timer (u) @ sends a timer message to the parent (the operating system) demanding a provoked interupt after a certain time.

E.1.3.2 Job mode control

corelock (u) @ sends a corelock message to the parent (the operating system) demanding that the job should stay in the core the specified number of seconds.

coreopen (u) @ sends a coreopen message to the parent (the operating system) signalling the end of a corelock period (cf. *corelock*). The program is only used on process control installations.

online (u) @ turns the job into the conversational mode where the current input to the job is typed on the terminal at run time. A conversational job is very resource demanding and the user must have a special option in the user catalog.

E.1.3.3 Input / Output control

convert (u) @ sends a convert message to the parent (the operating system) who is then expected to print the specified bs-area. A file with scope login is not accepted and the file must not be in use. A temporary file converted will immediately disappear from the reach of the job. Each convert operation performed by BOSS requires a cbuffer which must be reserved in the job specification.

end (u) * returns current input to previous current input at the positions where it was left.

i (u) * selects a new file as current input. The former file may later be resumed at the position where it was left (for instance by a call of *end*).

o (u) * selects a new file as current output.

E.1.3.4 Name base control

base (u) * changes the catalog base of the job process, thereby changing the name base, and displays the process bases of the job process.

E.1.3.5 Resource control

claim (u) * lists selected parts of the bs area claims of the process together with area, buf, size and first core.

claimtest (u) checks the claims of the calling process according to the call parameters and leaves the ok bit true if the claims specified are present, false otherwise.

E.1.3.6 Device control

change (m) @ sends a change-paper-message to the parent (operating system). The program is only used when a job executed under BOSS uses job controlled printer.

convert (u) @ sends a convert message to the parent (the operating system) who is then expected to print the specified backing storage area. A file with scope login is not accepted and the file must not be in use. A temporary file converted will immediately disappear from the reach of the job. Each convert operation performed by

BOSS requires a cbuffer which must be reserved in the job specification.

deletelink (m) * removes links, i.e external processes with links to LAN device handlers.

disccopy (m) * is primarily intended for binary copying of discpacks or parts of discpacks from one discpack to another, and save and load of bs files, using discpacks as backup medium.

kit (u) @ sends a mount disk message to the parent (the operating system) demanding a disk kit with a specified name to be mounted on a specified disk unit.

makelink (m) * creates links, i.e external process with links to LAN device handlers on LAN controllers.

mount (u) sends a mount message to the parent (the operating system) who is then expected to ask the operator to mount the tape reel. The program does not await the mounting, unless there is asked for mounting of an unspecified worktape.

mountspec (u) @ sends a mount special message to the parent (the operating system) limiting a later mounting of the specified magnetic tape reel to the station with the specified device number.

release (u) * sends a release message to the parent (the operating system) releasing the specified magnetic tape reel.

rewind (u) * sends a rewind operation to a magnetic tape document and returns.

enable/ring (u) **sends** a write enable message to the parent (the operating system). *ring* is rarely used as the software sends the write enable message automatically when needed.

suspend (u) @ sends a suspend message to the parent (the operating system) asking for suspension of the specified magnetic tape reel. This is relevant for worktapes only. The station is now available for mounting of another tape reel, but the suspended worktape is still reserved for the job until it terminates or releases the tape reel.

unload (u) * sends an unload operation to the magnetic tape process, allowing the job process to continue while the tape is unloading.

E.1.3.7 Operator interaction control

mount (u) sends a mount message to the parent (the operating system) who is then expected to ask the operator to mount the tape reel. The program does not await the mounting, unless there is asked for mounting of an unspecified worktape.

mountspec (u) @ sends a mount special message to the parent (the operating system) limiting a later mounting of the specified magnetic tape reel to the station with the specified device number.

opcomm (u) sends the parameter list in the call as a print message to the parent (the operating system) with request for an answer from the operator and types the answer on current output.

opmess (u) sends the parameter list in the call as a print message to the parent (the operating system).

E.1.3.8 Accounting

account (u) @ sends an account message to the parent (operating system) who is then expected to produce a record in the account file. Only used when jobs running under BOSS wants to produce special account information.

E.1.4 System Maintenance Programs

checkio (m) * may supervise all actions on a particular document.

discstat (m) * is a diagnistic tool for printing statistical information maintained by a disk driver.

disctell (m) * may (a) list the physical disk described in the monitor, (b) list the logical disks located on a given physical disk, and (c) list the logical disk and the logical slice and segment number of a given segment number of a given physical disk together with a description of a possible area in which the segment belongs.

do (m) * The do language is intended for monitor and hardware testing and supervising, but it is designed so that it can be used as a general programming language , in which the user can register and core store contents, execute monitor and FP procedures as well as slang instructions, output any information about the core store in an easy readable form.

mainstat (m) * is a diagnostic tool for printing statistical information and test output from an RC8000 or an RC9000-10 main process.

montest (m) * is used to display monitor tables and other data structures in the monitor.

movedump (m) * moves the coredump caused by autoload action from the system dump area on the autoload disk to a backing storage file.

printzones (m) * is intended for inspection of runtime system variables and/or zone and share descriptors, context block descriptors and activity descriptors in ALGOL and FORTRAN programs for debugging purposes.

scatup (m) handles the s-user catalog `susercat`. The program may create a new s-user catalog, insert entries in the s-usercatalog and list the contents of entries or all of the s-user catalog.

slicelist (m) may provide you with a survey showing the slices constituting a backing storage file and how these are distributed on cylinders.

E.1.5 LAN Maintenance Programs

ftscatup * is used to maintain the FTS catalog.

lanstat * gathers and displays information about activity on the LAN.

E.1.6 Moving Data Across The LAN

rr (fts) * transfers files from an FTS server to an FTS user.

wr (fts) * transfers files from the user to the server.

E.2 Frequently Used Utility And Maintenance Programs

backfile

Substracts one from the file number (unless it is 0) in the tails of the entries specified and signals reach of file 0.

Examples

If the catalog entries `old` and `new` describe file 4 of magtape mt310514 and file 2 of mt310515 respectively, then the command

```
backfile old new
```

will change old to describe file 3 of mt310514 and new to describe file 1 of mt310515. A repeated call will change old to decribe file 2 and new to describe file 0 and set the warning bit to yes. A following call will change old to descibe file 1 and leave new unchanged - the ok bit is set to no.

If the entry t describes file no. 7 on magtape mt310514 the call

```
backfile t t t t t t
```

will make it describe file no. 1.

Call

```
backfile { <name> }1-*
```

base

The program changes the catalog base of the job process according to the parameters in the call.

The catalog, standard, user and max bases of the process may be listed on current output before as well as after the change of catalog base.

The catalog base may be changed to any interval which is contained in (or equal to) the max base and which also contains the standard base, equals it or is contained in the standard base.

Example 1

The call

```
base user
```

will change the catalog base to equal the user base.

Example 2

The call

```
base abs 1532 1584
```

will change the catalog base to the interval 1532, 1584, provided it is a legal catalog base for the process.

Example 3

The call

```
base std what
```

will make the catalog base return to the original state and show you the actual bases.

Call

```
base (what)0-* [<specifier>] [<modifier>] [what]

<specifier> ::= std / user / max / abs
<modifier>  ::= <modif1> [<modif2>]
<modif1>    ::=
<modif2>    ::= <integer> / <int1>.<int2>
<int1>      ::=
<int2>      ::= <integer>
```

basemove

The program basemove changes the entry base of all catalog entries of a specified name base to another entry base, or the entry base of entries specified by name and base to another entry base.

Example

The entry base of the entry areal is changed from (700, 740) to (790, 800) by this call:

```
*basemove name.areal from.700.740 to.790.800
```

and the program will list on current output file:

```
areal  bases moved properly  from. 700  740 to. 790  800
  1 entry found and moved ok
```

Call

```
{outfile =} basemove  ,
name.[<name>] {from.<low>.<up> to.<low>.<up>}1-*
```

where

```
<name> ::= name of a catalog entry
<low>  ::=
<up>   ::= integer
```

binin

The program can input files generated by the program *binout*, verifying the contents of the file by checksum control.

Example

A magnetic tape file, described in the catalog by the entry t, was produced by the FP command:

```
t=binout fup
```

It may be loaded by the FP command

```
binin t
```

and thereby the catalog entry fup, the area and its contents are reestablished.

Call

```
[<outfile> =]
binin [list.(yes/no)] [disc.<disk>] ,
<binout file> [.<modifier>]

<outfile>          ::= <name of output file>

<disk>             ::= (<name> )
                       (0/1/2/3)

<name>             ::= <name of a disk>

<binout file>      ::= <name of input file>

                       (   <binout segments> )
<modifier>         ::= ( s.<binout segments> )
                       ( c.<binout segments> )

<binout segments>::=
(<no of binout segments> [.<first binout segment>])

<no of binout segments>::=
<first binout segments>::= integer
```

binout

The program can output catalog entries and contents of files in a format (a binout file) which may be input by the program *binin*. The output file is added checksum information for verification by *binin*.

Example

The program file named *fup* is output on magnetic tape file, described by the entry t, by the FP command

```
t=binout fup
```

(compare with the example under the program *binin*).

Call

```
<outfile> = binout <input description>

<input description> ::= <name> {modifier} 0-*

                        {.p          }
                        {.b  .<halfs>}
                        {.s  .<field>}
<modifiers>         ::= {.a  .<field>}
                        {.np         }
                        {.ne         }

<bytes>             ::=     <no of halfwords>

                            {<no of blocks>                 }
<field>             ::=     {                               }
                            {<no of blocks> . <first block>}


<no of blocks>      ::=
<first block>       ::= integer
```

cat

The program works as catsort, except no sorting of entries prior to output takes place, i.e. the entries specified in the call are listed in the order in which they are found in the catalog.

Example 1

The call

```
cat scope.system
```

will list all entries in the main catalog of scope system in the order they are found in the catalog.

Example 2

The call

```
cat
```

will list all entries with entry base inside or equal to project base from the main catalog in the order they are found.

Example 3

The call

```
cat cat.disc3 name.pip docname.pip
```

will list all entries in the catalog with the document name `disc3` which have either the name `pip` or the document name `pip`.

Call

Exactly as for catsort, except that in spite of sorting parameters, no sorting takes place.

catsort

Lists on current output selected parts of the main catalog (or any auxiliary catalog) sorted according to parameters. At last also the total number of entries and segments are listed.

Example 1

The FP command:

```
catsort base.project.min
```

will list all entries with a base contained in the project base, e.g. belonging to the actual project. The parameter min causes that only name, segments, docname, date, and scope is output.

Example 2

The FP command:

```
catsort scope.project min.yes
```

will do the same, but only permanent entries with an entry base equal to the project base are output.

Example 3

The FP command:

```
catsort
```

will list all entries with a base inside or equal the to project base from the main catalog sorted according to base and entry name.

Call

```
                              {<catalog spec>} 0-*
[<outfile>= ]     catsort     {<entry spec>  }
                              {<sorting spec>}

<catalog spec> ::=  { main.{yes/no}           }
                    {                         }
                    {      {yes/no          }}
                    { cat. {<integer>       }}
                    {      {<document name>}}
                    {      {main            }}
```

```
<entry spec>    ::=  {         { only  }        }
                     {system.  { yes/no}        }
                     {                          }
                     {name.    <entry name>     }
                     {docname.<document name>   }
                     {                          }
                     {base.   {<scope>    .min }}
                     {        {<lower>.<upper>}}
                     {                          }
                     {scope.  {<scope>    .min }}
                     {        {<lower>.upper>  }}
                     {                          }
                     {before. <clock>           }
                     {after.  <clock>           }
                     {size.   <lower>.<upper>   }
                     {cont.   <lower>.<upper>   }
                     {min.    {yes/no}          }

<sorting spec.>::=   {basesort } .  {yes/no}
                     {docsort  }
                     {slicesort}
                     {sort     }

                     {system }
                     {project}
                     {user   }
<scope>         ::=  {login  }
                     {temp   }

<lower>         ::=  <integer>
<upper>         ::=  <integer>

<clock>         ::=  <yymmdd>.<hhmmss>
<yymmdd>        ::= <integer>
<hhmmss>        ::= <integer>
```

changeentry

Changes an existing catalog entry according to the parameters in the call. The program is a supplement to the program *set* and *entry* and is used when one wants to change some of the elements in the entry tail by copying from the tails of other catalog entries.

Example

Suppose that the catalog entry named `source` contains the name of a magnetic tape reel in the document name field.

By the FP commands

```
filex=changeentry filex source filex filex filex filex filex
```

the entry filex is changed to contain the name of the tape reel.

The catalog entry named `source` containing the name - say mt471100 - may be created by a call of *set*:

```
source = set mtl6 mt471100
```

Call

```
                                                              0-4
<newname> = set [<kind> [<docname> [<date> (<word>)   ] ] ]

<newname>          ::= name
<name>             ::= name, apostrophized name,
                       generalized name or general text

(<kind>    )       ::= (<integer>              )
(<docname>)            (<integer>  .<integer>)
                       (<name>                 )

<date>             ::= (<integer>              )
                       (<integer>  .<integer>)
                       (<name>                 )
                       (<name>     .<name>     )
                       (d.<isodate>.<clock>    )

<word>             ::= (<integer>              )
                       (<integer>  .<integer>)
                       (<name>                 )
                       (<name>     .<name>     )

<isodate>          ::= (<yymmdd>   / 0)

<clock>            ::=  <hhmm>

<yymmdd>           ::=
<hhmm>             ::=  <integer>
```

checkio

Checkio may supervise all actions on a particular document. That is performed in the following way: Assume that checkio is executed in a job process called *dev*. Then all messages sent to dev are passed on by checkio to the document supervised. The answer from the document is passed back to the original sender process, which seems to be handling the document in the normal way.

Checkio can easily print all messages and answers as they are passed on, and you can later find out about parity errors from the document, rereading etc.

Call

The process dev must be created with a size big enough to store the data blocks passed by.

Checkio must be called with one parameter specifying the name of the document to be supervised. The messages and answers are printed on the current output of dev. A console communication to start dev may look like this:

```
to s
new dev size 5000 run; if monitor older than 8.0
                      ; mode 0 also
to dev
o lp                  ; print on the line printer
checkio t6            ; check the document t6
```

You will then see nothing from dev until some messages are sent to it, for instance from another job doing like this:

```
t=set mto dev 0 6     ; edit to the 'magnetic tape'
                      ; dev,
t=edit infile         ; which acts as the magnetic
                      ; file t6.
```

Dev will now start listing the communication on the line printer. When you have finished using dev for supervising of t6, you can proceed like this:

```
to s
proc dev break        ; the remaining output from
                      ; dev appears on the printer.
*** fp break
checkio printer       ; select a new document etc.
```

claim

Lists claims of specified processes.

Examples

In an installation with the 2 disks:
`disc and disc1`

the call

```
claim
```

may print:

```
name : fgs       area : 10  buf : 12  size : 200000  first 149790

disc: 84 segm/slice
  temp    1596 segm   19 slices    20 entr
  login    924 segm   11 slices    21 entr
  perm     924 segm   11 slices    21 entr

disc1: 42 segm/slice
  temp      42 segm    1 slices
  login     42 segm    1 slices     1 entr
  perm      42 segm    1 slices     1 entr
```

The call

```
claim key.disc
```

would print:

```
name : fgs       area : 10  buf : 12  size : 200000  first 149790

disc: 84 segm/slice
  key 0   1596 segm   19 slices    22 entr
  key 1    924 segm   11 slices    22 entr
  key 2    924 segm   11 slices    21 entr
  key 3    924 segm   11 slices    21 entr
```

and the call:

```
claim perm.disc temp
```

would print:

```
name : fgs       area : 10  buf : 12  size : 200000  first 149790

disc: 84 segm/slice
  perm     924 segm   11 slices     21 entr

disc: 84 segm/slice
  temp    1596 segm   19 slices     20 entr

disc3: 42 segm/slice
  temp      42 segm    1 slices
```

Call

```
                                                          0-* 0-*
[<outputfile>=]  claim ([<process>] (<spec>)   )

<outputfile> ::= <name of file>

<process>    ::= <name of internal pricess> / all

<spec>       ::= <disk> / <scope> / <scope>.<disk> /
                 <disk>.<scope> / <other>

<disk>       ::= <name of disk>

<scope>      ::= temp / login / perm / key

<other>      ::= area / buf / size / first
```

clean

The program removes all catalog entries with catalog base within the specified limits.

Example

The FP call

```
clean 2800 2899
```

removes all catalog entries with catalog base within the limits 2800 2899.

Call

```
clean <lower limit> <upper limit>
```

where

```
<lower limit> ::=
<upper limit> ::= <integer>
```

clear

Removes catalog entries with name and scope as specified.

Example

By the FP command

```
clear user text4
```

the catalog entry (if any) with scope user and name text4 is removed from the catalog. A catalog entry with the same name but another scope is not affected.

Call

```
                                  1-*
clear <scope spec>  {<name>}

<scope spec>  ::= <scope> [.<disk name>]

<scope>       ::=  { temp    }
                   { login   }
                   { user    }
                   { project }

<disk name>   ::= <name of disk>
```

copy

Copies one or several text files into another file and calculates the number of characters copied and the sum of their ISO values. Blind characters (ISO values 0 and 127) are not copied. Furthermore the program may be used for check reading of text files.

Example

The text files `text1 text2` are output as one magnetic tape file `text`, by the FP command

```
text=copy text1 text2
```

and the number and the sum of the characters are printed on current output. One may then check the file by reading it in a job by the FP command

```
copy text
```

Call

```
[ <outfile>= ]  copy [list.(yes/no)]    ,
                                                    1-*
     ( <infile>                                    )
     ( <lines>                                     )
     ( [<infile>]  <count>                         )
     ( message.(yes/no)                            )

<infile>         ::= <name>
<count>          ::= <iso value>.<appearances>
<lines>          ::=
<appearances>    ::= <integer>
<iso value>      ::= (<integer>/<letter>)
```

correct

The program corrects specified words in the backing storage file according to the parameters. The program may also be used to print specified bits as integers.

Example

The FP call:

```
correct bsfile.4                          ,
addr.0 bits  0.11  if   700 then -456,
       bits 12.23  if -1234 then 4000,
addr.8             if     0 then    1
```

will make the following corrections on segment 4 of `bsfile`:

```
halfword 0   is changed to -456 (in case it is   700)
-        1   -  -        -  4000 (-  -    -  -  -1234)
-        8:9 -  -        -     1 (-  -    -  -      0)
```

No corrections are made if <oldvalue> is not correct in all cases.

Call

```
correct <bsfile>.<segmno>  ,

{ address.<addr> {[<bitspec>] if <oldvalue>  ,
                          then <newvalue>}1-*  }0-*

<bitspec>      ::=  bits.<firstbit>.<lastbit>

{ <oldvalue> } ::= { <integer>          }
{ <newvalue> }     { negative.<integer> }

{ <segmno>   } ::= <integer>
{ <addr>     }
{ <firstbit> }
{ <lastbit>  }
```