RCSL No: 52-AA1156

Edition: January, 1983

Author: Bo Bagger Laursen

Title:

RC3502 Operating Guide

RCSL No 42-i 2164

A/S REGNECENTRALEN
af 1979

Keywords:

RC3502, OPSYS Commands, Debug Commands, Autoloading.

Abstract:

This manual describes how to operate the RC3502: Power-on, power-off, commands to the operating system OPSYS, operation of the DEBUG console, autoloading.

(This manual replaces RCSL: 52-AA1016).

(58 printed pages).

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL No 42-I 2164

TABLE OF CONTENTS PAGE

1. SWITCHES AND INDICATORS 1
1.1 Operator´s Control Panel 1
1.2 Processor Front Panel 2
1.2.1 Switches 3
1.2.1.1 Bus Switches 4
1.2.1.2 Mode Switch 4
1.2.2 Indicators 4
1.3 Power Supply 6

2. CONSOLE OPERATION 8
2.1 Terminal Mode 8
2.1.1 Operator Process 8
2.1.2 OPSYS Commands 9
2.1.3 Messages from OPSYS 17
2.1.4 Messages from LOADER 19
2.2 Debug Mode 20
2.2.1 Activation 20
2.2.2 Display Commands 20
2.2.3 Control Commands 22
2.2.4 Command Parameters 22

3. AUTOLOADING 23
3.1 Autoload Switch Format 23
3.2 Autoload Messages 28
3.3 Generating Autoload Files 30
3.3.1 Generating a Paper Tape Autoloadfile 31
3.3.2 Generating an FPA/IFP Autoloadfile 31
3.3.3 Generating TES202 Eproms 31

4. ERROR PROCEDURES 32
4.1 Monitor Testoutput 32
4.2 Boot Testoutput 32

APPENDICES:

A. REFERENCES 35

B. OPSYS COMMANDS 36

C. AUTOLOAD SWITCH LAYOUT 38

D. SECRET VECTOR LAYOUT 39

E. INSTRUCTION CODES 40

F. INSTRUCTION MNEMONICS 41

G. PROCESS INCARNATION DESCRIPTOR LAYOUT 42

H. MESSAGE HEADER LAYOUT 43

I. EXCEPTION CODES 44

J. WORKING REGISTER LAYOUT 45

TABLE OF CONTENTS (continued) PAGE

K. CONTROL MICROPROCESSOR RAM LAYOUT 47

L. INSTALLATION STANDARDS AND RECOMMENDATIONS 48
L.1 Input/Output Modules 48
L.2 Memory Modules 48

M. CATCHWORD INDEX 51

1. SWITCHES AND INDICATORS 1.

1.1 Operator´s Control Panel 1.1

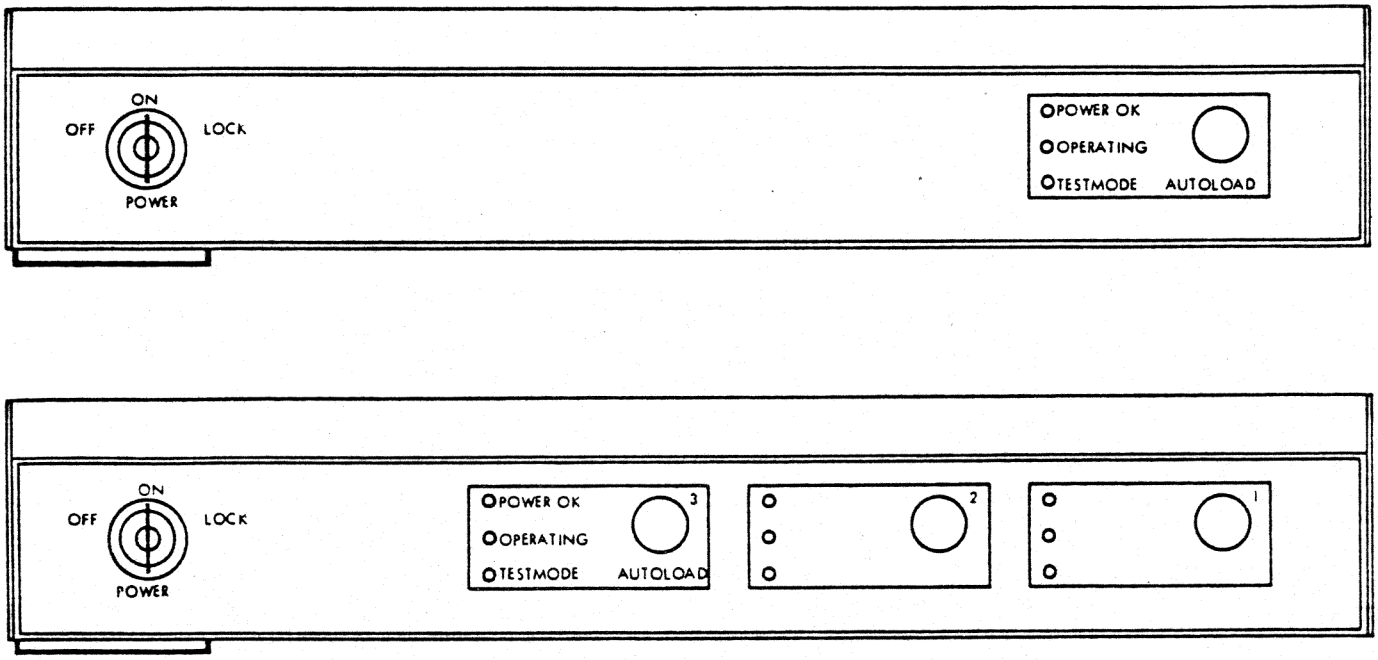


Fig. 1. OCP for Rack with One RC3502 or Three RC3502.

Power off of the RC3502(s) is done by turning the power key to the OFF position.

Power on of the RC3502(s) is done by turning the key to the ON position (or further on to the LOCK position).

The AUTOLOAD button(s) is (are) enabled when the key is in the ON position, and disabled, when in the LOCK position.

The AUTOLOAD button initiates autoloading of the RC3502 in question.

The POWER OK indicator is illuminated during power OK condition on the RC3502.

The OPERATING lamp indicates that the RC3502 is running normally.

The TEST MODE lamp indicates that the RC3502 is executing the built-in test programs.

1.2 Processor Front Panel 1.2

The front panel of the processor board contains five switches, five indicators, and a jack.

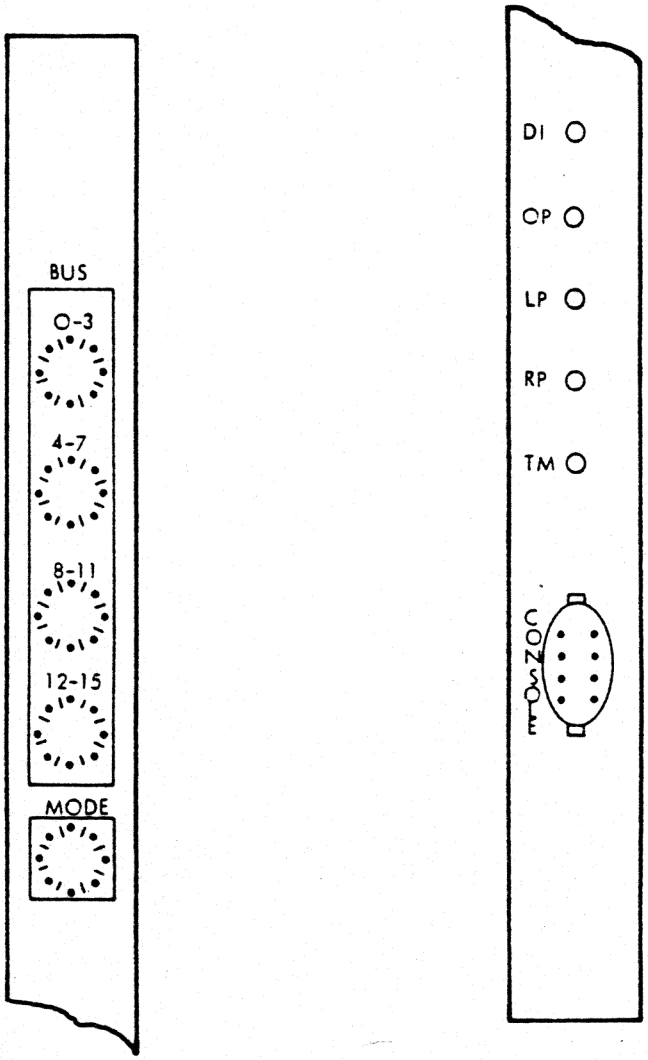


Fig. 2. Processor Front Panel, Switches and Indicators.

1.2.1 Switches

1.2.1

All of the switches are rotary switches with 16 positions, indicated by the hexadecimal numbers 0 to F. The switches are set by means of a screwdriver.

1.2.1.1 Bus Switches

The four switches marked BUS are used to supply the processor with data. There is a switch for bits 0 to 3, 4 to 7, 8 to 11, and 12 to 15.

1.2.1.2 Mode Switch

The switch marked MODE is used to control the baud rate for the console and the execution of the built-in test programs (sect. 1.2.2).

If the mode switch is equal to, or greater than 8, the console is locked to Terminal-mode (T-mode), i.e. the console will not switch to Debug-mode (D-mode) by activating the BELL key (CTRL G). The mode switch is only read after power restart.

Settings	Baud Rate	Execution Mode
0 (8)	300 bps	run test, loop
1 (9)	1200 bps	run test, loop
2 (A)	300 bps	skip test
3 (B)	1200 bps	skip test
4 (C)	300 bps	run test, no loop
5 (D)	1200 bps	run test, no loop
6 (E)	300 bps	skip test
7 (F)	1200 bps	skip test

Test Program Execution Modes

run test — The test programs are executed whenever the autoload button is pressed.

skip test — The test programs are not executed.

loop — The test programs are executed in an endless loop.

no loop — The test programs are executed once.

1.2.2 Indicators

DI Disabled Interrupt
This lamp, when lit, indicates that the processor is running in the disabled interrupt mode.

OP Operating
This lamp, when lit, indicates that the processor is running normally; when it is extinguished, the processor has stopped.

LP Left Parity Error
This lamp, when lit, indicates that a parity error has been detected during a memory read in the left byte. The lamp can be extinguished only by autoloading.

RP Right Parity Error
This lamp, when lit, indicates that a parity error has been detected during a memory read in the right byte. The lamp can be extinguished only by autoloading.

TM Test Mode
This lamp, when lit, indicates that the processor is executing the built-in test programs. The current program is indicated by the OP, LP, RP, and TM lamps, TM representing the least significant bit of the program number.

If an error is detected by a test program, one of the following messages is displayed on the console:

1 8085 Communication Test
Message: ERR 1 <dummy><dummy>
Y5D gives 6 bytes of transmitted data.
Y70 gives 6 bytes of received data.

3 Communication Test
Microprogram interrupt of control microprocessor.
No message is displayed, but RP and TM are lit.

5 Working Register Address Test
Message: ERR 5 <address><data read>
W <address> gives data read.

7 Working Register Data Test
Message: ERR 7 <address> <data read>
W <address> gives data read.

9 Memory Address Test
Message: ERR 9 <address.displacement> <data read>
<err.type>
Y40 gives memory module number.
M <memory module.displacement>
gives data read.
err.type: + 1 = left parity error
+ 2 = right parity error
+ 4 = dataerror

B Memory Data Test
Pattern 5555.
Message: ERR B <address.displacement> <data read>
<err.type>
Y40 gives memory module number.
M <memory module.displacement>
gives data read.

D Memory Data Test
Pattern AAAA.
Message: ERR D <address.displacement> <data read>
<error type>
Y40 gives memory module number.
M <memory module.displacement>
gives data read.

1.3 Power Supply

1.3

The power supply can be either POW201 or POW204.

The power supply (POW201) is supplied with a number of controls:

AC MAINS:	Manual and automatic circuit breaker.
LOCK:	Spring loaded lock switch, which inhibits manual intervention of AC MAINS and AUTO functions.
POWER OK:	Indicator which is illuminated during power OK condition.
AUTO:	Push button for manual generation of an autoload signal.

TEMP: Indicator which remains on after a power break caused by an overheating condition.

The power supply POW204 is supplied with the following controls:

POWER: Circuit breaker, lit when power on.

POWEROK: Indicator which is illuminated during power ok condition.

POWER FAILURE:
OVER-TEMPERATURE:
OVER-VOLTAGE: Error indicators which are illuminated after an error condition. These indicators are reset after activating the circuit-breaker, or after activating the RESET push-button.

RESET: Push-button for manual generation of an autoload signal and a reset of the error indicators.

2. CONSOLE OPERATION

The console may be in one of two possible modes: Debug-mode (D-mode) or Terminal-mode (T-mode). A switch between the two modes takes place when the BELL key (CTRL G) is activated.

2.1 Terminal Mode

The console may work as terminal for the RC3502 software system, while in T-mode.

An operator process coordinates the communication between the software system and the operator.

2.1.1 Operator Process

Input and output messages to the operator are identified by a name. The operator process holds a variable ´current name´, which identifies the current process incarnation for input or output.

´current name´ is updated in the following situations:

1. (ESC) <name> (NL)

   The input line contains a name which is assigned to ´current name´. The operator searches its queue of pending input messages for a message with <name> as identification. If at least one message is found, it is activated. If it is a reactivation, the old input is repeated.

   If no message is found, BELL is echoed.

   Note: (ESC) (NL) is attention to ´current name´.

2. By output.

   If the output message has name = ´current name´, the output message is printed on the console.

   If the output message has name <> ´current name´, ´current name´ is updated and

   > "current name"

is printed followed by the text from the output message.

3. (ESC) ? (NL)

Prints the identifications of all pending input messages. ´current name´ is updated to the last name in the printout.

The operator has a number of facilities for controlling output and for editing purposes:

(CTRL)	E	- deletes the whole line
(BS)		- deletes the last character
(ESC) (NL)		- repeats the whole line
(RUBOUT)		- deletes the last character. ´<-´ is echoed.
(CTRL)	S	- stops output.
(CTRL)	Q	- starts output.
(CTRL)	O	- skips output. The function is modulo 2 and is reset to ´no skip´ by ESC .

Note: (NL) may be Carriage Return or Line Feed.

2.1.2 OPSYS Commands

2.1.2

OPSYS interprets the following commands. The underscored characters are sufficient for the interpretation.

More than one command may be typed on one line, unless the syntax is terminated by <nl>.

Whenever <incarnation> or <process> is listed - unless otherwise stated - we refer to children of ADAM with heading:

```
PROCESS pip (VAR sv: system_vector);
```

In the following all numbers are decimal, unless otherwise stated.

BREAK <incarnation>

The child <incarnation> is broken with the current value of excode as exception code.

The child may be any incarnation in the incarnation tree.

E.g.: BREAK S

CHECK <module no>

performs a CRC16 check of the memory module <module no>. If an error is detected, the word address interval, the expected, and computed checksum are printed.

E.g.: CHECK 25

CREATE <incarnation> { AS <process> } 1
                     { }   0

creates an incarnation of the process <process>. If <process> is omitted the process is supposed to have the same name as the incarnation. The size of the stack is the current value of SIZE (see the SIZE command).

E.g.: CREATE T1 AS TEST CREATE T2 AS TEST

DATE { <year>.<month>.<day> <hour>.<minute>.<seconds> } 1
     { }   0

If the parameters are included the date is initialized. The command always responds with the current date.

E.g.: DATE 83.01.04 15.30.20

EXCODE <integer>

initializes the current value of excode to <integer>.

Default: excode = -1.

E.g.: EXCODE 47

FREE lists the free areas in the RAM modules. The start displacement and size in words of the holes are listed, besides the number of holes, the minimum hole, the maximum hole and the sum of the holes per module.

FROM {PTR / FPA}

initializes the current value for load kind to PTR or FPA. Load kind controls the kind of driver used for dynamic load (see LOAD). Default load kind is taken from the hexadecimal switches.

E.g.: FROM PTR

HELP lists the available OPSYS commands.

IN <inchannel>

initializes the current value of the I/O channel used for load. Default <inchannel> is taken from the hexadecimal switches.

E.g.: IN 82

KILL <incarnation>

works as the REMOVE command.

LINK <process>

links a process with name <process> to a process declaration in ADAM.

E.g.: LINK CPUSE

LIST {<incarnation>}n {<incarnation>}0 <nl>

lists the incarnation tree with root <incarnation>. If <incarnation> is missing 'adam' is taken as root.

E.g.:

```
LIST OPSYS TIMER OPERATOR
incarnation depth level prio state   size     stack      father
opsys         0     0    1   run     512   00c8.e660   adam
incarnation depth level prio state   size     stack      father
timer         0     2        wait    157   00c8.df3a
incarnation depth level prio state   size     stack      father
operator      0     0    1   wait    190   00c8.e484   adam
console       1     0    1   wait    201   00c8.efb8   operator
debugout      2     4        wait     98   00c8.f2b0   console
debugin       2     3        wait     98   00c8.f1ac   console
```

LOAD $\left\{\begin{matrix}\\ \text{<program>}\\ \\ \end{matrix}\right\}^{n}_{0}$ <nl>

loads the programs from an external device of kind ´current load kind´ (see the FROM command) in the I/O channel ´current inchannel´ (see the IN command). If no programs are specified, all appearing programs are loaded.

E.g.: FROM PTR IN 82 LOAD PRINTCHAR PRINTNL

Binary relocatable papertapes for load from PTR are produced like

```
JOB A 1 DEVICE PUNCH
TPN = PUNCH16 MODE.CRC16 {Bxxxx}n
                         }1
FINIS
```

where Bxxxx is the binary output from the PASCAL80 compiler.

A binary relocatable file for load from FPA is produced like

```
JOB A 1
xxxx = PUNCH16 MODE.CRC16 {Bxxxx}n
                          }1
```

```
SCOPE yyyy xxxx

FINIS
```

where Bxxxx is the binary output from the PASCAL80 compiler.

The file xxxx is loaded by e.g.:

```
JOB A 1 DEVICE 18

main35001 = autoload xxxx start.no

FINIS
```

LOOKUP {<program>}n <nl>
       {   }1

makes a lookup in the LINKER catalog for the listed programs.

E.g.:

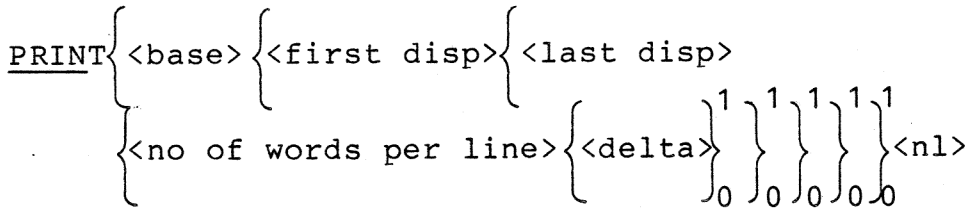
```
LOOKUP CREATE LINKER OUTINTEGER
FUNCTION    create       1982.11.23 13.58  1 12000   768    34  5
PROCESS     linker       1982.11.29 08.32  1 12000  3238   358  1
PROCEDURE   outinteger   1982.11.23 13.58  1 12000   300    23  3

  1)          2)                 3)       4) 5)    6)     7)  8)
```

The output has the interpretation:

1. program kind
2. program name
3. date of compilation
4. number of code pages
5. pagesize in bytes
6. last page size in bytes
7. default appetite (create size) in words

8. number of parameters



outputs the specified memory area with a fixed format (see the example): <base>, <first disp>, and <last disp> are hexadecimal.

<delta> defines the default increase of <first disp> if <last disp> is not specified. After the PRINT command <first disp> := <last disp> + 2.

E.g:

```
PRINT C8 E660 E760
 address
00c8.e660  00c8    200     0 200
00c8.e662  e661  -6559   230  97    a
00c8.e664  0000      0     0   0
00c8.e666  0000      0     0   0
00c8.e668  00c8    200     0 200
00c8.e66a  e6ae  -6482   230 174
00c8.e66c  00c8    200     0 200
00c8.e66e  d7ed -10259   215 237
00c8.e670  4000  16384    64   0   @
00c8.e672  e7eb  -6165   231 235
00c8.e674  ffff     -1   255 255
00c8.e676  00c0    192     0 192
00c8.e678  5fc9  24521    95 201   _
00c8.e67a  00c0    192     0 192
00c8.e67c  aeba -20806   174 186
00c8.e67e  ea5d  -5539   234  93    A
00c8.e680  0020     32     0  32
00c8.e682  e7f9  -6151   231 249
00c8.e684  e7eb  -6165   231 235
00c8.e686  00c0    192     0 192
00c8.e688  46ce  18126    70 206   F
00c8.e68a  0000      0     0   0
00c8.e68c  ea5f  -5537   234  95    _
00c8.e68e  00c8    200     0 200
00c8.e690  e13e  -7874   225  62    >
00c8.e692  00c8    200     0 200
00c8.e694  e74e  -6322   231  78    N
00c8.e696  00c8    200     0 200
00c8.e698  e73e  -6338   231  62    >
00c8.e69a  4000  16384    64   0   @
00c8.e69c  e77a  -6278   231 122    z
00c8.e69e  00c8    200     0 200
00c8.e6a0  eeec  -4372   238 236
00c8.e6a2  4000  16384    64   0   @
00c8.e6a4  0000      0     0   0
00c8.e6a6  4000  16384    64   0   @
00c8.e6a8  e661  -6559   230  97    a
```

```
00c8.e6aa  4000  16384   64   0  @
00c8.e6ac  0000      0    0   0
00c8.e6ae  4000  16384   64   0  @
00c8.e6b0  e3a0  -7264  227 160
00c8.e6b2  4000  16384   64   0  @
00c8.e6b4  0000      0    0   0
00c8.e6b6  00c0    192    0 192
00c8.e6b8  00a0    160    0 160
00c8.e6ba  00c0    192    0 192
00c8.e6bc  6156  24918   97  86  a V
00c8.e6be  6f70  28528  111 112  o p
00c8.e6c0  7379  29561  115 121  s y
00c8.e6c2  7320  29472  115  32  s
00c8.e6c4  2020   8224   32  32
00c8.e6c6  2020   8224   32  32
00c8.e6c8  2020   8224   32  32
00c8.e6ca  00c8    200    0 200
00c8.e6cc  e074  -8076  224 116     t
00c8.e6ce  00c8    200    0 200
00c8.e6d0  e0e6  -7962  224 230
00c8.e6d2  4000  16384   64   0  @
00c8.e6d4  e3a0  -7264  227 160
00c8.e6d6  5555  21845   85  85  U U
00c8.e6d8  aaaa -21846  170 170
00c8.e6da  4000  16384   64   0  @
00c8.e6dc  0000      0    0   0
00c8.e6de  00c0    192    0 192
00c8.e6e0  0054     84    0  84     T
00c8.e6e2  00c8    200    0 200
00c8.e6e4  e6e6  -6426  230 230
00c8.e6e6  00c8
```

PRIORITY <integer>

initializes the current value for priority used by the START or RUN command.

Default: priority = -3.

E.g.: PRIORITY -1

REMOVE <incarnation>

removes the child <incarnation> of ADAM and the associated subtree.

E.g.: REMOVE S

RUN <incarnation> { AS <process> }1 / 0

links, creates, and starts an incarnation with name <incarnation> of the process <process>.

If <process> is omitted, the process is supposed to have the same name as the incarnation.

E.g.: RUN T1 AS TEST RUN MIRROR

SIZE <integer>

initializes the current value of SIZE used when creating ADAM children.

Default: SIZE = 0.

(Note: SIZE = 0 will trigger the use of the default create size for the program (see the LOOKUP command)).

E.g.: SIZE 763

START <incarnation>

starts the ADAM child <incarnation> with the current value of PRIORITY as priority.

E.g.: START S

STOP <incarnation>

stops the ADAM child <incarnation>.

E.g.: STOP S

UNLINK <process>

unlinks a process with name <process> from a process declaration in ADAM.

E.g.: UNLINK CPUSE

UNLOAD { <program> }n <nl>
       { <program> }1

deletes the programs from the LINKER catalog, if the programs are not referenced by other programs. If other programs become not referenced after the delete, these programs are also deleted.

E.g.: UNLOAD CPUSE TEST

2.1.3 Messages from OPSYS

The rest of the command line is skipped if any of the following messages appear:

*** loader not ready

- the LOADER is not included in the system or is unable to run owing to lack of memory.

*** command not implemented

- the command is not available in this version of OPSYS

*** syntax error

- misspelling of a command

*** processname missing

*** unknown incarnation

*** unknown process

*** processname busy

- incarnations of this process still exist

*** incarnationname missing

*** name in use

*** no free processdeclarations

- you must release a process declaration in ADAM by the command UNLINK

*** process not loaded

- the LINKER catalog does not contain the stated process

*** process parameters not equal

- a process with the stated name exists in the LINKER catalog, but the parameter list does not fulfil the declaration

```
PROCESS pip (VAR sv : system_vector);
```

*** size too small or too large

- use the SIZE command to adjust the SIZE parameter

*** process not linked

- use the LINK command

*** unknown program

- the program is not in the LINKER catalog. The program may be loaded by the LOAD command

*** program busy

- the program is still accessed by other programs in the system

2.1.4 Messages from LOADER

*** install more ram memory

- the LOADER cannot get enough memory to run

*** loaddriver no stack

- the driver cannot be created due to lack of memory

*** level reservation trouble

- the interrupt level requested for load is occupied by another process incarnation in the system

scan no: x from fpa in xxxx

- initialize load from RC8000 when the first scan is announced (Note: xxxx is decimal). Later scans are performed by the loader itself with no need for operator assistance.

scan no: x from ptr in xxxx

- place the paper tape in the reader whenever a scan is announced

expected: xxxx
received: xxxx

- the crc16 data check reports an error. The programs should be reloaded

end loader

- normal finis message from the LOADER. A list of the loaded programs is printed with name and compilation date. The list may be extended with the information

*** warning: versionerror

- the program should be recompiled, but loading continues

*** overlap

- the program is already in the LINKER catalog. The program in the catalog is used instead and the loaded program deleted

*** skipped

- the loaded program is deleted because of problems during load. E.g. an external reference could not be defined

*** not loaded

- the program was not in the LINKER catalog or amongst the loaded programs.

2.2 Debug Mode

2.2.1 Activation

If the MODE switch is set in the range 0 to 7, the console may be put in Debug-mode (D-mode) at any time by pressing the BELL key (CTRL and G keys) without stopping instruction execution in the processor. A number of display and control commands become available for technical purposes.

2.2.2 Display Commands

Display commands cause the display of eight words of data. The following display commands are available:

M <addr> — Modify Memory
Displays the contents of the 8 memory locations starting at <addr>.

W <register> — Modify Working Registers
Displays the contents of the 8 working registers starting at <register>.

L <level> — Modify Working Registers
Displays the level number and the contents of the 8 working registers belonging to <level>.

Y <yaddr> Modify Control Microprocessor RAM

Displays the contents of the 8 control microprocessor RAM locations starting at <yaddr>.

Display commands are executed in the following situations:

1. When a display command is entered

One can now modify the displayed data by entering new data in the same positions on the following line. Pressing the space bar will move the cursor one position to the right.

A display command is terminated by pressing one of the following keys:

CR The CR key terminates the current display command. The console will await the next command.

\+ The + key terminates the current display command and executes a display command for the succeeding 8 words (M or Y) or the 8 registers on the succeeding level (W).

\- The - key terminates the current display command and executes a display command for the preceding 8 words (M or Y) or the 8 registers on the preceding level (W).

ESC The ESC key terminates the current display command, but no data modification takes place. The text ESC is displayed. The console will await the next command.

2. When a control command (sect.3.1.3) is terminated

The last executed display command is repeated, but modification of the displayed data is not allowed. The console will await the next command.

2.2.3 Control Commands

The following control commands are available:

R — Run
The processor will start instruction execution.

S — Instruction Step
The processor will execute one instruction, stop, and reactivate the console.

S <steps> — Multi-Instruction Step
The processor will execute <steps> instructions, stop and reactivate the console.

2.2.4 Command Parameters

All numbers entered or displayed are hexadecimal.

At any time the entering of an empty command (i.e. pressing the CR key) will cause the previous command to be repeated.

An address (<addr>) is entered using one of the following formats:

<base> : <disp>

or

: <disp>

<base> is the leftmost 16 bits of the 32-bit address.

<disp> is the displacement within the selected memory module, i.e. the rightmost 16 bits of the address.

If the second format (: <disp>) is used, the last entered address base will be echoed and used.

3. AUTOLOADING 3.

The autoload function may be initiated by:

- Power Restart
- Watchdog Restart

Power Restart

Power Restart happens:

- when power ON is performed manually on the OCP or on the power supply,
- by temporary power failure,
- by manual activation of the autoload button on the OCP or the AUTO push button on the power supply.

The built-in test programs are activated controlled by the ´MODE´ switch, the CPU initializes the registers, whereafter control is passed to the autoload program residing on the first memory module.

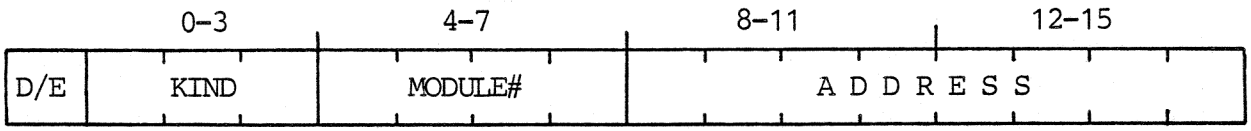
Watchdog Restart

Watchdog Restart may be activated both manually by means of the ´Y´ debug console command and from the software.

The CPU initializes the registers, whereafter control is given to the autoload program. No built-in test programs are activated.

3.1 Autoload Switch Format 3.1

The autoload program interprets the four BUS switches on the Processor Front Panel according to the format:



D/E — Autoload Disabled/Enabled
is intended for drivers controlling external devices, which may autoload RC3502. A driver may activate the watchdog function if autoload enabled.

0 ~ enabled
1 ~ disabled

KIND — Autoload Kind
defines which algorithm the autoload program executes.

0 JUMP
An unconditional jump is performed to the location defined by ´MODULE#´ and ´ADDRESS´. Only jumps with base in the range 00C0 to 00DE are possible.

1 IFP MB Master
Defines this RC3502 as the Multibus Master in a multi processor configuration. Autoload is from IFP whereafter the software in EPROMs is included.

2 FPA
Autoload is from FPA, and the software in EPROMs is included.

3 IFP MB Slave
Defines this RC3502 as a Multibus Slave in a multi processor configuration. Autoload is from IFP, and the software in EPROMs is included.

4 PTR
Autoload is from paper tape reader, and the software in EPROMs is included.

5 Autoload is from COM204 (Intelligent HDLC Controller), and the software in EPROMs is included.

6 EPROM
No autoload from external device. Only software in EPROMs is included.

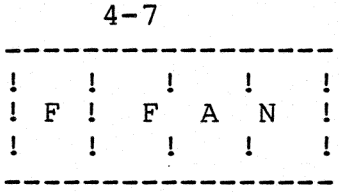
7 (Not used).

MODULE# Module Number
Defines the value of the module select field in an address.

KIND = JUMP (0)
Module is the module number in the jump address.

KIND = IFP (1 or 3)
Module is the module number in the start address of the IFP memory.

KIND = COM204 (5)
The field has another interpretation:



F - defines the type of load request transmitted on the HDLC line:
0 NNP loadrequest
1 X.25/3 loadrequest

FAN - defines the maximum relative channel no. used for autoload in a cyclic way if a channel fails. Channels 0,1,...FAN relative to the start channel (see later) are used.

ADDRESS specifies the displacement part of an address or interrupt level (input/output channel) depending on the value of KIND.

KIND	ADDRESS
0	Displacement
1	IFP interrupt level
2	FPA100 REC input/output channel
3	IFP interrupt level
4	PTR input/output channel
5	CHANNELx128+COM204 interrupt level
6	'Not used'
7	'Not used'

Example 0 GOTO 00C8.007A

```
--------          --------
! 047A !   or     ! 847A !
--------          --------
```

Example 1 Autoload from IFP which starts in address 0086.0000 and interrupts on level 112 (decimal)

```
--------          --------
! 1370 !   or     ! 9370 !
--------          --------
```

Example 2 Autoload from FPA in channel 81 (decimal)

```
--------          --------
! 2x51 !   or     ! Ax51 !
--------          --------
```

Example 3 (See example 1)

Example 4 Autoload from paper tape reader in channel 83 (decimal)

```
--------          --------
! 4x53 !   or     ! Cx53 !
--------          --------
```

Example 5 Autoload from COM204 address 0092.8000 level 19 (decimal) channel 1 using NNP loadrequest and a fan consisting of 3 channels:

```
--------          --------
! 5293 !    or    ! D293 !
--------          --------
```

Conventions:

1. Start address of the reference controller must be 0090.0000.

2. Interrupt level of the reference controller must be a multiplum of 8.

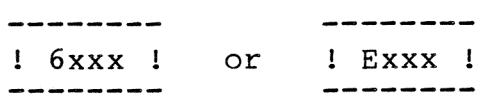
3. The controllers must be consecutive and increasing memorywise and according to interrupt levels (see the following table).

COM204 (Example 5 cont.)

	address	level	channel	
Reference Controller	0090.0000	16	0 1	
	0090.8000	17	0 1	
	0092.0000	18	0 1	
First Controller	0092.8000	19	0 1	<- 1)
Last Controller	0094.0000	20	0 1	<- 2)
	0094.8000	21	0 1	
	0096.0000	22	0 1	
	0096.8000	23	0 1	

1) first channel in fan
2) last channel in fan

Example 6 No autoload, only inclusion of software in EPROMs.



3.2 Autoload Messages

Autoload from ifp in xxxxH
Autoload from ifp in xxxxH
Autoload from ptr in xxxxH
Autoload from COM204 addr xxxx.xxxx level xxxxH channel xxxxH
Autoload from eprom
*** undefined switchkind

....

- a full stop is printed for every program loaded.

*** exception: xxxx at: xxxx.xxxx

- consult appendix I for interpretation of the exception code.

*** buy more memory!!!!

- more RAM memory must be installed to hold the autoloaded programs.

**** warning: versionerror at xxxx.xxxx

- the program identified by the address must be recompiled to be autoloaded. Consult the output from CROSSLINK for identification.

**** warning: <program> overlapping at xxxx.xxxx

- the program identified by the address has a double in the system. The program is included and operation continues.

```
expected: xxxx
received: xxxx
```

- the crc16 data check reports an error. If more than 5 errors occur, the autoload program will start all over again.

3.3 Generating Autoload Files

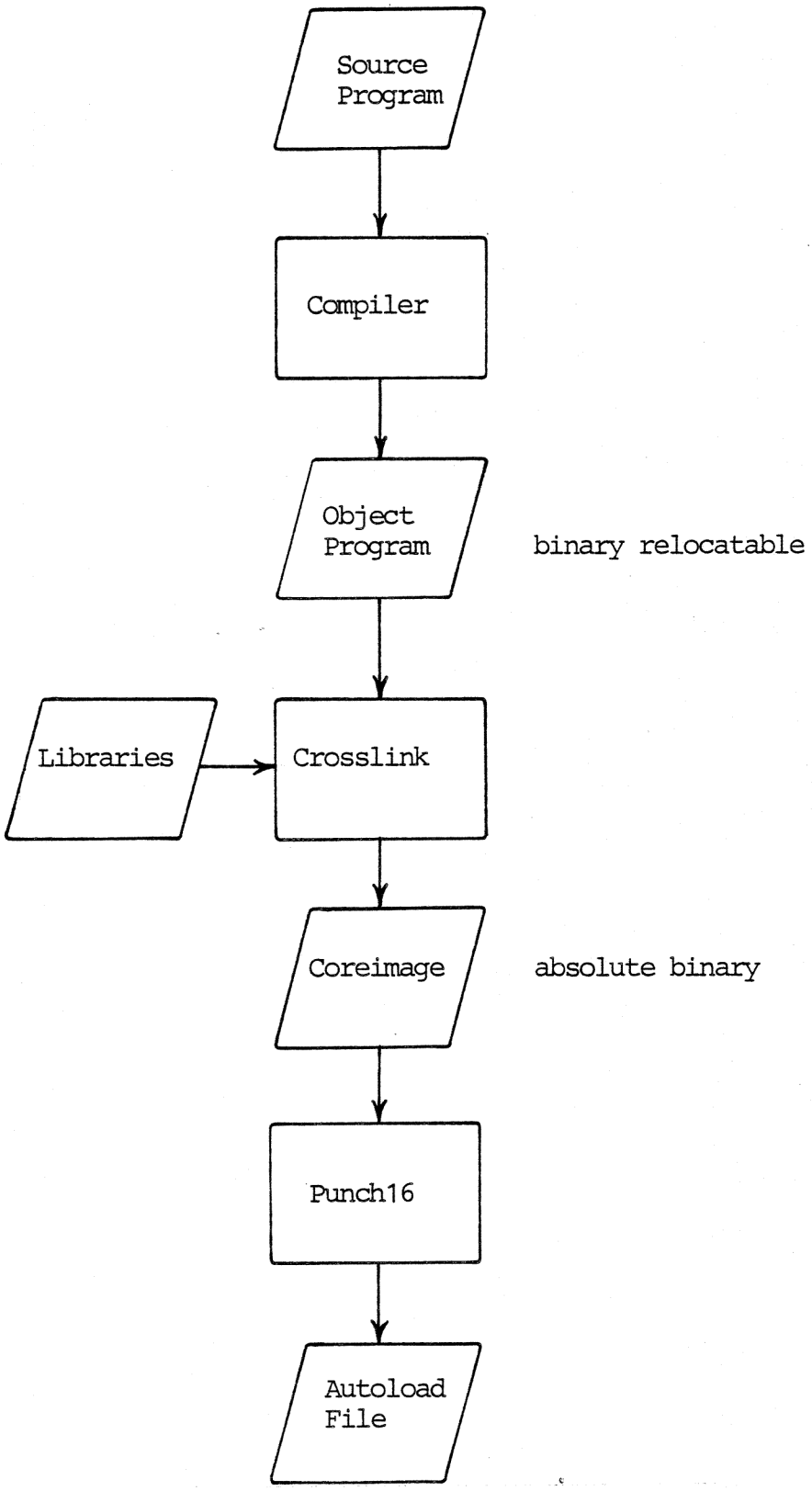


Fig. 3. Autoload File Generation.

3.3.1 Generating a Paper Tape Autoloadfile 3.3.1

If the file ´coreimage´ has been produced by the ´CROSSLINK´ program, the following call will generate a paper tape with the correct format for the autoloadprogram in RC3502:

```
tpn = punch16 mode.crc16 coreimage
```

3.3.2 Generating an FPA/IFP Autoloadfile 3.3.2

If the file ´coreimage´ has been produced by the ´CROSSLINK´ program, the following calls will generate an autoloadfile and autoload the RC3502 if connected via the process ´main35001´:

```
bootfile = punch16 mode.crc16 coreimage
main35001 = autoload bootfile start.no
```

3.3.3 Generating TES202 Eproms 3.3.3

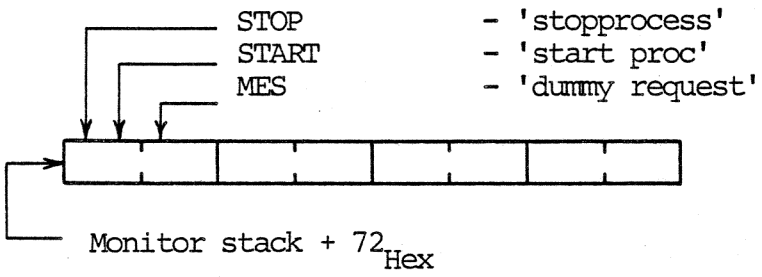
Consult ref. 10.

4. ERROR PROCEDURES

For use in error situations it might be useful to get testoutput from the autoload program ´BOOT´ or from the process MONITOR which starts and stops process incarnations.

4.1 Monitor Testoutput

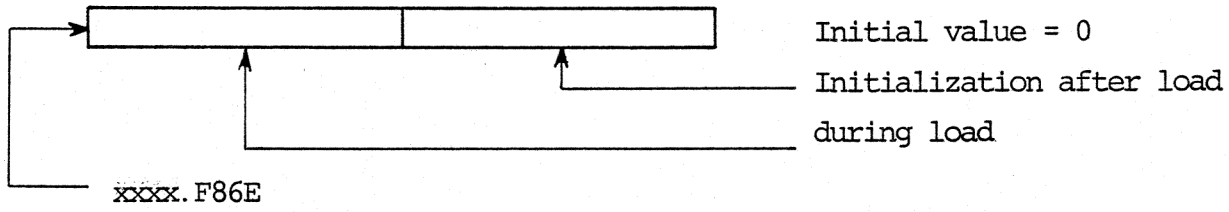
The monitor stack address is obtained by the LIST MONITOR command to OPSYS or from the testoutput from BOOT (see 4.2).



Initial value = 0.

4.2 Boot Testoutput

The BOOT stack address is xxxx.F800 where xxxx is the base address of the last RAM memory module in the RC3502. If testoutput during autoload from FPA is wanted, you must increase the software time out in the FPA xmt driver in RC8000 from 1 sec. to at least 5 sec.


Initial value = 0
Initialization after load
during load
xxxx.F86E

A. REFERENCES A.

1. RCSL No. 42-i1577
   PASCAL80, Introduction

2. RCSL No. 52-AA964
   PASCAL80, Report

3. RCSL No. 42-i1539
   PASCAL80, User´s Guide

4. RCSL No. 31-D617
   PASCAL80, Driver Conventions

5. RCSL No. 31-D627
   RC3502, Introduction

6. RCSL No. 52-AA972
   RC3502, Reference Manual

7. RCSL No. 52-AA1167
   RC3502-PASCAL80
   Reference Manual

8. RCSL No. 30-M300
   RC3502 PROM Blasting Program
   User´s Guide

B. OPSYS COMMANDS

B.

break <incarnation>

check <module no>

create <incarnation>{ as <process> }$_0^1$

date { <year>.<month>.<day> <hour>.<minute>.<seconds> }$_0^1$

excode <integer>

free

from { ptr / fpa }

help

in <inchannel>

kill <incarnation>

link <process>

list { <incarnation> }$_0^n$ <nl>

load { <program> }$_0^n$ <nl>

lookup { <program> }$_1^n$ <nl>

print { <base> { <firstdisp> { <lastdisp> { <no_of_words_per_line> { <delta> }$_0^1$ }$_0^1$ }$_0^1$ }$_0^1$ }$_0^1$ <nl>

priority <integer>

remove <incarnation>

run <incarnation> as <process>

size <integer>

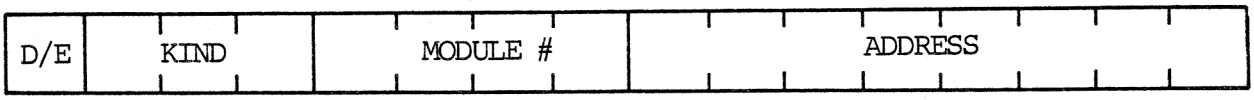
start <incarnation>

stop <incarnation>

unlink <process>

unload <program> <nl>

C. AUTOLOAD SWITCH LAYOUT



D/E - Autoload Disabled/Enabled

disabled = 1
enabled = 0

Intended for drivers controlling External Devices, which may autoload RC3502. The drivers may activate the watchdog function, if D/E = 0. BOOT ignores this switch.

KIND

0 JUMP
1 IFP MB master
2 FPA
3 IFP MB slave
4 PTR
5 COM204
6 EPROM

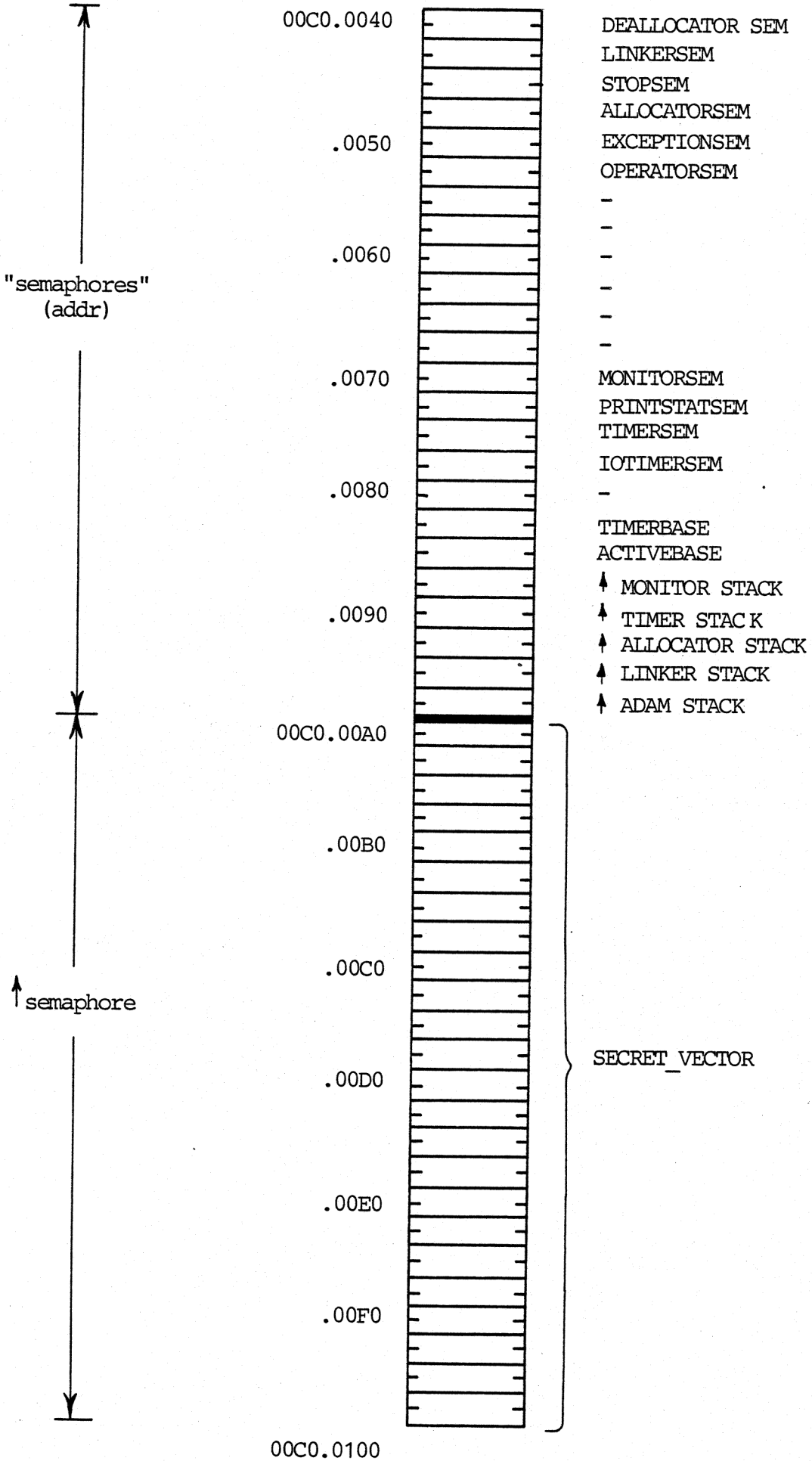
ADDRESS

Displacement	if JUMP kind
I/O channel	if FPA or PTR
Interrupt level	if IFP
Channel x 128 + interrupt level	if COM204

MODULE#

Module number	if JUMP kind
Multibus Module No.	if IFP
Load request x 8 + fan	if COM204

D. SECRET VECTOR LAYOUT D.

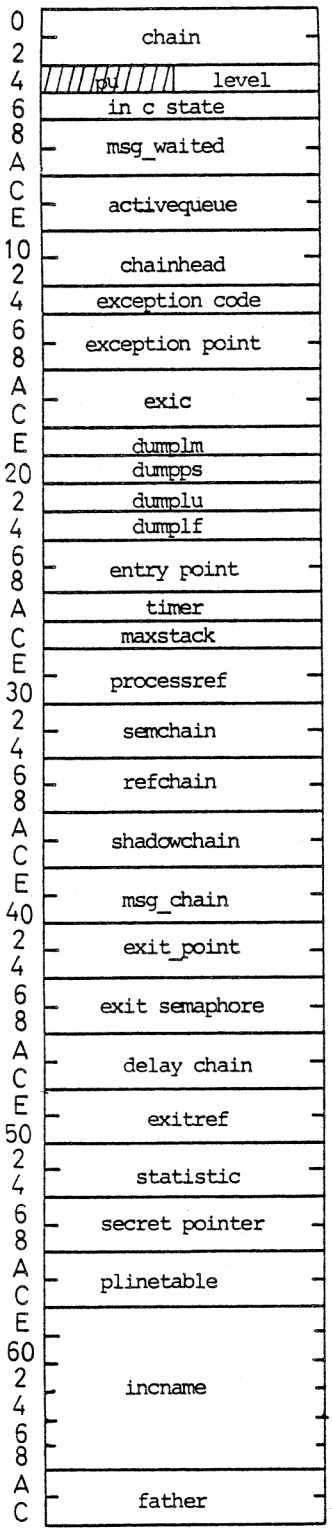


```
00: except
01: rec1
02: rec2
03: rec3
04: rec4
05: rec5
06: rec6
07: rec7
08: rec8
09: rec9
0a: rec10
0b: rec11
0c: rec12
0d: rec13
0e: rec14
0f: rec15
10: cwait
11: csign
12: crele  1
13: cllst
14: cskip
15: csens
16: cwtac
17: mwtac
18: mwis
19: mtime
1a: csell
1b: cstop
1c: cstdr
1d: sched
1e: cslev
1f: cgreg
20: mwt
21: iowc
22: iors
23: iorw
24: ioww
25: iogo
26: iogi
27: ionci
28: mwit
29: iocci
2a: iocda
2b: ioibx
2c: trape  1
2d: trapr  1
2e: mxept
2f: mnoop
30: mwst
31: ult
32: eq
33: ne
34: lt
35: gt
36: le
37: ge
38: mwist
39: tnill
3a: topen
3b: tlock
3c: teqad
3d: not
3e: except
3f: except
40: madd
41: msub
42: uadd
43: usub
44: add
45: sub
46: umul
47: udiv
48: umod
49: mul
4a: div
4b: mod
4c: and
4d: or
4e: xor
4f: crc16
50: neg
51: abs
52: compl
53: shc
54: shc8
55: except
56: setcr
57: settm
58: mcis
59: seteq
5a: setsb
5b: setsp
5c: setun
5d: setin
5e: setdi
5f: setad
60: rec0
61: jmzeq  2
62: jmzne  2
63: jmzlt  2
64: jmzgt  2
65: jmzle  2
66: jmzge  2
67: jmprw  2
68: mcit
69: jmphc  4
6a: jmppd
6b: jmcht  4
6c: intrs
6d: index
6e: inprs
6f: inpss
70: iorbbc
71: iorbb
72: iowbbc
73: iowbb
74: iorbwc
75: iorbw
76: iowbwc
77: iowbw
78: mcist
79: pcald
7a: pcals  1 4
7b: pexit  2
7c: lpush
7d: lpop
7e: lrese
7f: llock  2
80: svsb0
81: rvsb0
82: svsb2
83: rvsb2
84: svsb4
85: rvsb4
86: svsb6
87: rvsb6
88: mwi
89: revgbs 1
8a: stvlbs 1
8b: revlbs 1
8c: stnhb  2
8d: rvsb12
8e: renpb
8f: renhb  2
90: readb
91: crget
92: stvgb  2
93: revgb  2
94: stvib  1 2
95: revib  1 2
96: stvlb  2
97: revlb  2
98: stvsb  2
99: revsb  2
9a: stvab  4
9b: revab  4
9c: svsb28
9d: svsb29
9e: svsb30
9f: svsb31
a0: svsw0
a1: rvsw0
a2: svsw2
a3: rvsw2
a4: svsw4
a5: rvsw4
a6: svsw6
a7: rvsw6
a8: rechw  2
a9: revgws 1
aa: stvlws 1
ab: revlws 1
ac: moveb
ad: rvsw12
ae: moveg
af: revpw
b0: readw
b1: crput
b2: stvgw  2
b3: revgw  2
b4: stviw  1 2
b5: reviw  1 2
b6: stvlw  2
b7: revlw  2
b8: stvsw  2
b9: revsw  2
ba: stvaw  4
bb: revaw  4
bc: svsw28
bd: svsw29
be: svsw30
bf: svsw31
c0: svsf0  1
c1: rvsf0  1
c2: svsf2  1
c3: rvsf2  1
c4: svsf4  1
c5: rvsf4  1
c6: svsf6  1
c7: rvsf6  1
c8: rechws 1
c9: revgfs 1 1
ca: stvlfs 1 1
cb: revlfs 1 1
cc: revsm
cd: rvsf12 1
ce: reagds 1
cf: realds 1
d0: crram
d1: mbtes  2
d2: stvgf  2 1
d3: revgf  2 1
d4: stvif  1 2 1
d5: revif  1 2 1
d6: stvlf  2 1
d7: revlf  2 1
d8: stvsf  2 1
d9: revsf  2 1
da: stvaf  4 1
db: revaf  4 1
dc: svsf28 1
dd: svsf29 1
de: svsf30 1
df: svsf31 1
e0: reagd  2
e1: rvsd0
e2: reaid  1 2
e3: rvsd2
e4: reald  2
e5: rvsd4
e6: reasd  2
e7: rvsd6
e8: rechd  4
e9: revgds 1
ea: stvlds 1
eb: revlds 1
ec: setst
ed: rvsd12
ee: stcea
ef: revpd
f0: cwram
f1: mbset  2
f2: stvgd  2
f3: revgd  2
f4: stvid  1 2
f5: revid  1 2
f6: stvld  2
f7: revld  2
f8: stvsd  2
f9: revsd  2
fa: stvad  4
fb: revad  4
fc: reard  2
fd: reaxd
fe: cexch
ff: except
```

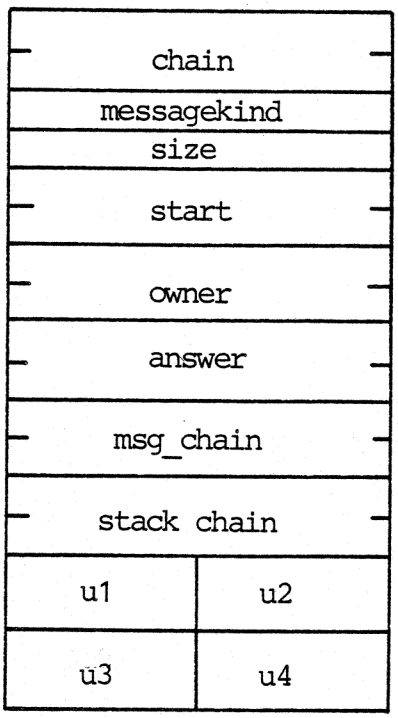
F. INSTRUCTION MNEMONICS

abs	51	iorbw	75	mwtac	17	revgbs	89	rvsw6	a7	svsb30	9e
add	44	iorbwc	74	mxept	2e	revgd	f3	sched	1d	svsb31	9f
and	4c	iors	22	ne	33	revgds	e9	setad	5f	svsb4	84
cexch	fe	iorw	23	neg	50	revgf	d3	setcr	56	svsb6	86
coreg	1f	iowbb	73	notinstr	3d	revgfs	c9	setdi	5e	svsf0	c0
cllst	13	iowbbc	72	or	4d	revgw	b3	seteq	59	svsf2	c2
compl	52	iowbw	77	pcald	79	revgws	a9	setin	5d	svsf28	dc
crc16	4f	iowbwc	76	pcals	7a	revib	95	setsb	5a	svsf29	dd
crele	12	iowc	21	pexit	7b	revid	f5	setsp	5b	svsf30	de
crget	91	ioww	24	readb	90	revif	d5	setst	ec	svsf31	df
crput	b1	jmcnt	6b	readw	[illegible]	reviw	b5	settm	57	svsf4	c4
crram	d0	jmphc	69	reagd	e0	revlb	97	setun	5c	svsf6	c6
csell	1a	jmppd	6a	reagds	ce	revlbs	8b	shc	53	svsw0	a0
csens	15	jmprw	67	reaid	e2	revld	f7	shc8	54	svsw2	a2
csign	11	jmzeq	61	reald	e4	revlds	eb	stcea	ee	svsw28	bc
cskip	14	jmzge	60	realds	cf	revlf	d7	stnhb	8c	svsw29	bd
cslev	1e	jmzgt	64	reard	fc	revlfs	cb	stvab	9a	svsw30	be
cstdr	1c	jmzle	65	reasd	e6	revlw	b7	stvad	fa	svsw31	bf
cstop	1b	jmzlt	63	reaxd	fd	revlws	ab	stvaf	da	svsw4	a4
cwait	10	jmzne	62	rec0	60	revpd	ef	stvaw	ba	svsw6	a6
cwram	f0	le	36	rec1	01	revpw	af	stvgb	92	teqad	3c
cwtac	16	llock	7f	rec10	0a	revsb	99	stvgd	f2	tlock	3b
div	4a	lpop	7d	rec11	0b	revsd	f9	stvgf	d2	tnill	39
eq	32	lpush	7c	rec12	0c	revsf	d9	stvgw	b2	topen	3a
exception	00	lrese	7e	rec13	0d	revsm	cc	stvib	94	uadd	42
exception	2c	lt	34	rec14	0e	revsw	b9	stvid	f4	udiv	47
exception	2d	mada	40	rec15	0f	rvsb0	81	stvif	d4	ult	31
exception	3e	mbset	11	rec2	02	rvsb12	8d	stviw	b4	umod	48
exception	3f	mbtes	d1	rec3	03	rvsb2	83	stvlb	96	umul	46
exception	55	mcis	58	rec4	04	rvsb4	85	stvlbs	8a	usub	43
exception	ff	mcist	78	rec5	05	rvsb6	87	stvld	f6	xor	4e
ge	37	mcit	68	rec6	06	rvsd0	e1	stvlds	ea		
gt	35	mnoop	2f	rec7	07	rvsd12	ed	stvlf	d6		
index	6d	mod	4b	rec8	08	rvsd2	e3	stvlfs	ca		
inprs	6e	moveb	ac	rec9	09	rvsd4	e5	stvlw	b6		
inpss	6f	moveg	ae	rechd	e8	rvsd6	e7	stvlws	aa		
intrs	6c	msub	41	rechw	a8	rvsf0	c1	stvsb	98		
iocci	29	mtime	19	rechws	c8	rvsf12	cd	stvsd	f8		
iocda	2a	mul	49	renhb	81	rvsf2	c3	stvsf	d8		
iogi	26	mwi	88	renpb	8e	rvsf4	c5	stvsw	b8		
iogo	25	mwis	18	revab	9b	rvsf6	c7	sub	45		
ioibx	2b	mwist	3b	revad	fb	rvsw0	a1	svsb0	80		
ionci	27	mwit	28	revaf	db	rvsw12	ad	svsb2	82		
iorbb	71	mwst	30	revaw	bb	rvsw2	a3	svsb28	9c		
iorbhc	70	mwt	20	revgb	93	rvsw4	a5	svsb29	9d		

G. PROCESS INCARNATION DESCRIPTOR LAYOUT G.



H. MESSAGE HEADER LAYOUT H.



I. EXCEPTION CODES

Code (hexadecimal)	Meaning
1	signal: reference = nil
2	renpb odd number of bytes
3	illegal field (last < first)
4	field overflow
5	reference = nil
6	not channel message
7	block i/o at level 0
8	not data message
9	size too small
A	last < first
B	arithmetic overflow
C	index out of bounds
D	undefined opcode
E	odd length in sets
F	setad truncation error
10	stack overflow
11	subrange out of bounds:
12	pointer = nil
13	push: first param = nil
14	push: first param not empty
15	push: identical arguments
16	push: second param locked
17	pop: first param <> nil
18	pop: second param = nil
19	pop: second param locked
1a	wait: reference <> nil
1b	reference locked
1c	lock: not data message
1d	lock: size error
1e	multiple wait on locked semaphore
1f	pool: no core
20	break: shadow = nil
21	remove: shadow = nil
22	start: shadow = nil
23	stop: shadow = nil
24	illegal switch in case construction
25	upper limit in call of succ
26	lower limit in call of pred
27, 28	system error
29	local reference variable not nil at routine exit
2A-2E	system error

"system error" indicates hardware error or microprogram error.

J. WORKING REGISTER LAYOUT J.

Description of the working registers:
(for a more detailed description, refer to the reference manual)

usage of the w-registers:

000-007	regset 0
008-00f	regset 1
-	-
-	-
-	-
3d8-3df	regset 123
3e0-3e7	monitor regset
3e8-3ef	com8085
3f0-3f7	work regset
3f8-3ff	dummy regset

regset 0 thru regset 123:

normal mode

1st reg.	lm	
2nd reg.	ps (bit 7-15):	psc, pst, pss, psi, ri, 0, to, eoi, supp arith ovf
3rd reg.	pb	
4th reg.	lu	
5th reg.	sf	
6th reg.	pr	
7th reg.	ib	
8th reg.	ic	

dummy mode

7th reg.	ffff
the other registers are undefined	

block i/o mode

1st reg.	top = last+1
2nd reg.	ps (as above)
3rd reg.	pb
4th reg.	next address.disp - 1
5th reg.	next address.base
6th reg.	pr
7th reg.	83xx, where xx = instruction code (which is <= 7f)
8th reg.	count

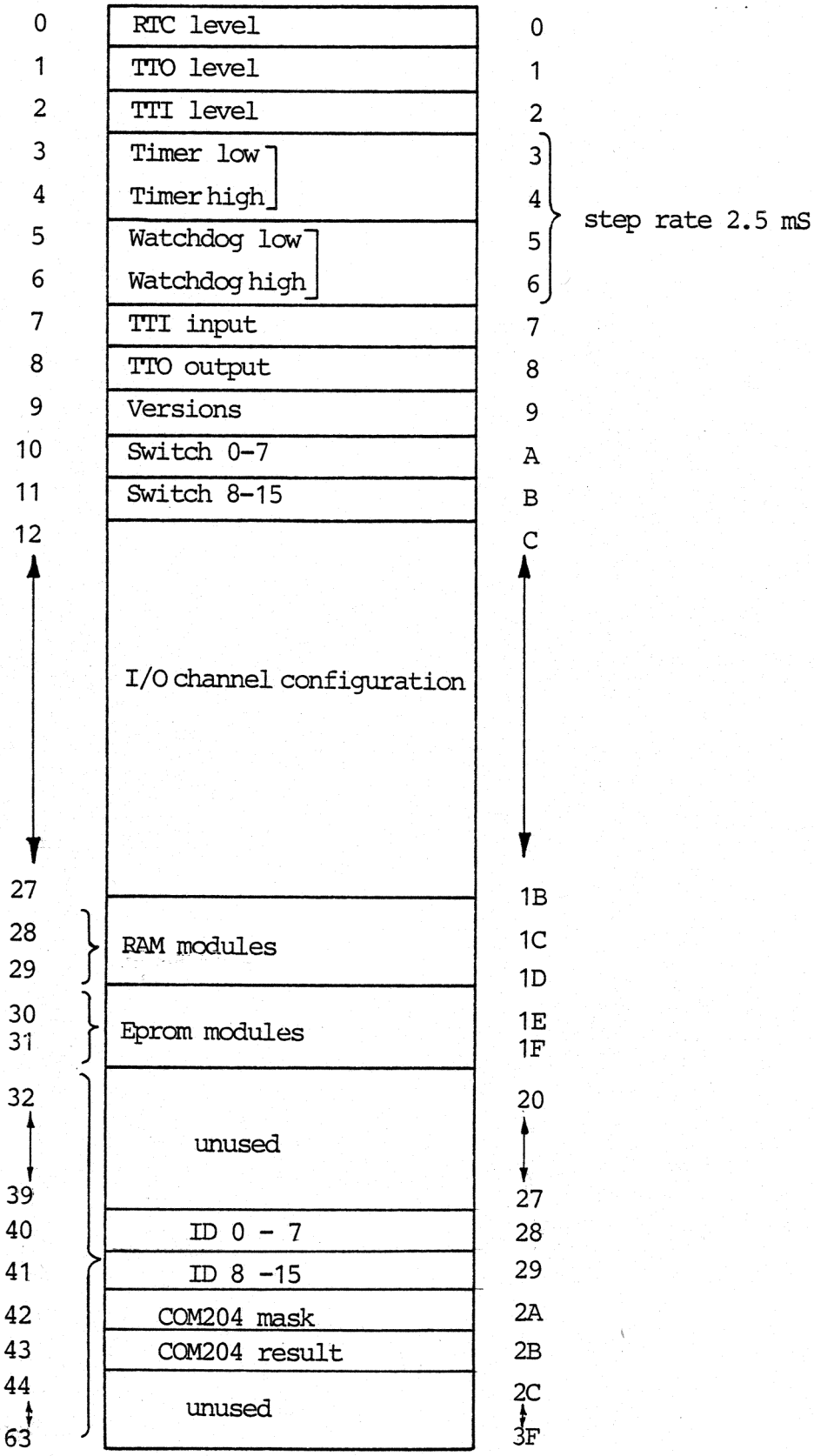
```
monitor regset:
     1st reg.    ↑actq(0).base
     2nd reg.    ↑actq(0).disp
     3rd reg.    n
     4th reg.    m
     5th reg.    k
     6th reg.    nxt
     7th and
     8th reg.    dummy loop counter

com8085:
     1st reg.    fifo5, fifo6
     2nd reg.    fifo3, fifo4
     3rd reg.    fifo1, fifo2
     4th reg.    cow (value, disp)
     5th reg.    unused
     6th reg.    message.errorcode
     7th reg.    message.base
     8th reg.    message.disp

work regset:
     used as working registers

dummy regset:
     1st reg.    breakpoint address.base
     2nd reg.    breakpoint address.disp
     3rd reg.    puerrmsk + stat base
     4th reg.    breakpointmode
                 (8000 means breakpoint active)
     5th reg.    dummy interrupt.lastlev
     6th reg.    dummy interrupt.count
     7th reg.    parity error address.base
     8th reg.    parity error address.disp
```

K. CONTROL MICROPROCESSOR RAM LAYOUT K.



L. INSTALLATION STANDARDS AND RECOMMENDATIONS

This is a recommendation concerning installation of hardware modules in the RC3502. The guidelines concern interrupt levels, input/output priorities, module number selections, DMA priority.

L.1 Input/Output Modules

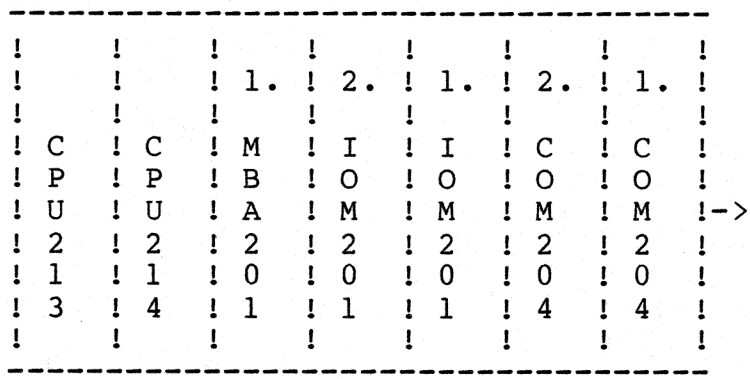
The priorities and interrupt levels in the following table should be followed according to input/output channels and priority. The first module in a group should be the lowest channel number.

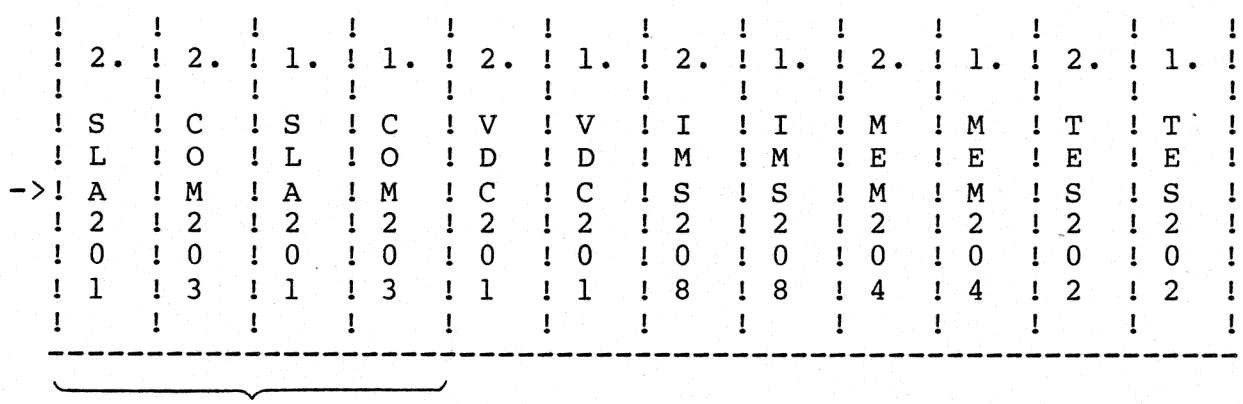
L.2 Memory Modules

MEM204 occupies two module addresses: one RAM module address in the INTERNAL memory section modules 0-15 and one PROM address in the INTERNAL memory section modules 16-31.

TES201/TES202 should be installed in the INTERNAL memory section modules 16-31. To avoid conflict between MEM204 and TES201/TES202 concerning the PROM module numbers 16-31, the first MEM204 module should be installed with the lowest possible module number (0) and the following in increasing, consecutive order. TES201/TES202 groups should be installed with the last module with the highest module number (31), and the other modules in decreasing, consecutive module numbers (30,29,28,...).

To obtain that input/output priorities and DMA priorities follow interrupt levels, the modules should be installed as follows:





The following three points should be observed:

1. The interrupt level priority chain starts at the CPU and must not be broken by empty positions until the last module, which uses interrupt level priority.

2. The DMA priority chain starts at the CPU and must not be broken by empty positions until the last module, which uses DMA priority.

3. The module, which is closest to the CPU, has highest priority, both according to interrupt level and DMA.

Module name	Interrupt level (Hex)	Dual Port Memory Moduleno. Base (Hex)	Dual Port Memory Moduleno. Moduleno. (Decimal)	No. of interrupt levels (Decimal)	No. x salesnumber
	Lowest priority				
CPU	00-04	-	-	5	
SPARE	05-07	-	-	3	
IMS208	08-0F	80-8E	EXT 0- 7	8	8 x RC3542
VDC201	10-2F	-	-	32	16 x RC3522
COM203	30-4F	-	-	32	4 x RC3543
COM204	48-4F	90-96	EXT 8-11	8	8 x RC3546
IOM201	50-6F	-	-	32	4 x RC3521
MBA201	70-77	98-9E	EXT 12-15	8	1 x RC3555
SPARE	78-7B	-	-	4	
	Highest priority				

M. CATCHWORD INDEX M.

address..... 25
as..... 10, 16
autoload..... 23
autoloadfile..... 31

break..... 9

check..... 10
COM204..... 25, 26
create..... 10

D-mode..... 8, 20
date..... 10
Debug-mode..... 8

EPROM..... 25
excode..... 10

f..... 25
fan..... 25
FPA..... 11, 24, 26
free..... 10
from..... 11

help..... 11

IFP..... 24, 26
in..... 11
indicators..... 2, 4

jack..... 2
jump..... 24

kill..... 11
kind..... 24

L..... 20
link..... 11
list..... 11
load..... 12
lookup..... 13

M..... 20
module..... 25

power restart..... 23
print..... 14
priority..... 16
PTR..... 11, 25

R..... 22
remove..... 16
run..... 16

S..... 22
size..... 16
start..... 16
stop..... 16
switches..... 2, 3

T-mode..... 8
Terminal-mode..... 8
testoutput..... 32

unlink..... 17
unload..... 17

W..... 20
watchdog restart..... 23

Y..... 21

RETURN LETTER

Title: RC3502 Operating Guide RCSL No.: 52-AA1156

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

. . . . . Fold here . . . . .

. . . . . Do not tear - Fold here and staple . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark