



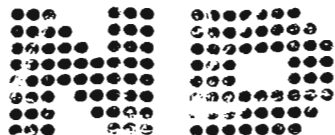
ND-5000 HARDWARE MAINTENANCE

ND-05.017.01 EN



ND-5000 HARDWARE MAINTENANCE

ND-05.017.01 EN



Norsk Data A.S

M A N U A L

ND-5000 HARDWARE MAINTENANCE

ND-05.017.01 EN

Written: : 25.03.1988

The manual

This manual covers the ND-5000 computer systems. The ND-110 part of the system is covered by the ND-100 Hardware Maintenance Manual. The intension of the manual is to be a helping hand for the service staff, thus used in addition to:

- The Service Handbook.
- ND-100 Maintenance Manual.

The reader

This manual has been prepared for the ND field service engineers and technical personell directly involved in maintaining the ND-5000 computer systems.

Prerequisite knowledge

A basic knowledge of the hardware in the ND-5000 computer systems.

Related manuals

<u>ND Number</u>	<u>Title</u>
ND-30.008	ND-100 Hardware Maintenance Manual
ND-30.005	Test Program Description for ND-100
ND-05.020	ND-5000 Hardware Description
ND-30.073	ND-5000 Macrotest Description

Table of contents

1	ND-5000 OVERVIEW.	1
1.1	The ND-5000 Family	1
1.1.1	The ND-5000 Model-range	3
	The ND-5000 Compact Systems	3
	The ND-5000 Large Systems	4
1.2	The Cabinets	5
1.2.1	The Large Cabinet	5
	The Card Rack	6
1.2.2	The Small Cabinet	8
	The Card Rack	9
1.3	The Power Supply	11
	The Small Cabinet	11
	The Large Cabinet	13
	Power Fail/Restart	22
1.4	The CPU-types	25
	ND-5000 basic CPU type 1	25
	ND-5000 Basic CPU type 2	26
	ND-5000 basic CPU type 3	27
	The difference between the CPU-types	28
	Microprograms versions	29
2	HARDWARE UPGRADING OF ND-5000 SYSTEMS	30
2.1	System description	30
2.1.1	Upgrading from a ND-5200 to a ND-5400 system	36
2.1.2	Upgrading from a ND-5400 to a ND-5500 system	36
2.1.3	Upgrading from a ND-5500 to a ND-5700 system	36
2.1.4	Upgrading from a ND-5700 to a ND-5800 system	36
2.1.5	Upgrading from a ND-5800 to a ND-5900/2/3/4 system	37
2.1.6	Setting of the ND-5000 CPU model	37
	Updating tool to be used on ND-5000 Compact systems	37
	Updating tool to be used on ND-5000 large cabinet version	38
	EXAMPLE	38
3	OCTOBUS COMMUNICATION	40
3.1	Introduction to Octobus hardware	40
3.2	Octobus frame format.	42
3.3	Introduction to the protocol	43
3.3.1	The message format	44
3.4	Octobus hardware on ND5000 CPU.	45
3.5	Internal octobus cable.	46
3.6	The ACCP	48
3.6.1	Connecting the ACCP console	48
3.6.2	ND-5000 selftest	52
	The Selftest status	53
	The Selftest Run	54

4	THE HARDWARE TRACE MODULE	56
4.1	The Trace module memory	56
4.2	Software control of the Trace module	56
	INITIATE-TRACER	57
	ARM-TRACER	58
	DISARM-TRACER	58
	DUMP-TRACE-MEMORY	58
	WRITE-TRACE-FILE	58
	READ-TRACE-FILE	58
	EXAMINE-TRACE	59
4.3	Dump of trace memory when an error situation has occurred	61
4.4	ND-5000 Trace module decoding tools	62
5	MAINTENANCE	64
5.1	Preventive maintenance	64
5.2	Fans	64
5.3	Voltages	65
5.4	LEDs	66
5.5	The ECO-system	66
5.6	Substituting modules	70
5.7	The CPU transport box	71
6	TROUBLESHOOTING	72
6.1	ND-100 Error	72
6.2	Errors during startup	73
6.2.1	Loading of control store	73
6.2.2	Different error messages that can occur during start-up of ND-5000	76
	Attempt to start the ND-500-monitor don't succeed	76
	ND-5000 Selftest is failing	77
	Errors during loading of control-store	80
6.2.3	ND-500 Error Messages	83
	General	83
	Fatal System Error Messages	84
	Hardware Fault	95
	Page Fault	102
	Protect Violation	108
	Index Scaling Error	118
	Illegal Instruction Code	123
	Instruction Sequence Error	127
	Illegal Operand Specifier	130
	Trap Handler Missing	134
7	SOME USEFUL DEBUGGING COMMANDS IN THE ND-5000 MONITOR	138
7.1	Debugging commands	138
7.2	LOOK-AT-PROGRAM	138
7.3	LOOK-AT-DATA	139
7.4	LOOK-AT-FILE	139
7.5	LOOK-AT-STACK	139
7.6	LOOK-AT-RELATIVE	139

7.7	LOOK-AT-REGISTER	139
7.8	LOOK-AT-SRF	140
7.9	LOOK-AT-RESIDENT-MEMORY	140
7.10	LIST-ACTIVE-SEGMENT	141
7.11	Set breakpoints	141
7.12	Address trace	142
7.13	Reset all debuggings activity	143
7.14	Trap handling	143
7.15	Program execution control	143
7.16	Display error messages from monitor calls	143
7.17	Resident place	144
7.18	List eco level and version of basic ND-5000 hardware/software	144
7.19	List memory configuration	145
7.20	Forced stop on the ND-5000	145
7.21	Dump of the Swapper datasegments	146
7.22	Dump all hardware registers	146
7.23	Operations against the control store	147
7.24	Reset ND-5000 cpu	147
7.25	Set cache mode	149
7.26	Attach another process	149

8 TEST AND UTILITY PROGRAMS --- 150

8.1	ND-5000 TEST MICRO PROGRAMS	150
8.1.1	Definitions	150
8.1.2	Switches on the Cache	150
8.1.3	Loading and starting SEMICS	150
	Starting the TPE monitor	151
	Starting SEMICS	151
8.1.4	Using SEMICS	152
	Prepare for testing	152
	Test numbering	152
	The initial parameter values	152
	Accessing files through SEMICS	153
	Running microtests	153
	Error in memory configuration	154
	Time Consumption	155
8.1.5	Logging errors	155
8.2	ND-5000 TEST MACRO PROGRAMS	156
8.2.1	TEST PROGRAMS	156
8.2.1.1	DESMODUR	156
	How to get Started	158
	Messages that Occur During a Testrun	163
	Logging errors	163
8.2.1.2	FLOTILJE	164
	General	164
	Options of testrun made before starting	165
	Options if error is discovered	166
	Options for environment	166
	Running FLOTILJE	167
	Logging errors	168
8.2.1.3	PIPELINE-EXERCISER	169
	General	169
	How to run the test	170
	Logging errors	171
8.2.1.4	MMS-TEST	172
	Running the MMS Test	173

	Logging errors	175
8.2.1.5	PAGE FAULT EXERCISER	176
	First the swap-file selection	176
	Prepare for testing.	177
	Precautions.	177
	A session example	178
	Error reporting	180
	The error message	180
	Example ONE:	181
	Example TWO:	182
8.2.1.6	FLOATING POINT TEST	183
	General	183
	Getting started	184
	Running FLOAT-TEST	185
	Logging errors	185
	Reading the error log file	186
	Miscellaneous	186
8.2.1.7	OCTOBUS	187
	Octobus test commands	187
	SET-PARAMETERS	187
	LIST-HARDWARE-CONFIGURATION	188
	RUN	188
	Logging errors	189
8.2.2	VERIFICATION PROGRAMS	190
8.2.2.1	SUPER	190
	Test Description	190
	Running SUPER	190
	Logging errors	193
8.2.2.2	SIBAS Test	194
	Logging errors	195
8.2.2.3	CXTEST	197
	Logging errors	197
8.2.2.4	INVERSE-MATRIX, WHETSTONE, DHRYSTONE and LACOURT Tests	198
	Logging errors	199
8.2.2.5	LIBTEST	200
	Logging errors	201
8.3	MULTIJOB TELEFIX	202
8.4	UTILITY PROGRAMS	203
8.4.1	N500X-MESSAGE	203
	Program description	203
	Running N500X-MESSAGE	204
8.4.2	TEST FUNCTIONS IN ND-5000/MF FIRMWARE	205
8.5	MF Bus Test and Maintenance Program	205
8.5.1	Connecting the console terminal to the controller	205
8.5.2	Description of the most useful commands	207
	INITATE-EEPROM:	208
	CONFIGURATE-SLOT:	208
	LIST-CONFIGURATION:	208
	TEST-MEMORY:	209
	SYNDROME-TEST:	209
	AUTOINITIATE-BANK:	210
	LIST-OCTOBUS-STATION:	210
	OCTOBUS-SELFTEST	210
	ACCESS-OCT-REG	210
	OCT-CONTROL-FUNCTION	211
	OCT-TRANSMIT-STATUS	211
	OCTOBUS-DRIVER	211

	LIST-SUBPROC-TABLE	212
	READ-OCTOBUS-RECEIVE	212
	TRANSMIT-OCTOBUS	212
8.5.3	Configuring the MF-system	215

9 SWITCHES AND INDICATORS --- 224

The Mother board (5502)	224
The MPM Line Driver	225
The Cache Module (5610)	227
The MFB Bus Port	228
The Dynamic RAM (5411)	229
The MFB Controller (5464)	230
The Double Bus Controller	231

(x)

APPENDIX A: _____ 232

A.1	Octobus messages from/to the ND5000 microprogram	232
A.2	Contents of the different messages sent as an Octobus message .	232
A.3	Commands received by the ACCP	234
A.4	The Context block(Register block)	236
A.5	Allocation of registers in the Scratch Register File	238
A.6	ND-100/ND-5000 Communication	241
A.7	The Extended CPU datafield.	242
A.8	The message buffer.	247
A.9	The Message block	248
A.10	Trap message	252

List of Figures

1.	The ND-5000 Computer System	2
2.	The ND-5000 Cabinet	5
3.	The ND-5000 Compact Cabinet	8
4.	Power Supply in the ND-5000 Compact Cabinet	12
5.	Power Supply in the ND-5000 Large Cabinet	13
6.	The Switches and the LEDs on the Power Supply Modules	15
7.	Power Fail Interrupt Connection on the Large Cabinet (new type)	22
8.	Power Fail Interrupt Connection on the Large Cabinet (old type)	23
9.	Layout of CPU type 1	25
10.	Layout CPU type 2	26
11.	Layout CPU type 3	27
12.	Octobus Cabling On ND-5000	47
13.	ACCP-Console Connection on the Large Cabinet	49
14.	ACCP-console Connection on ND-5000 Compact Cabine	50
15.	The fans	65
16.	The CPU transport box	71
17.	ND-100 Error Conditions	72
18.	Connecting the MF console on the large cabinet	205
19.	Connecting the MF console on the Compact cabinet	206

List of Tables

1.	ND-5000 Compact Configuration Overvie	3
2.	ND-5000 Compact Configuration Overvie	4
3.	The MF-bus Card Rack — old type	6
4.	The ND-5000 Card Rack — new type	7
5.	The ND-5000 Compact Card Rack	9
6.	The ND-5000 Compact Card Rack — new type	10
7.	Specifications for the ND-5000 Power Supply	17
8.	The difference between the CPU-models	28
9.	Survey of ND-5000 error messages	83

CHAPTER 1 ND-5000 OVERVIEW.

1.1 THE ND-5000 FAMILY

The ND-5000 is a 32-bit general purpose super-mini computer, and it makes the high end in the range of computer systems from Norsk Data.

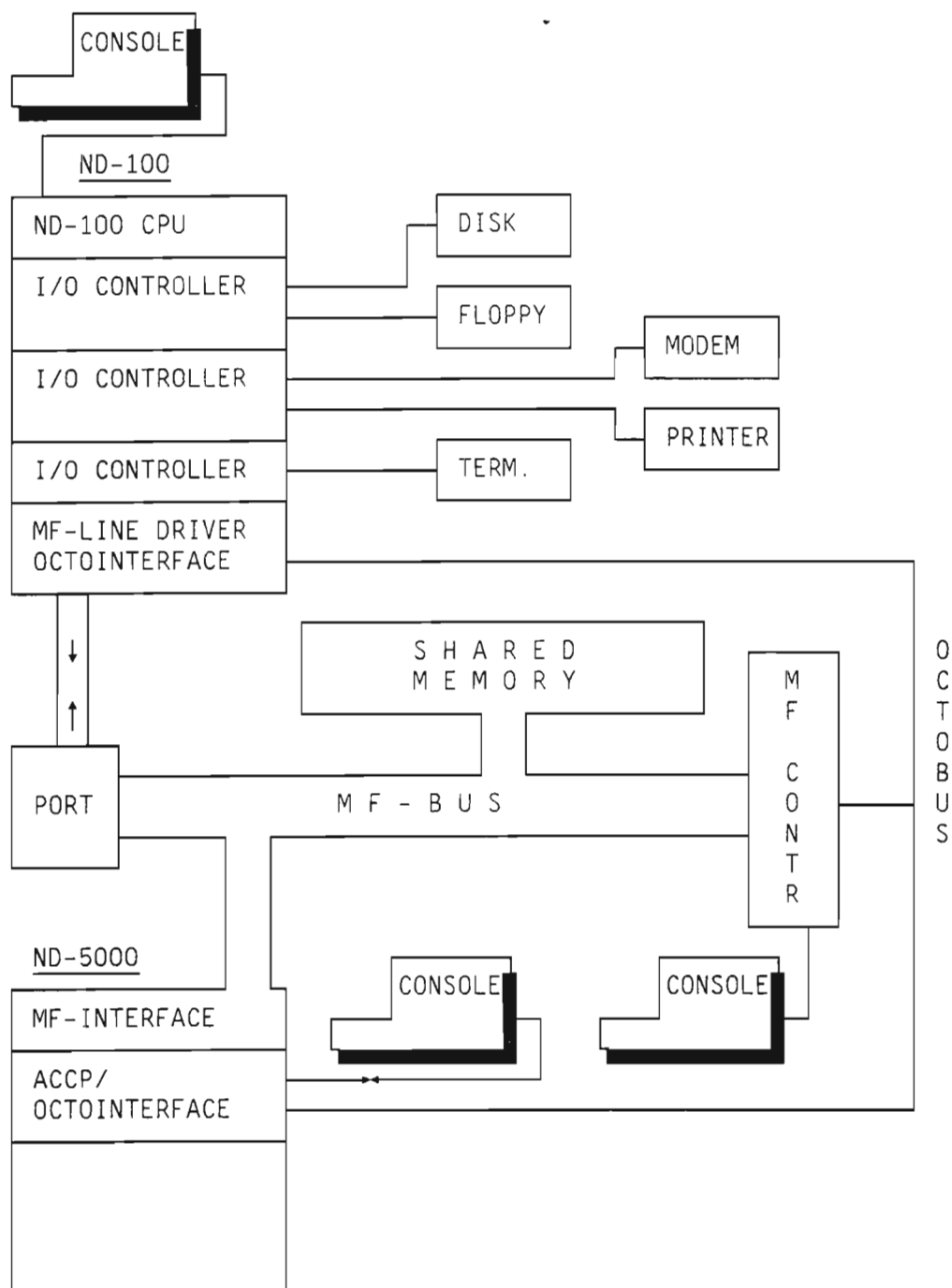
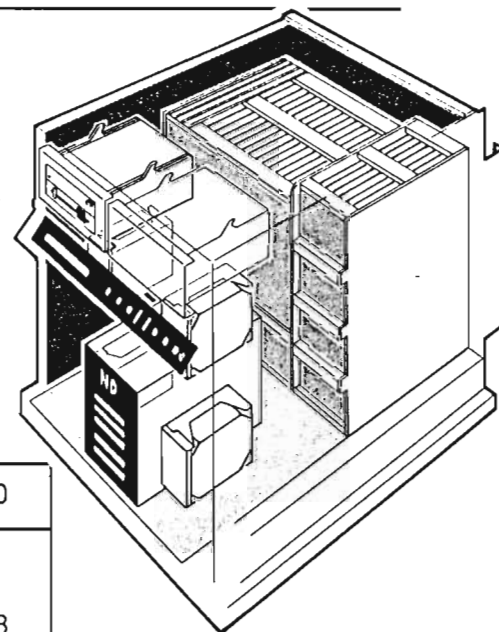


Figure 1. The ND-5000 Computer System

1.1.1 THE ND-5000 MODEL-RANGE

The ND-5000 systems are delivered in the Compact cabinet and in large cabinets.

THE ND-5000 COMPACT SYSTEMS



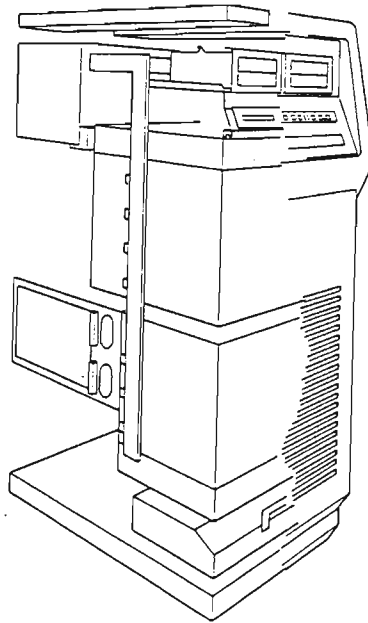
Parameter	5200	5400	5500	5700
CPU type	1	2	2	2
CPU model	2	4	5	7
CPU ND-number	110249	110248	110247	110218
Mic. program vers. *	110<xx>	111<xx>	112<xx>	113<xx>
I/O processor	ND110	ND120	ND120	ND120/CX w/2Mb
Disk size (A-models) (Internal)	60 to 4*125 Mb	125 to 4*125 Mb	125 to 4*125 Mb	125 to 4*125 Mb
Disk size (Model B) (External)	Up to 3.6 Gb	Up to 3.6 Gb	Up to 3.6 Gb	Up to 3.6 Gb
Streamer	A-models	A-models	A-models	A-models

System type	ND-5200 Compact	ND-5400 Compact	ND-5500 Compact	ND-5700 Compact
DETAILED DISK CONFIGURATION				
Model:				
A0/A10	60 MB	-	-	-
A1/A11	125 MB	125 MB	125 MB	125 MB
A2/A12	2X125 MB	2X125 MB	2X125 MB	2X125 MB
A3/A13	3X125 MB	3X125 MB	3X125 MB	3X125 MB
A4/A14	4X125 MB	4X125 MB	4X125 MB	4X125 MB
B	External	External	External	External

Table 1. ND-5000 Compact Configuration Overview

*= Valid for SINTRAN K, WM 406.

THE ND-5000 LARGE SYSTEMS



Parameter	5200	5400	5500	5700	5800	5900*
CPU type	1	2	2	2	3	3
CPU model	2	4	5	7	8	8
CPU ND-number	110249	110248	110247	110218	110171	110171
Mic. program vers. **	110<xx>	111<xx>	112<xx>	113<xx>	114<xx>	114<xx>
I/O processor	ND110	ND120	ND120	ND120/CX w/2Mb	ND120/CX w/4Mb	ND120/CX w/4Mb
Memory size shared/local	4/2	4/4	8/4	12/6	16/10	24/6
Cache size						
Data (KB)	-		64	64	64	2 X 64
Instruction	-	320	320	320	320	2 X 320
Disk size (External)	Up to 29 Gb					

Table 2. ND-5000 Compact Configuration Overview

* = Model 5900 contain 2, 3 or 4 CPUs.

**= Valid for SINTRAN K, WM 406.

1.2 THE CABINETS

The ND-5000 is delivered either in a small or a large cabinet. This section gives a short description of the two cabinets, the card crates etc.

1.2.1 THE LARGE CABINET

The computer cabinet looks like this:

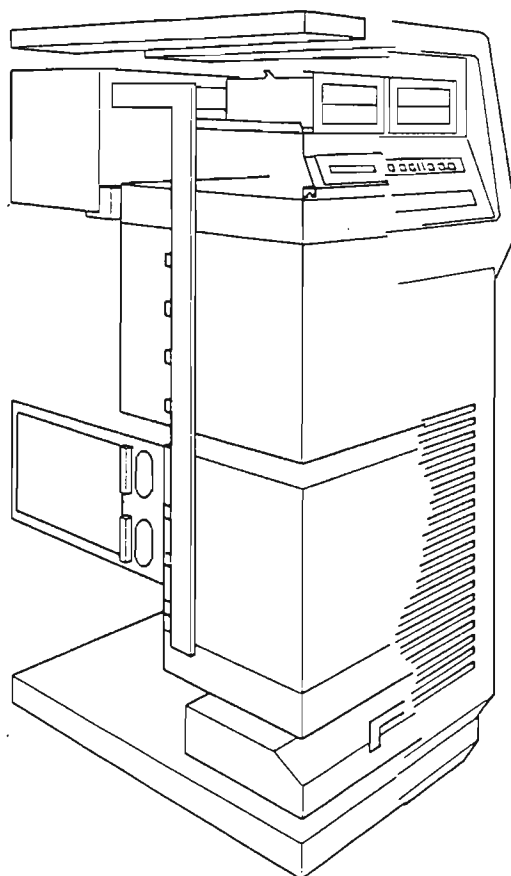


Figure 2. The ND-5000 Cabinet

THE CARD RACK

Card Position	Card Type	Comments
1		Earlier used for ND-570 floating point unit cards.
2		
3		
4		
5		
6	ND-5000 CPU	
7		
8	(MFB Dynamic RAM)	
9	(MFB Dynamic RAM)	CPU type 3: pos 6,7,8 and 9.
10	MFB Dynamic RAM	Positions 9 to 24 are for memory and ports.
11	MFB Dynamic RAM	
12	Ports or memory by choice	
13		
14		
15		Note that the Multi Function Bus Controller only has 15 external request and grant lines. Masters can then not be placed in pos. 8-11. These can only be used for memory.
16		
17		
18		
19		
20		The Multi Function Bus Ports are the same cards as the MPM-5 ports.
21		
22		
23		
24		
25	Ports or memory by choice	The MFB Dynamic RAM cards are the same as the MPM-5 dynamic RAM cards.
26	Multi Function Bus Controller (5465)	

Table 3. The MF-bus Card Rack - old type

Card Position	Card Type	Comments
1	MFbus controller (5465)	
2	MFbus port (5155)	
3	Ports or memory by choice	
4	"	
5	ND-5000 CPU 4	Pos 5 - 8
6		
7		
8		
9		
10	ND-5000 CPU 3	Pos 10 - 13
11		
12		
13		
14		
15	ND-5000 CPU 2	Pos 15 - 18
16		
17		
18		
19		
20	ND-5000 CPU 1	
21		CPU type 1: pos 20 and 21.
22		CPU type 2: pos 20 - 22.
23		CPU type 3: pos 20 - 23.
24		

Table 4. The ND-5000 Card Rack — new type

1.2.2 THE SMALL CABINET

Figure 3 shows the ND-5000 Compact cabinet with its main modules. The new and old card racks are shown in table * and 6

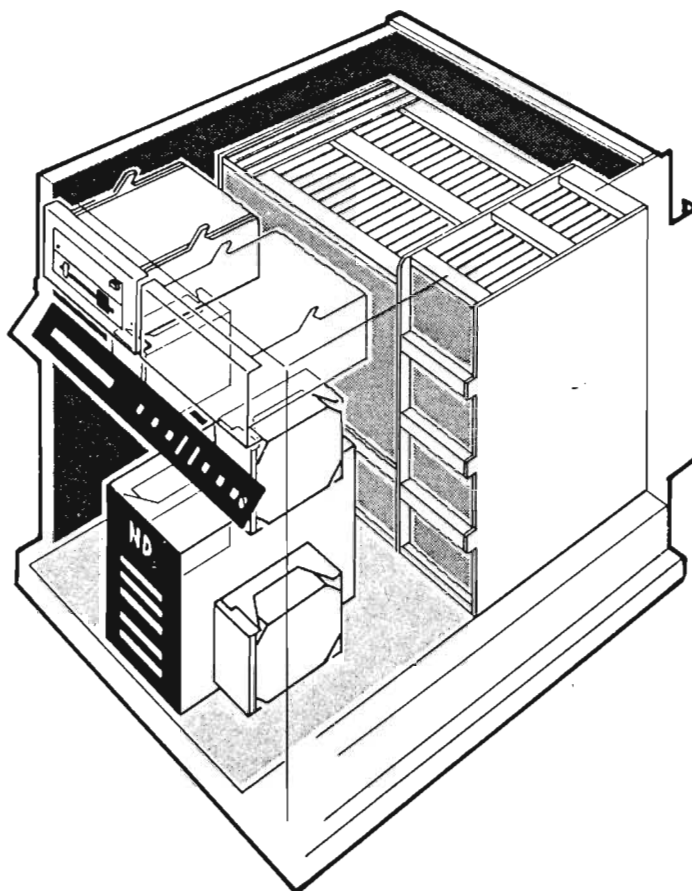


Figure 3. The ND-5000 Compact Cabinet

THE CARD RACK

Table 5 shows the ND-5000 in a double bus backwiring used in the Compact cabinet for models A0-A4 and model B.

THE ND-5000 COMPACT CARD RACK		
Card Pos.	Card Type	Comments
ND-500 size cards	1 ND-5000 CPU card	CPU type 1: pos 1 and 2. CPU type 2: pos 1 - 3.
	2	
	3 (MFB Dynamic RAM)	
	4 MFB Dynamic RAM	4, 8 or 16 Mbyte Dynamimic RAM
	5 Double Bus Controller	
ND-110 size cards	6 ND-110/CX CPU	
	7 Tracer/Memory	
	8 HDLC/Megalink/Memory	
	9 "	
	10 8 term./PIOC/Memory	
	11 "	
	12 "	
	13 ST506 Disk Controller/Triangel/Memory/Free	
	14 Floppy-SCSI/Free/Memory	
	15 Free	
	16 Free	
	17 Free	
	18 Plug board 1	From pos.10: 8-term./PIOC From pos.5 & 6: Console From pos.6:Telefix/Tfix-prt
	19 Plug board 2	From pos.11: 8-term./PIOC From pos.8: HDLC/Megalink
	20 Plug board 3	From pos.12: 8-term./PIOC From pos.9:HDLC/Megalink

Table 5. The ND-5000 Compact Card Rack

Table 6 shows the ND-5000 in a double bus backwiring used in a Compact cabinet for models A10-A14 and model B.

THE ND-5000 COMPACT CARD RACK		
Card Pos.	Card Type	Comments
ND-500 size cards	1 ND-5000 CPU card	CPU type 1: pos 1 and 2.
	2	
	3 (MFB Dynamic RAM)	CPU type 2: pos 1 - 3.
	4 MFB Dynamic RAM	
	5 Double Bus Controller	4 , 8 or 16 Mbyte Dynamimic RAM
ND-110 size cards	6 ND-110 CPU	
	7 Tracer/Memory/Ethernet/(Token ring)	
	8 HDLC/Megalink/Memory/Ethernet/(Token ring)	
	9 "	
	10 8 term./PIOC/Memory	
	11 "	
	12 "	
	13 Floppy & SCSI controller	
	14 Free	
	15 Free	
	16 Free	
	17 Free	
	18 Plug board 1	
	19 Plug board 2	
	20 Plug board 3	

Table 6. The ND-5000 Compact Card Rack — new type

1.3 THE POWER SUPPLY

This section gives a short introduction to the power supplies in the ND-5000 computers. This is a new generation MPS (Multi Power System). The system consists of one or more modules, depending on the power consumption in the different cabinets.

This power supply will to a larger extent than earlier models also supply internally installed peripherals, like 5 1/4" disk drives, floppy drives etc.

THE SMALL CABINET

The small cabinet is equipped with one power supply:

DC110

This power supply is accessible from the front of the cabinet. It is of the plug-in-type, and it can be removed by loosening the four screws in the front and pulling it out (see fig.).

The DC110 supplies the following voltages and currents:

5V/120A
12V/15A
5V/7A Standby

- +5V/120A does mainly supply power for the CPU and the memory.
- +5V/7A standby supplies power for the memory in the case of a power failure.
- +12V/15A supplies the power for peripheral equipment mounted in the compact cabinet, like 5 1/4" disk and floppy.

For more information (switches, LEDs etc.), see the description on page 6.

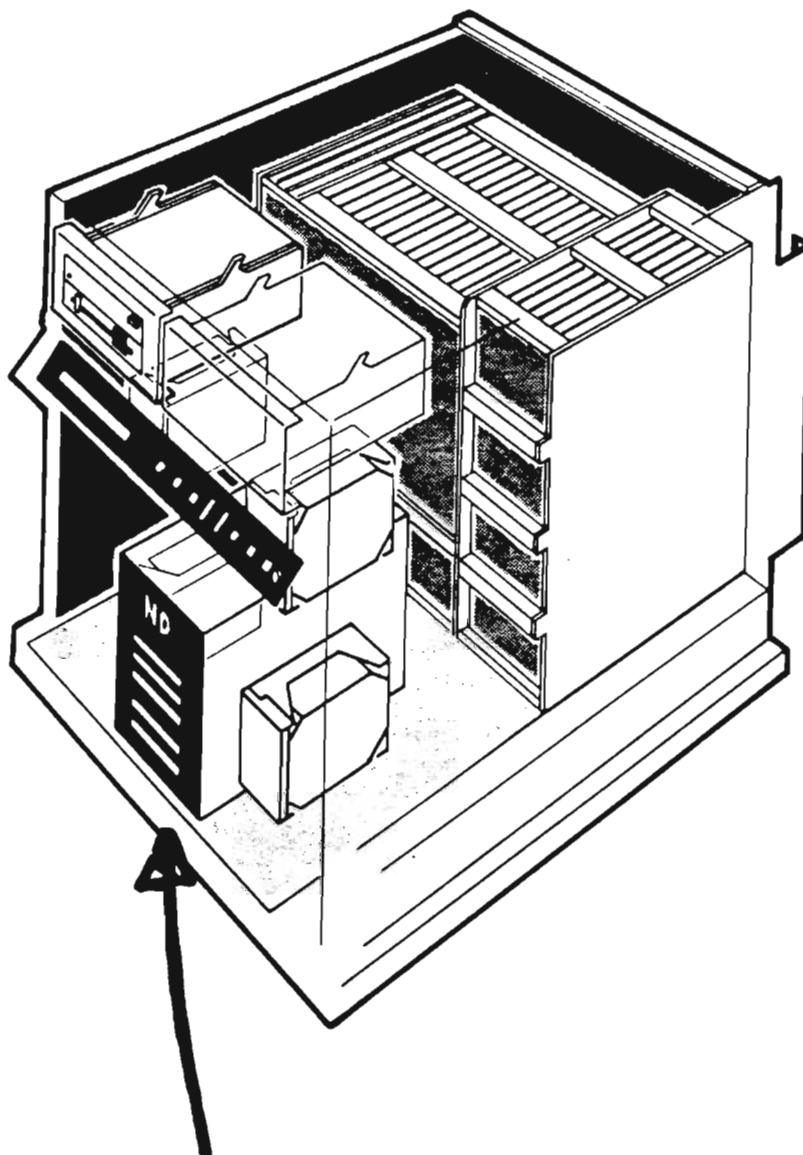


Figure 4. Power Supply in the ND-5000 Compact Cabinet

THE LARGE CABINET

The ND Multi Power Supply is mounted in the upper part of the cabinet, accessible from the rear:

They are of the plug-in-type, and they can be removed by loosening the four screws in the front and pulling it out.

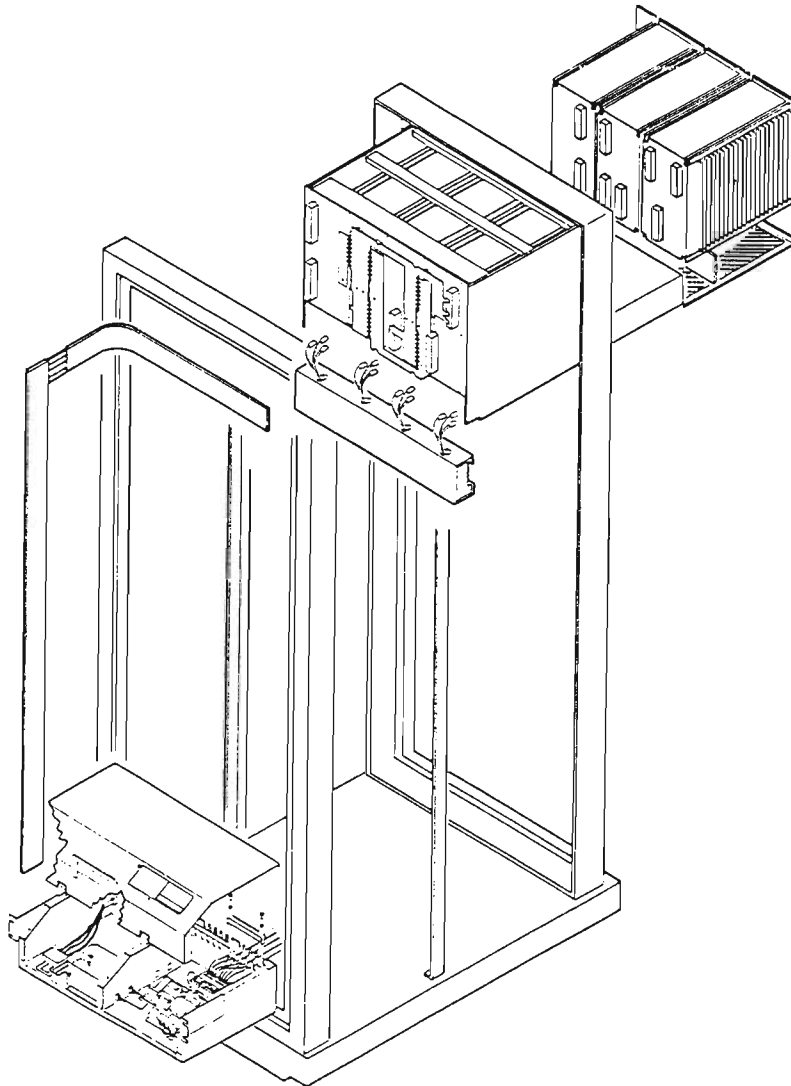


Figure 5. Power Supply in the ND-5000 Large Cabinet

f

The Cabinet, rear view

DC110	DC200	DC300	DC400
5V/120A 12V/15A 5V/7A STB	5V/200A	5V/50A STB	

The different modules:

- DC110 - Delivers the following voltages and currents:

5V/120A
*5VSTB/7A
12V/15A

*In this case DC110 is used in a multi power configuration together with other modules, and the standby power (STB) is not active. STB is in this case taken care of by DC300.

- DC200 - Delivers the 5V/200A.
- DC300 - Delivers the 5V/50A standby for 10 minutes.
- DC400 - For future use.

Switches, LEDs etc This is a short description of the switches, LEDs etc. found on the front of the modules (see fig 6):

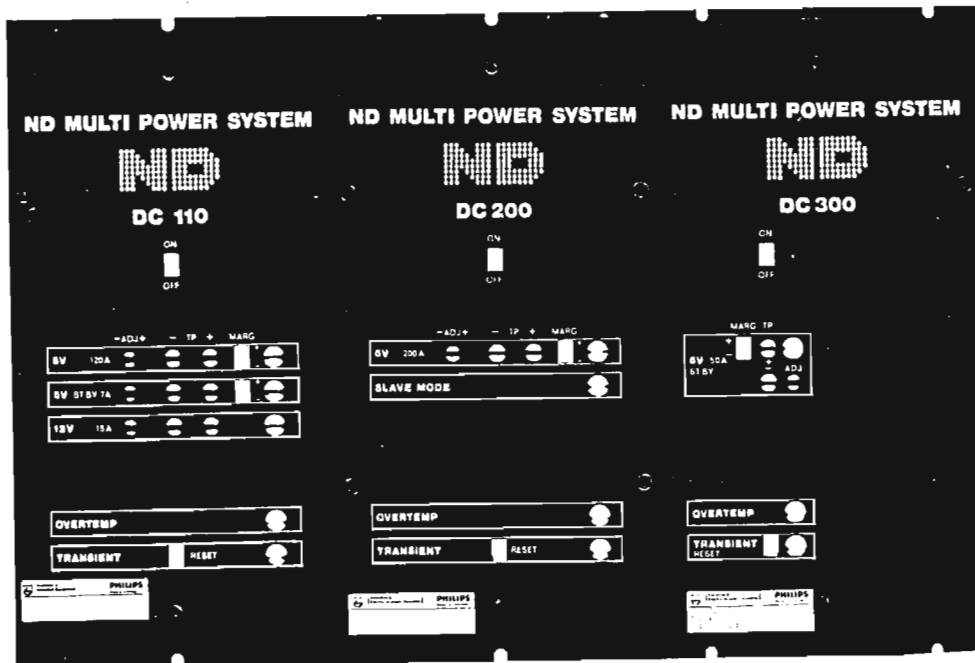


Figure 6. The Switches and the LEDs on the Power Supply Modules

- ON/OFF SWITCH - This switch is used to turn the power module ON or OFF.
- ADJUSTMENT - The screws called Adj is used to adjust the output, which can be checked on the test points (TP).
- MARG - These switches increase/decrease the output with +/- 5%. This will NOT activate the transient indicator if the voltages are correct.
- OVERTEMP - This red LED is lit if the internal temperature exceeds 90 degrees C.
- TRANSIENT - This red LED is lit if any voltage varies more than 10% from nominal value. The LED must be reset by the RESET-switch, which has three positions:
 - The MIDDLE position (default) enables both the visual and the audible alarm.
 - The UPPER position, where the switch will be locked, disables the the audible alarm. The visual alarm is still enabled.
 - The LOWER position (spring return)

resets the alarm.

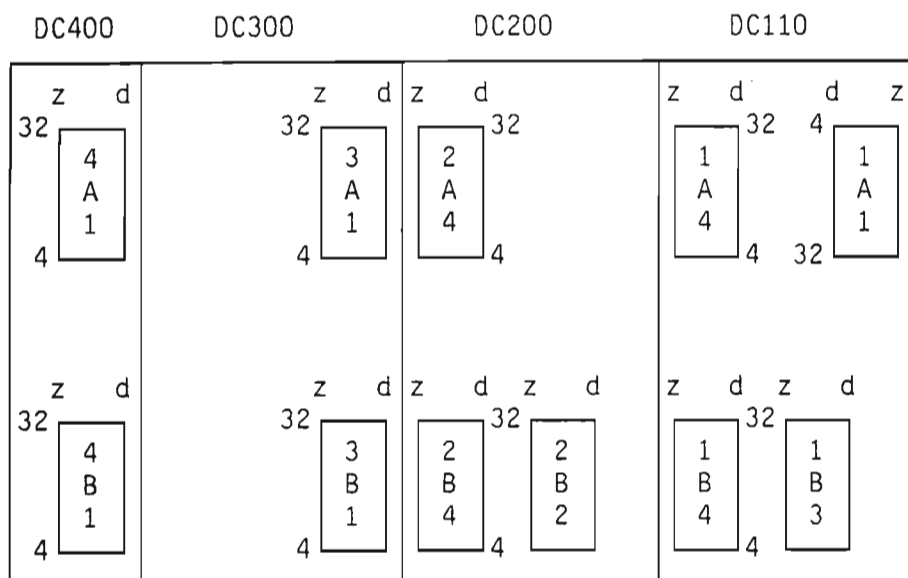
- SLAVE MODE - This green LED is lit when DC200 delivers power in slave mode (controlled by DC110). When the load is below approx. 16 A, DC110 will supply the 5V alone, and this LED is not activated.

	DC110			DC200	DC300
	+5V	+5V SB	+12V	+5V	+5V SB
Maximal current	120 A	7 A	15 A	200 A	50A
Rippel, bandwidth 30 Mhz (mVpp)	50	50	100	50	50
Overvoltage protection	>=6V	>=6V	>=15V	>=6V	5.8 -6.5V
Overscurrent protection	130-150A	7.7-9.8A	16.5-21A	220-240A	55-60A
Short-circuit protection	4V	4V	10V	4V	4V
Switch-on current	Max. 150A peak				
Margin control	+/-5%			+/-5%	+/-5%
Static regulation for "worst case" combination of line og load change	Max.+/-1%			Max.+/-1%	Max.+/-1%
Voltage adjustment	+/- 5-7%			+/- 5-7%	-
Temperature coefficient output voltage, per °C	Max.+/- 200ppm (1mV/°C)			Max.+/-200ppm (1mV/°C)	Max.+/-200ppm (1mV/°C)

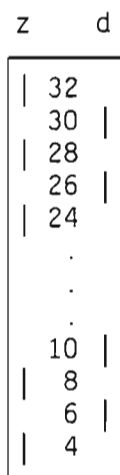
Table 7. Specifications for the ND-5000 Power Supply

The Plug Connectors

Note that connector 1A1 on DC110 is turned upside down.



The connectors on the rear side of the power modules have a lay-out like this:



SIGNAL-LISTSDC110 plug 1A1

D06	Not used	Z04	Not used
D10	12V return	Z08	12V return
D14	+12V	Z12	+12V
D18	+5V SB	Z16	+5V SB
D22	5V SB return	Z20	5V SB return
D26	Common	Z24	Remote off
D30	Memory inhibit	Z28	Common
		Z32	Power fail int.

DC110 plug 1A4

Z32 to Z04	5V return	D30 to D06	+5V
------------------	-----------	------------------	-----

DC110 plug 1B3

Z32	5V sense +	D30	Battery +
Z28	5V sense -	D26	Battery -
Z24	Vc (0-5V)	D22	Rem. 5V SB marg
Z20	Vc return	D18	Rem. +5V marg
Z16	Common	D14	N.C.
Z12	N.C.	D10	Neutral (net)
Z08	N.C.	D06	Line (net)
Z04	GND (net)		

DC110 plug 1B4

Z32 to Z04	5V return	D30 to D06	+5V
------------------	-----------	------------------	-----

DC200 plug 2A4

Z32 to Z04	5V return	D30 to D06	+5V
------------------	-----------	------------------	-----

DC200 plug 2B2

Z32	5V sense +	D30	Vc return
Z28	5V sense -	D26	Vc (0-5V)
Z24	Power fail int	D22	Common
Z20	Optional jump.	D18	Remote OFF
Z16	Optional jump.	D14	N.C.
Z12	N.C.	D10	Neutral (net)
Z08	N.C.	D06	Line (net)
Z04	GND (net)		

DC200 plug 2B4

Z32 to Z04	5V return	D30 to D06	+5V
------------------	-----------	------------------	-----

DC300 plugg 3A1

D06	+5V SB	Z04	5V SB return
to		to	
D30		Z32	

DC300 plug 3B1

D06	Not used	Z04	Not used
D10	Not used	Z08	Sense +
D14	Rem. 5VSB marg.	Z12	Sense -
D18	Power Fail Int.	Z16	Common
D22	Not used	Z20	Remote OFF
D26	Neutral (net).	Z24	Not used
D30	Line (net)	Z28	Not used
		Z32	Ground (net)

POWER FAIL/RESTART

Large cabinet (new type)

The power sense signal from the +5V power is connected to the PFI plug on the plug board (Print 5234) located in the backplane at the rear side of the MF-bus controller.

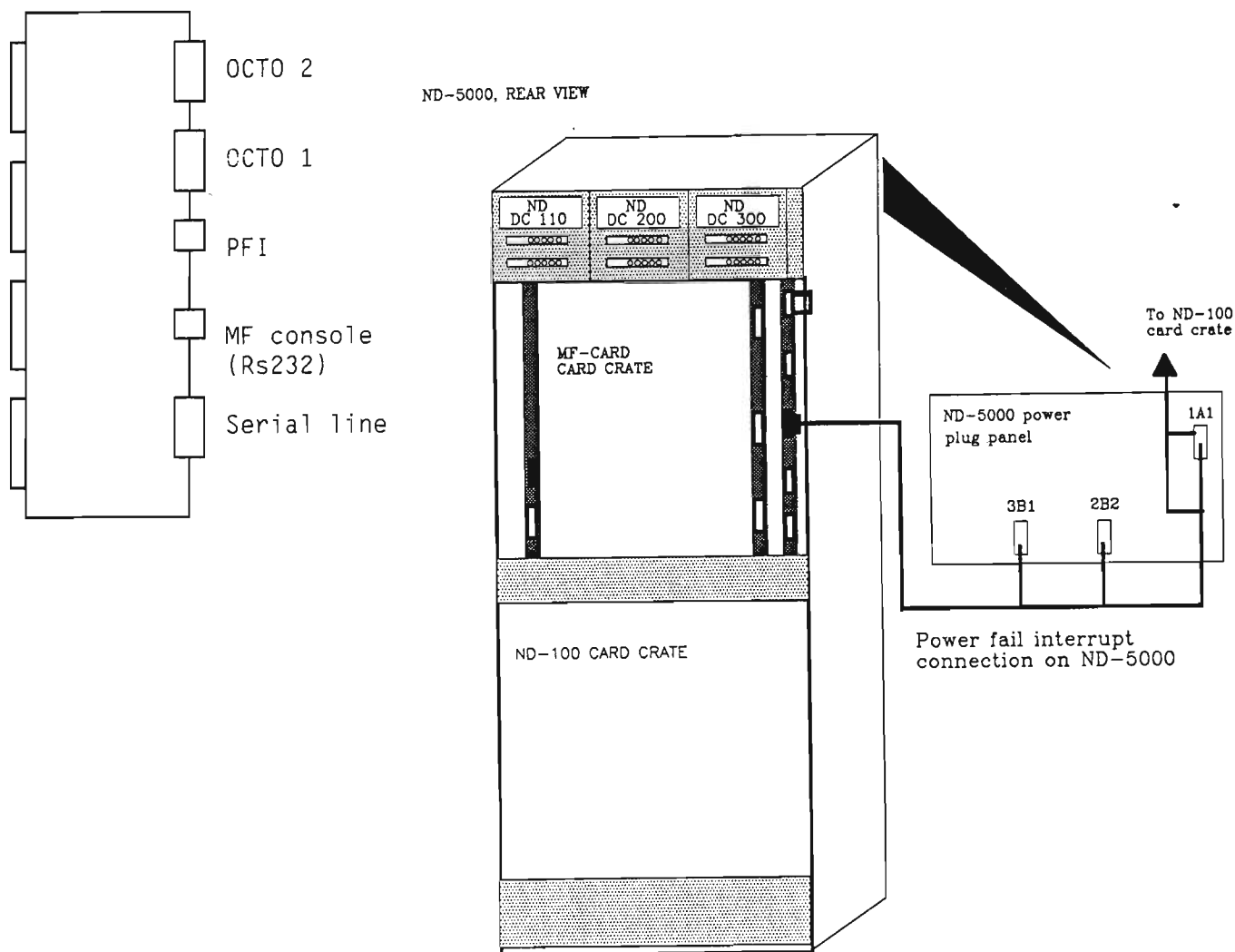
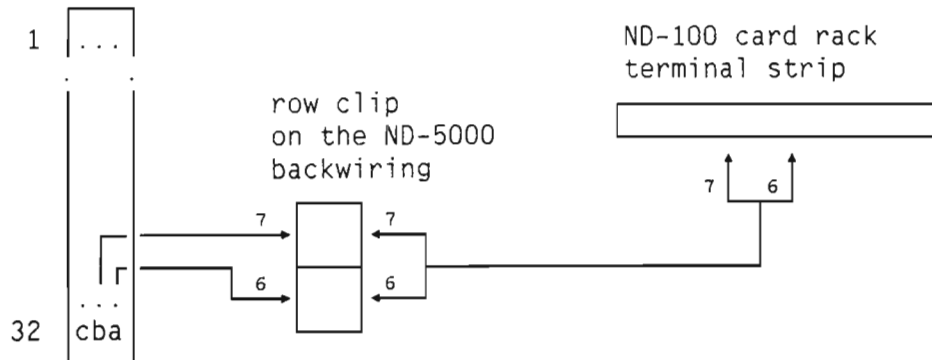


Figure 7. Power Fail Interrupt Connection on the Large Cabinet (new type)

Large cabinet (old type)

The power sense is taken care of by a kit (No 323338) mounted as shown:

MF-crate pos.26



The Power fail connection on ND-5000 in large cabinet, old type

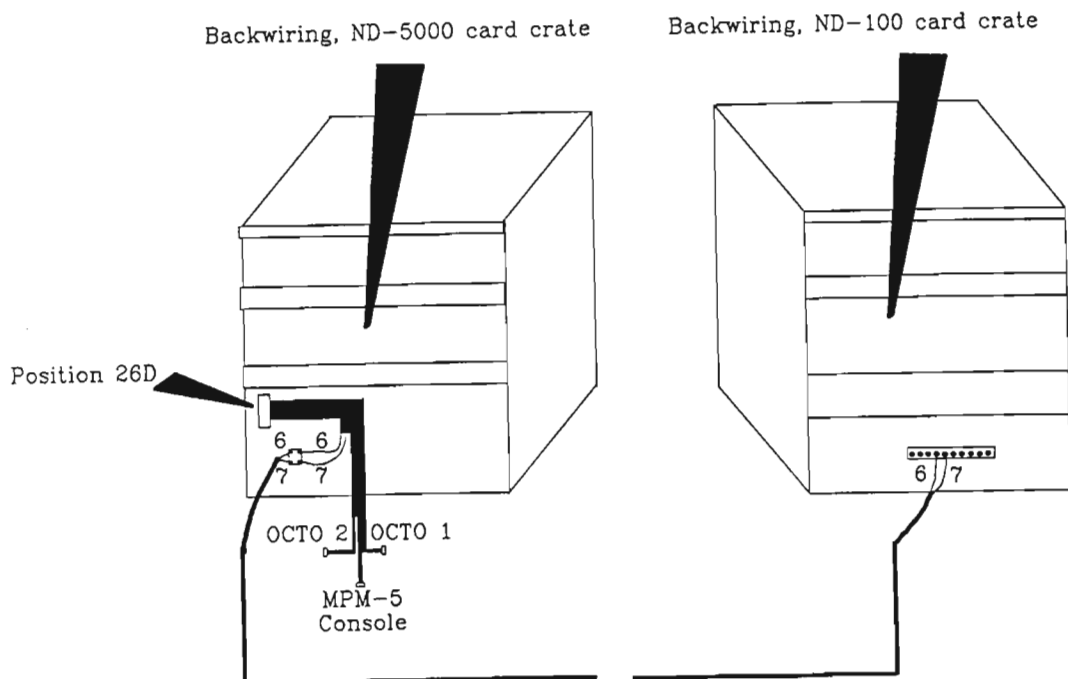
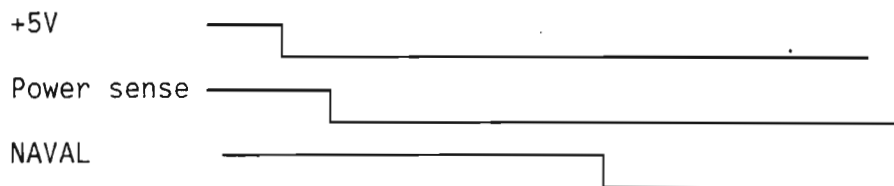


Figure 8. Power Fail Interrupt Connection on the Large Cabinet (old type)

When power fail occurs, the Context block and the dirty memory locations (if WICO mode) must be written to memory.



| ← approx 100ms → |

Dump dirty and
save context
block

The different modules must at least have Eco level:

- MPM5 Dyn Ram 5411 U
- Double Bus Controller 5464 C
- MFB Controller 5454 D
- MFB Controller 5465 B
- ND5000 Mother Board 5502 9b
- ND5000 ACCP Module 5602 5c
- Backplain partno. 324801 1c
(Only if "old" cabinet)
- ND110 (Rask 2) F
- Sintran III WM 406: - Patchfile 5400

**ND-5000 Compact
cabinet**

Power sense is taken care of in the backwiring.

1.4 THE CPU-TYPES

The ND-5000 CPU consists physically of

- a mother card
- Up to 3 layers of baby cards

The baby cards are placed on the mother card in a sandwich construction. The first layer of baby cards contains hardware to increase the CPU performance.

ND-5000 BASIC CPU TYPE 1

The CPU type 1 module will cover the CPU model 2 (ND-5200 system). Part number: 320001.

The CPU type 1 consist of two layers:

1. layer: MB
2. layer: ALU IDA MMS CS MIC ACCP

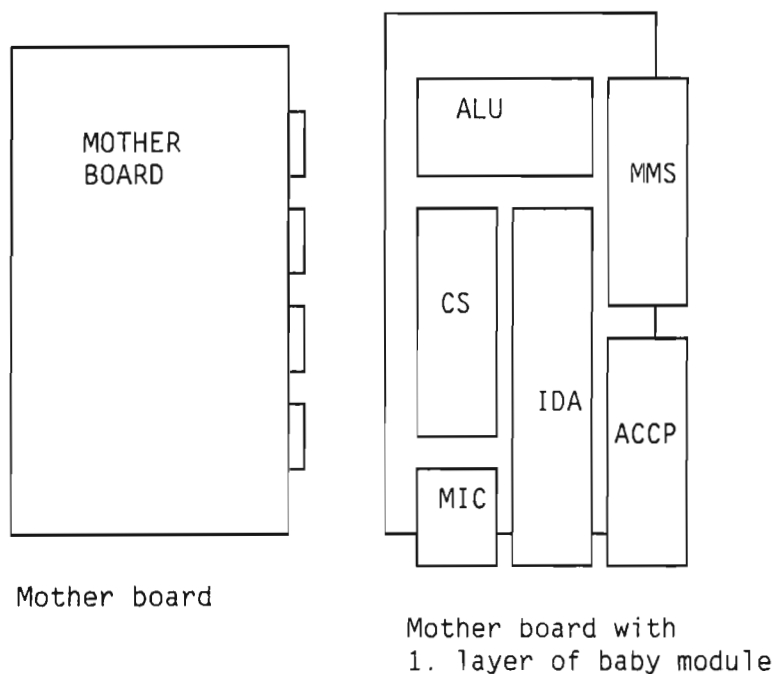


Figure 9. Layout of CPU type 1

Part no.	Module
324602	MB :Mother board
324701	MM :Instruction/data memory management controller
324702	ACCP:Access processor
324704	ALU :Arithmetic Logical unit
324707	CS :Control store - 16 K
324708	IDA :Instruction/data address controller
324709	MIC :Micro instruction controller

ND-5000 BASIC CPU TYPE 2

CPU type 2 covers the CPU model 4, 5 and 7 (ND-5400, ND-5500 and ND-5700 systems).

Part number: 320002.

CPU type 2 consist of 3 layers:

1. layer: MB
2. layer: CACHE ALU
3. layer: AAP IDA MMS CS MIC ACCP

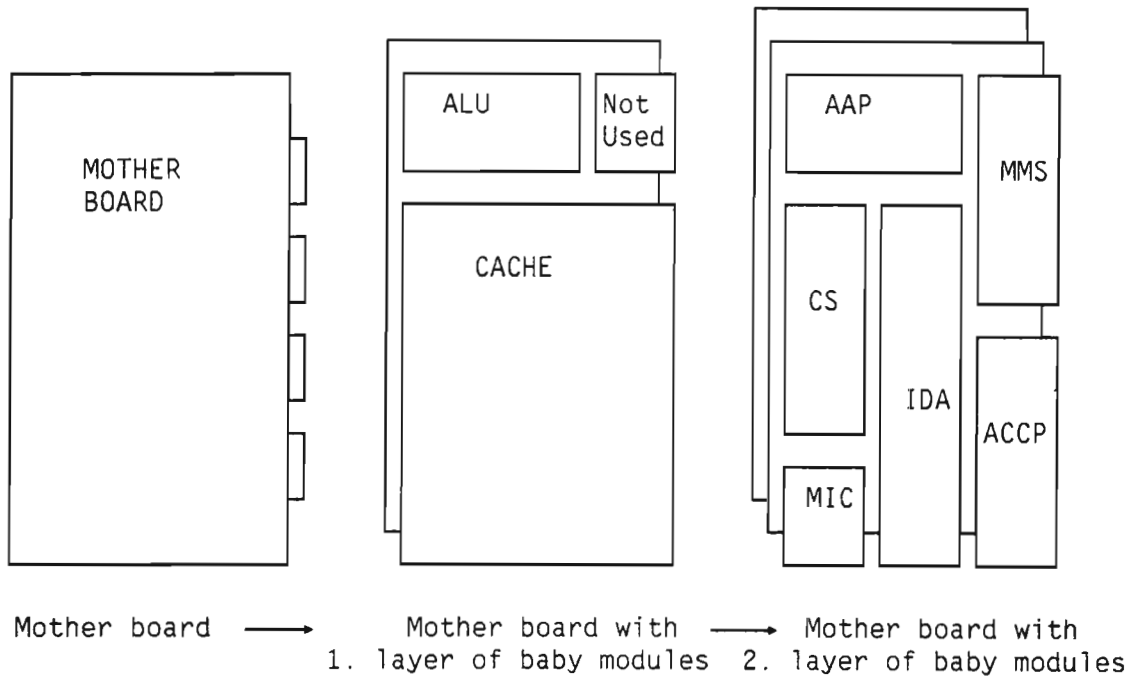


Figure 10. Layout CPU type 2

Part no.	Module	
324602	MB	Mother board
324701	MMS	Instruction/data memory management controller
324702	ACCP	Access processor
324704	ALU	Arithmetic logical unit
324707	CS	Control store
324708	IDA	Instruction/data address controller
324709	MIC	Micro instruction controller
324710	CACHE	Instruction/data cache module
324715	AAP	Additional arithmetic processor

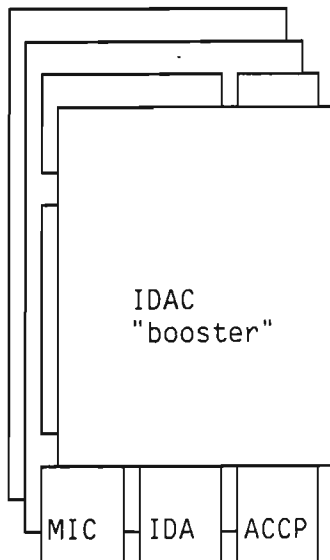
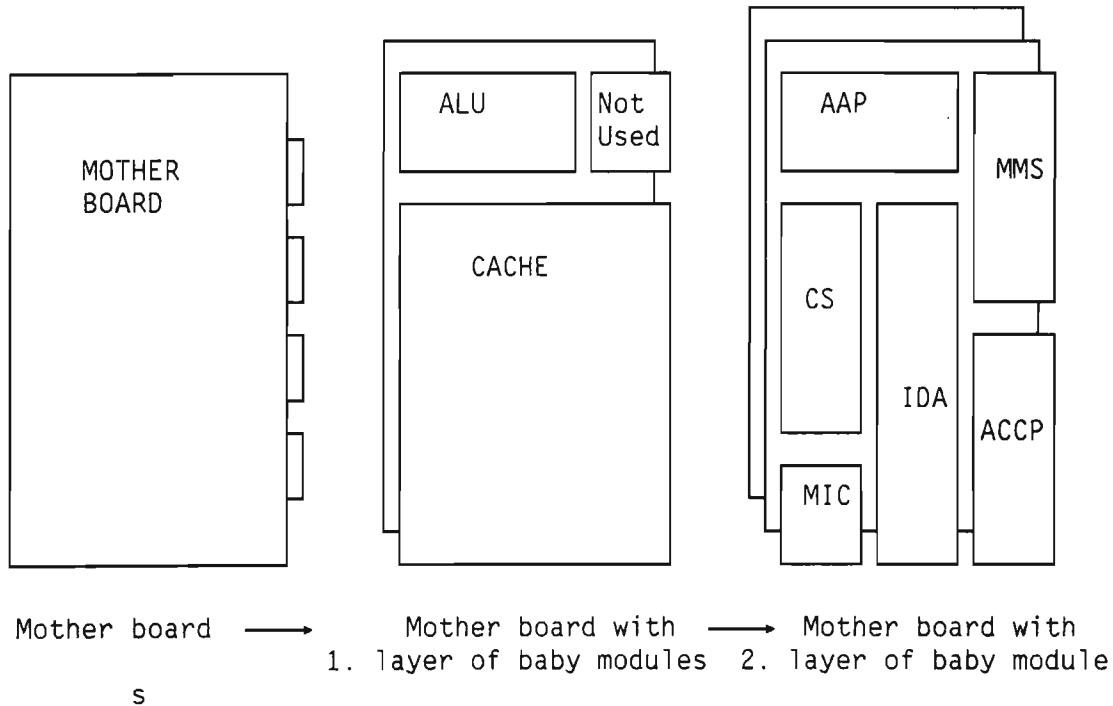
ND-5000 BASIC CPU TYPE 3

CPU type 3 covers the CPU model 8 (ND-5800 and ND-5900 systems).

Part number: 320003.

CPU type 3 consist of 4 layers:

1. layer: MB
2. layer: CACHE ALU
3. layer: AAP IDA MMS CS MIC ACCP
4. layer: IDAC



Part no.	Module	
324603	MB	Mother board
324701	MMS	Memory management controller
324702	ACCP	Access Processor
324704	ALU	Arithmetic Logical unit
324707	CS	Control store - 16 K
324718	IDA	Instruction/data address contr.
324709	MIC	Micro instruction controller
324717	CACHE	Instruction/Data Cache module
324714	IDAC	I-level Data Address Controller
324715	AAP	Additional Arithmetic Processor

Figure 11. Layout CPU type 3

THE DIFFERENCE BETWEEN THE CPU-TYPES

The table below shows the parameters signifying the different models.

ND5000 system:	CPU type:	Enabled function:	Disabled function:	Master clock speed:
ND-5200	1			Normal(70 ns)
ND-5400	2	Instr.Cache SIFGOC	Data cache Addr.cache	Slow (156 ns)
ND-5500	2	Data cache Instr.cache SIFGOC	Addr.cache WICO	Slow (156 ns)
ND-5700	2	Data cache Instr.cache Addr.cache SIFGOC	WICO	Normal(70 ns)
ND-5800	3	Data cache WICO Instr.cache Addr.cache SIFGOC		Normal(70 ns)

Table 8. The difference between the CPU-models

SIFGOC : Smart IFGO Control (Smart ifgo strategy enabled)
WICO : Write In Cache Only (Write ones strategy ("dirty"))

MICROPROGRAMS VERSIONS

PRE-RELEASED VERSIONS FOR SINTRAN K - WM406:

Version:

ND5000 w/microprogr.F1.p..(ND-5200):

File name: MIC-5200-23-400:DATA 11023

AAP.04 with eco > 2a

ND5000 w/AAP4 without FMUL/DMUL....:

File name: MIC-5400-23-400:DATA 11123

File name: MIC-5500-23-400:DATA 11223

File name: MIC-5700-23-400:DATA 11323

File name: MIC-5800-23-400:DATA 11423

RELEASED VERSIONS FOR SINTRAN K - WM406:

ND no.	CPU	Filename	Version:
211272	ND-5200	MICRO-5200-Axx:DATA	110xx
211273	ND-5400	MICRO-5400-Axx:DATA	111xx
211274	ND-5500	MICRO-5500-Axx:DATA	112xx
211275	ND-5700	MICRO-5700-Axx:DATA	113xx
211276	ND-5800	MICRO-5800-Axx:DATA	114xx

xx: Revision number.

CHAPTER 2 HARDWARE UPGRADING OF ND-5000 SYSTEMS

2.1 SYSTEM DESCRIPTION

ND-5000 Compact configurations

The ND-5000 Compact series is equipped with:

- ND-110/CX I/O Processor (ND-110 in ND-5200 Compact system)
- Internal disks or a controller for external disks
- One Streamer, 125 MB (Option on systems with external disks)
- One floppy-disk drive (1.2 MB capacity)
- 4 to 6 MB memory
- SINTRAN and utilities

All ND-5000 Compact systems are available in two models: A model with internal disks and B model with external disk option. A models include from one to four internal disks of 125 MB capacity each (called models A1 to A4). ND-5200 Compact system includes an extra model with one 60 MB internal disk (called model A0). Model B versions are delivered with a controller for external disks and can be configured with external disks and magtape.

The table below shows the models within each configuration:

System Type	ND-5200 COMPACT		ND-5400 COMPACT		ND-5500 COMPACT		ND-5700 COMPACT	
	Mod A	Mod B	Mod A	Mod B	Mod A	Mod B	Mod A	Mod B
Memory size shared/local	4/2		4/4		4/4		4/4	
Disk size MB	60 - 4X125	Ext.	125 - 4X125	Ext.	125 - 4X125	Ext.	125 - 4X125	Ext.
Streamer as backup media	Yes	No	Yes	No	Yes	No	Yes	No

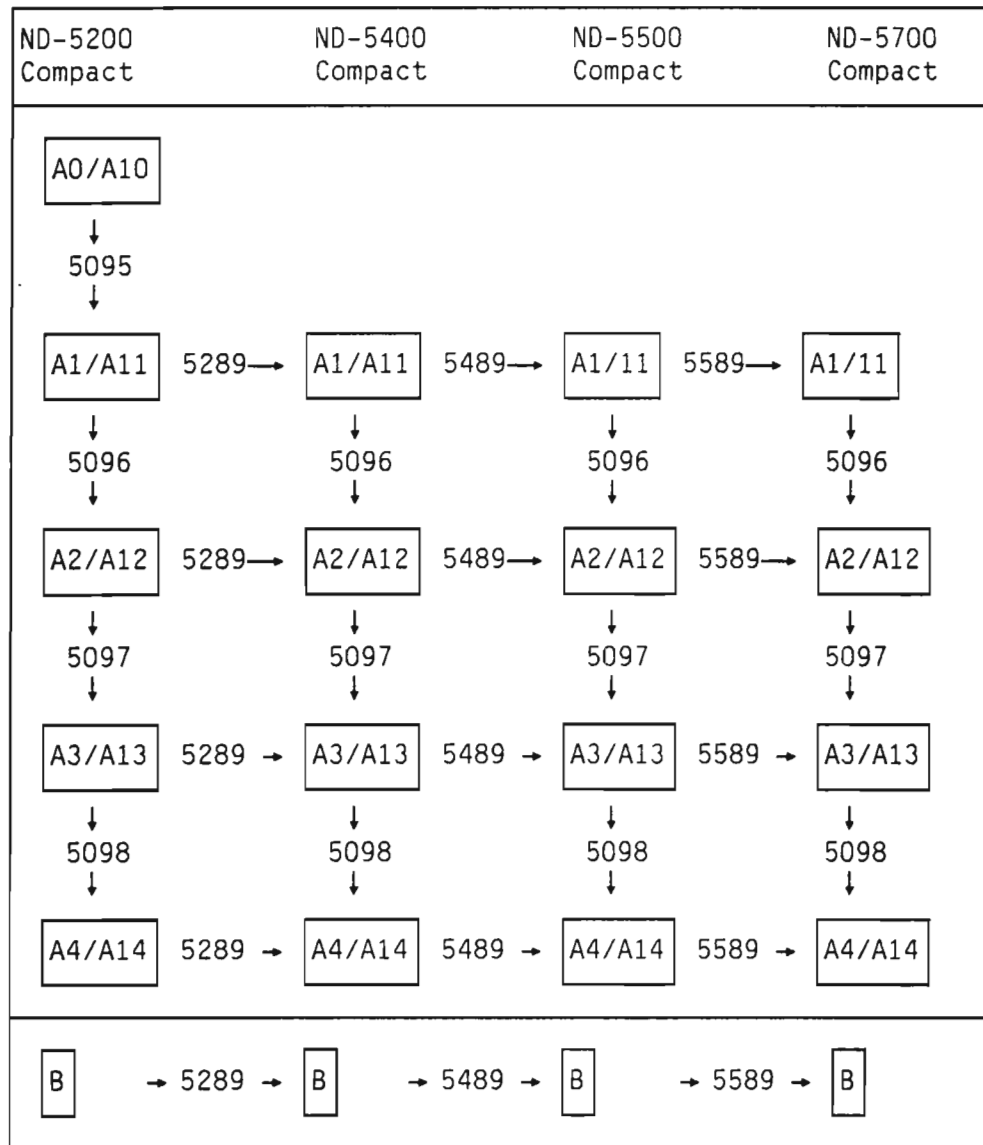
The following table shows the detailed disk configuration for the ND-5000 Compact systems:

System type	ND-5200 Compact	ND-5400 Compact	ND-5500 Compact	ND-5700 Compact
Model:				
A0/A10	60 MB	-	-	-
A1/A11	125 MB	125 MB	125 MB	125 MB
A2/A12	2X125 MB	2X125 MB	2X125 MB	2X125 MB
A3/A13	3X125 MB	3X125 MB	3X125 MB	3X125 MB
A4/A14	4X125 MB	4X125 MB	4X125 MB	4X125 MB
B	External	External	External	External

Upgrading possibilities

Full upgrading is possible from one system to the next one. The difference between Ax to A1x is the cabinet, so upgrading here is not possible. In addition, within each system, all models with internal disk can be upgraded to a model with larger internal disk capacity (i.e., model A0 through A4). However, it is not possible to add external disks to these models.

Upgrading from models A to B and from Ax to A1x is not allowed. The following diagram shows the upgrading paths:



The upgrade kit consist of:

5095

1 * 125 Mb SCSI disk

5096

1 * 125 Mb SCSI disk

5097

1 * 125 Mb SCSI disk

5098

1 * 125 Mb SCSI disk

5289

ND-5400 CPU

ND-120 with 4 Mb onboard memory

Name-label for ND-5400 COMPACT

Micro-program: MIC-5400-xx-400:DATA

5489

ND-5500 CPU

Name-label for ND-5500 COMPACT

Micro-program: MIC-5500-xx-400:DATA

5589

ND-5700 CPU

ND-120/CX with 4 Mb onboard memory

Name-label for ND-5700 COMPACT

Micro-program: MIC-5700-xx-400:DATA

NOTE

Do remember to bring with you the "updating tool" for setting the CPU model in the eeprom in backwiring.

ND-5000 large systems

The large-cabinet ND-5000 systems are equipped with:

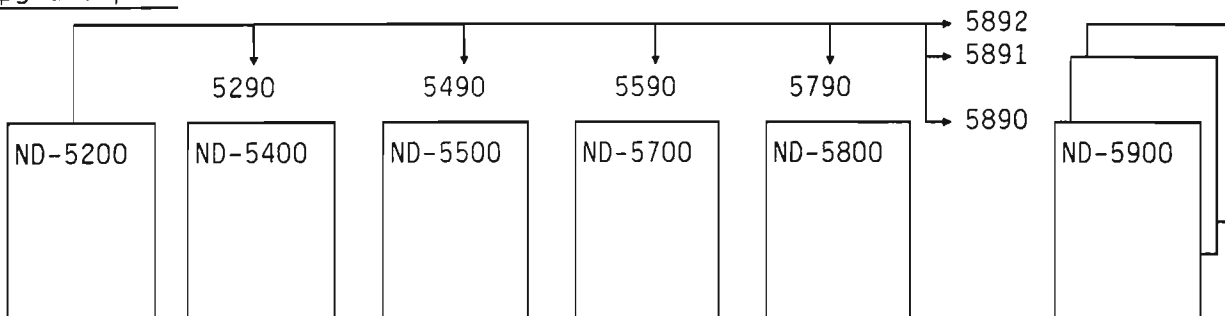
- ND-110 I/O processor in ND-5200
- ND-120 I/O processor in ND-5400 and ND-5500.
- ND-120/CX I/O processor in ND-5700 ==>

The table below shows the configuration for the large cabinet ND-5000 systems:

System type	ND-5200	ND-5400	ND-5500	ND-5700	ND-5800	ND-5900 Model 2	ND-5900 Model 3	ND-5900 Model 4
Memory size shared/loc.	4/2	4/4	8/4	12/6	16/10	24/6	24/6	24/6
Cache size Data (KB)	-		64	64	64	2 X 64	3 X 64	4 X 64
Instruction	-	320	320	320	320	2 X 320	3 X 320	4 X 320
Disk types	extern.	extern.	extern.	extern.	extern.	extern.	extern.	extern

Upgrading possibilities: The following diagram illustrates the upgrading paths for the ND-5000 series:

Upgrade path:



5290

ND-5400 CPU
ND-120 with 4 Mb onboard memory
Name-label for ND-5400 MAXON
Micro-program: MIC-5400-xx-400:DATA

5490

ND-5500 CPU
Name-label for ND-5500 MAXON
Micro-program: MIC-5500-xx-400:DATA

5590

ND-5700 CPU
ND-120/CX with 6 Mb onboard memory
Name-label for ND-5700 MAXON
Micro-program: MIC-5700-xx-400:DATA

5790

ND-5800 CPU
Name-label for ND-5800 MAXON
Micro-program: MIC-5800-xx-400:DATA

5890

ND-5800 CPU
Print for ACCP consol (in backwiring)
Name-label for ND-5900 MAXON

5891

ND-5800 CPU
Print for ACCP consol (in backwiring)
Name-label for ND-5900 MAXON model 3

5892

ND-5800 CPU
Print for ACCP consol (in backwiring)
Name-label for ND-5900 MAXON model 4

NOTE

Do remember to bring with you the "updating tool" for setting the CPU model in the eeprom in backwiring.

2.1.1 UPGRADING FROM A ND-5200 TO A ND-5400 SYSTEM

- Exchange the ND-5000 CPU from CPU type 1 to CPU type 2.
- Exchange the ND-110 CPU with ND-120 CPU with 4 Mb memory;.
- Use the updating tool (*) to set the CPU model to 4.
- Exchange the ND-5000 microprogram with version 144xx. (Remember to change switch settings for memory limits for MPM port and local 100 memory).

2.1.2 UPGRADING FROM A ND-5400 TO A ND-5500 SYSTEM

- Use the updating tool (*) to set the CPU model to 5.
- Exchange the ND-5000 microprogram with version 145xx.

2.1.3 UPGRADING FROM A ND-5500 TO A ND-5700 SYSTEM

- Use the updating tool (*) to set the CPU model to 7.
- Exchange the ND-5000 microprogram with version 147xx.
- Exchange the ND-110/CX CPU with ND-120/CX-4MB. (Remember to change switch settings for memory limits for MPM port and local 100 memory).

2.1.4 UPGRADING FROM A ND-5700 TO A ND-5800 SYSTEM

- Exchange the ND-5000 CPU from CPU type 2 to CPU type 3.
- Exchange the ND-120/CX-2MB CPU with ND-120/CX-4MB CPU. (Remember to change switch settings for memory limits for MPM port and local 100 memory).
- Use the updating tool (*) to set the CPU model to 8.
- Exchange the ND-5000 microprogram with version 148xx.

2.1.5 UPGRADING FROM A ND-5800 TO A ND-5900/2/3/4 SYSTEM

- Insert extra ND-5000 CPU type 3 (1,2 or 3 extra CPU's).
- Insert "Samson console print" behind each extra ND-5000 CPU.
- Use the updating tool (*) to configurate and set the CPU model to 8 for the extra ND-5000 CPU's:

ND-5000 CPU 1, octobus station no. 70B ND-5000 CPU 2,
octobus station no. 71B
ND-5000 CPU 3, octobus station no. 72B
ND-5000 CPU 4, octobus station no. 73B

(*): Updating tool to be used on ND-5000 Compact systems:

Part no: 350156 Double Bus Contr. updating tool.

Updating tool to be used on ND-5000, large cabinet version:

Part no: 350157 MF Bus Controller updating tool.

NOTE

The updating tool will be available in limited volume and will normally only be needed when upgrading the ND-5000 CPU.

2.1.6 SETTING OF THE ND-5000 CPU MODEL

Setting of the CPU model has to be done when the ND-5000 CPU has to be upgraded or when the content of the EEPROM in the MF backplane is cleared or lost.

UPDATING TOOL TO BE USED ON ND-5000 COMPACT SYSTEMS

Part no: 350156 Double Bus Contr. updating tool.

In this kit only the special PROM are available.

To be able to set the CPU model, exchange these PROMs with the one on the Double bus controller,

PROM version 27/11 -87
pos 16J, 18J, 16K and 18K

UPDATING TOOL TO BE USED ON ND-5000 LARGE CABINET VERSION

Part no: 350157 MF Bus Controller updating tool.

In this kit only the special PROM are available. To be able to set the CPU model, exchange these PROMs with the one on the MF bus controller,

PROM version 11/11 -87
pos 18C, 20C, 22C and 23C

Use the command SET-CPU-MODEL to set the correct model.

EXAMPLE

```
=====
=      MF bus - TEST AND MAINTENANCE PROGRAM -      =
=                                                    =
=      INTERNAL VERSION for 5465 (5454)                =
=      November 11, 1987                               =
=====
```

- * INITIALIZING MF-BUS MEMORY * -

BANK NOT PROPERLY INITIALIZED - NOT AVAILABLE

COMMENT

The MF bus will not be available when these PROMs are used. These PROMs are only to be used during initialization of the MF bus or setting the CPU model on ND-5000 CPU's. To set the CPU model, the command shown below must be used.

>SET-CPU-MODEL↵

DANGER! YOU CAN DAMAGE YOUR SYSTEM

PASSWORD:↵

SLOTNO:6↵

% Slot position of the ND-5000 CPU

CPU:7↵

% ND-5000 CPU model ref. list above.

% Values 2,4,5,7 or 8.

- WRITING TO NONVOLATILE MEMORY, PLEASE WAIT -

NEW-PASSWORD (Y/N):N↵

To verify that the CPU model is correct the following command can be used:

```
>LIST-CONFIGURATION↵
Slotno:6↵
SLOT 06 : ND 5000 MODEL: 00B
STATION NO: 000070B
POWER FAIL DESTINATION: 000001B
BROADCAST TYPE: 000000B
SPEED: 000001B
CPU MODEL: 000007B
MASTER CONTROL REG : 000201B
```

LIMITS THAT DEFINE ACCESS-AREAS FOR THIS SLOT

When the correct ND-5000 CPU model setting has been set, the normal PROMS has to be inserted again on the MF bus controller to be able to run the system.

NOTE

- The MF-bus will not be set available when using the upgrading tool. When setting of the CPU model is finished, exchange from the upgrading tool, back to the old MF controller with correct version of the MF PROM's.
- Check what kind of AAP module (Checkpoint 3) is installed and use the correct microprogram according to CPU model. Ref. overview of the ND-5000 microprograms.

WARNING

If the updating tool is not available, the following commands in the MF maintenance program must be avoided:

>INITIATE-EEPROM with slot number equal to the MF controller.

>CONFIGURATE-SLOT with slot number equal to the ND-5000 CPU and the configuration is saved.

These two commands will destroy the CPU model setting for the ND-5000 CPU.

CHAPTER 3 OCTOBUS COMMUNICATION

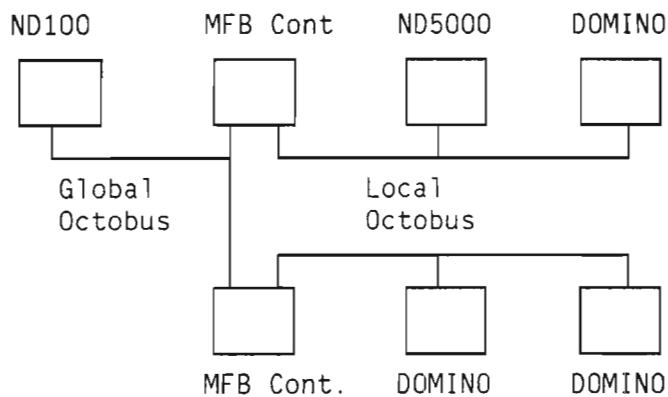
The octobus is a fast serial bus optimized for handling of short messages. A maximum of 62 stations (processors) can be connected to one bus. The octobus is used in low level operating system to provide interprocess synchronization and exchange of configuration parameters during initiation. The octobus will also be used as the communication medium between system components for debugging and maintainance. The octobus will not be visible above low level operating system. Process/process communication and synchronization within the operating system as well as at the application level will be provided by NUCLEUS.

3.1 INTRODUCTION TO OCTOBUS HARDWARE

The Octobus can be divided into a Global and Local Octobus. Only a device connected to the Global Octobus may be MASTER of the octobus chain. All devices connected to the Octobus chain is given an unique station number.

Definitions of Octobus station numbers:

Station no:	Octobus device
1	ND-100
2 - 7	MFB Controllers
10-13	SCSI Controllers (disk)
14-15	Matra VME
16-17	Multifunction communication
20	Hyperchannel
21-23	FDDI (Fibernet)
24-27	FPS-5000
30-33	Graphic controller
34-67	Free for expansion
70-76	ND5000



The Global Octobus consists of 4 differential signals, which is converted to TTL signals on the Local Octobus, which carries the following signals:

- XREQ
- XCLK
- XDAT
- XRFO

With the bus in a quiescent state, the three first lines are off, while, if MASTER is selected, the XRFO line is pulsing with a 15 usec. periode. If XRFO is not pulsing, indicating that no MASTER is selected, the stations connected to the Octobus, will automatically start to assign a MASTER. The one with lowest station number will end up as MASTER and start transmitting refresh signal (XRFO). When a MASTER is selected, the Octobus is ready to transfer messages between any of the stations connected to the bus. A transfer is initiated by a station when activating the XREQ-line. When the MASTER receives this request, it automatically starts to transmit clocks (XCLK) with the frequency specified for the Octobus (1 or 4 MHz). All requesting stations will then start to transmit its message into the Octobus (XDAT). Each requesting station will go on transmitting until it receives a "1" while transmitting a "0" itself. Then it will cease transmitting, wait until the current frame is finished and then start all over. At the time a station gives up, its priority is incremented, so on the next try, the chances are increased for a successful transfer.

3.2 OCTOBUS FRAME FORMAT.

The signals transmitted on the octobus during one frame is a Start and Stop bit pluss 30 bits.

30.....27 26.....21 201918..13 12.....5 4 3 2 1

Priority	Destination	C	B	Source	Information	Parity	Ack
----------	-------------	---	---	--------	-------------	--------	-----

← Direction of transmission.

Priority	Content of "Lost Access Counter"
Destination	When B=0 normal transmission this field contains one of 62 station numbers (1-76B). If B=1 (Broadcast) this field contains one of six station types.
C	If C=1 the attached information is a control byte. If C=0 the information field contain pure data.
B	If B=1 all stations of specified type shall accept this message (Broadcast). If B=0 only the station matching the destination number shall accept this message.
Source	Station number of transmitting device.
Information	One byte of data.
Parity	The number of "1"'s are counted and the two least significant bits of the count are attached to the end.
Ackn	Acknowledge of the frame is returned from the destination device.

Ackn.

- 00 : Timeout - 15 retries
- 01 : Successfully received
- 10 : Destination busy - 255 retries.
- 11 : If B=0
 - Parity error - 15 retries
 - If B=1
 - Ambiguous response

3.3 INTRODUCTION TO THE PROTOCOL

The current protocol used on the Octobus is protocol 5.

There are four separated message streams on the Octobus:

1. IDENT messages routed to IDENT ENTRIES. This message will immediately activate a process in the destination station with correct working set.
2. KICK messages routed to HANDLER ENTRIES. This message will immediately activate a process in the destination station. The destination process will receive kick messages from all stations and has own datastructure to find reason for activation.
3. MULTIBYTE messages routed to OCTOBUS MESSAGE DEVICES (OMD). This message will immediately activate a process in the destination station. The destination process will receive multibyte messages from all stations. Mainly used for initialization, debugging and maintainance.
4. EMERGENCY messages decoded by hardware or octobus driver.

3.3.1 THE MESSAGE FORMAT

The data sent/received has the following 16 bit format when sent on the octobus or read from the FIFO:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
C	B	Dest/Source						INFORMATION							

B=1 Broadcast octobus frame to all stations with specified type.

C=1 The information field contain a command.

Dest/Source When sending a frame on the Octobus this field contains destination station number or Broadcast Type if B=1. When receiving a frame from Octobus this field contains source station number.

The Information field is decoded as follows by the Octobus driver:

15 C	14 B	13 Dest/Source	12	11	10	09	08	07 E	06 K	05 M	04 S	03	02	01	00		
1	0							1	<emergency code>							Emergency msg	
1	0							0	1	<kick number>							Kick msg
1	0							0	0	0	<ident no>					Ident msg	
1	0							0	0	1	1	<OMD no>				Start multibyte message	
1	0							0	0	1	0	<OMD no>				End of multibyte message	
0	0							<data byte>								Part of multibyte message	

E Emergency, only used for hardware messages e.g. power fail, master clear, reset ACCP etc.

K Kick, used as a kick or "wake-up" signal to a handler. Used to activate/terminate ND-5000.

M Multibyte message. If S=1 means start of multibyte message. If S=0 means end of multibyte message. Used when loading control store or issue ACCP commands on ND-5000.

IDENT

Ident message activates a process in the destination station. The destination must be prepared for receiving IDENT messages from a specific station.
Not used yet in ND5000 CPU.

KICK

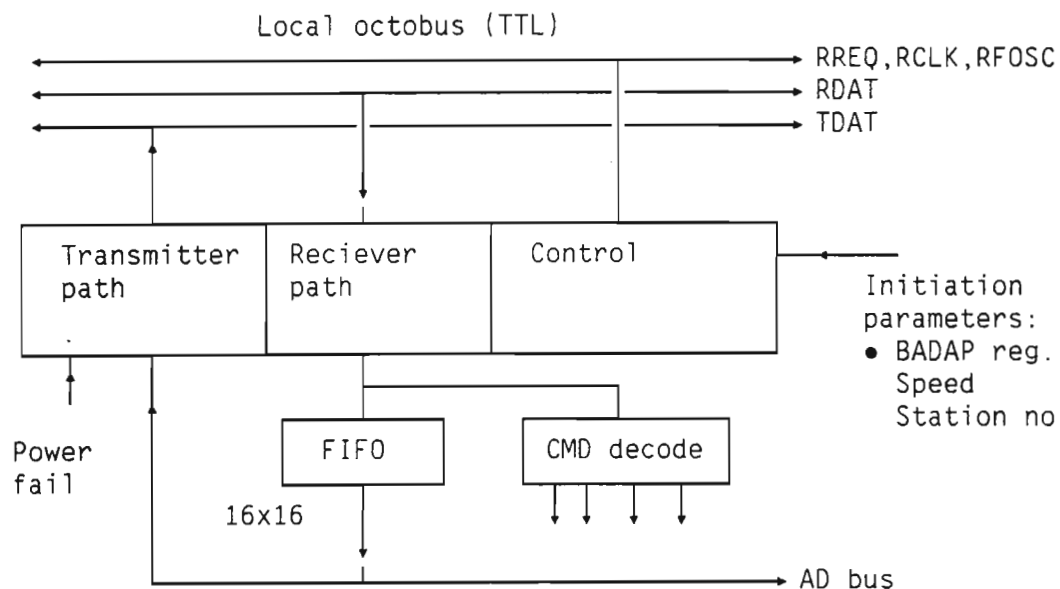
Kick message activates a process in the destination station. A Kick message can be received from all stations. Kick number 1 is used in ND5000 CPU to start scan the execution queue.

MULTIBYTE

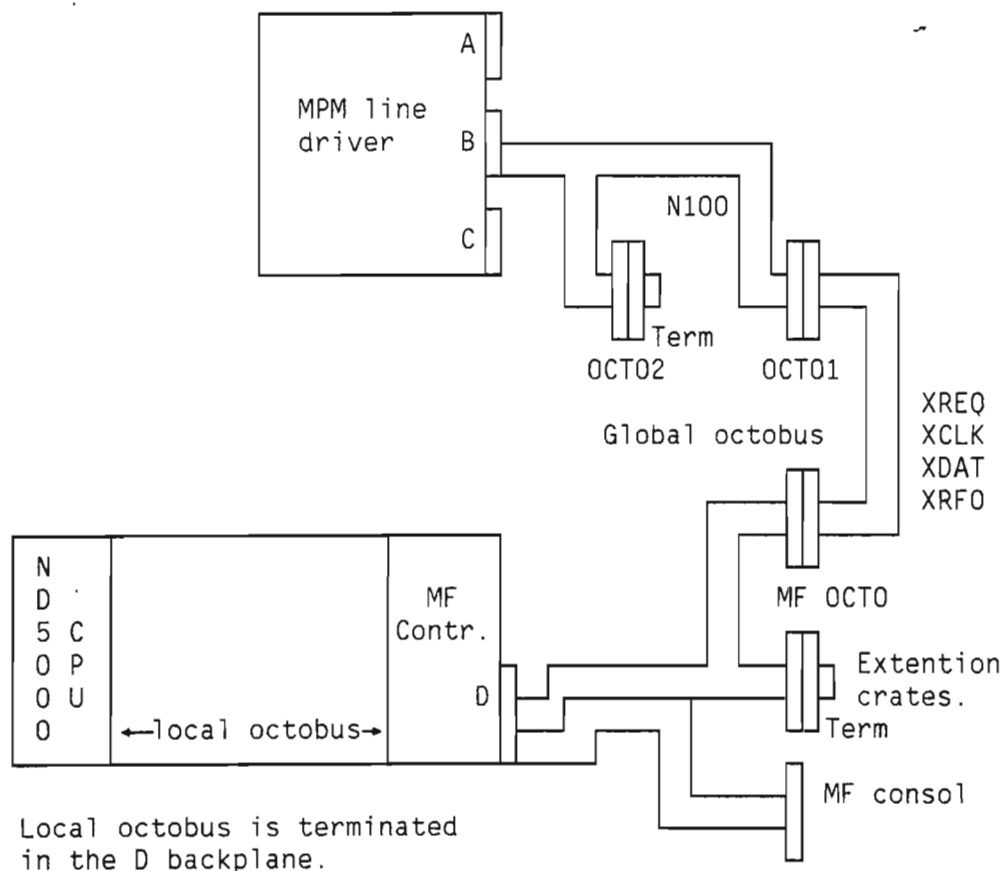
Multibyte messages will be by the destination process routed to the specified OMD routine. In ND5000 CPU multibyte messages with OMD number 3 will be handled by the ACCess Processor.

3.4 OCTOBUS HARDWARE ON ND5000 CPU.

The ND5000 CPU is connected to the local octobus in the MF crate.

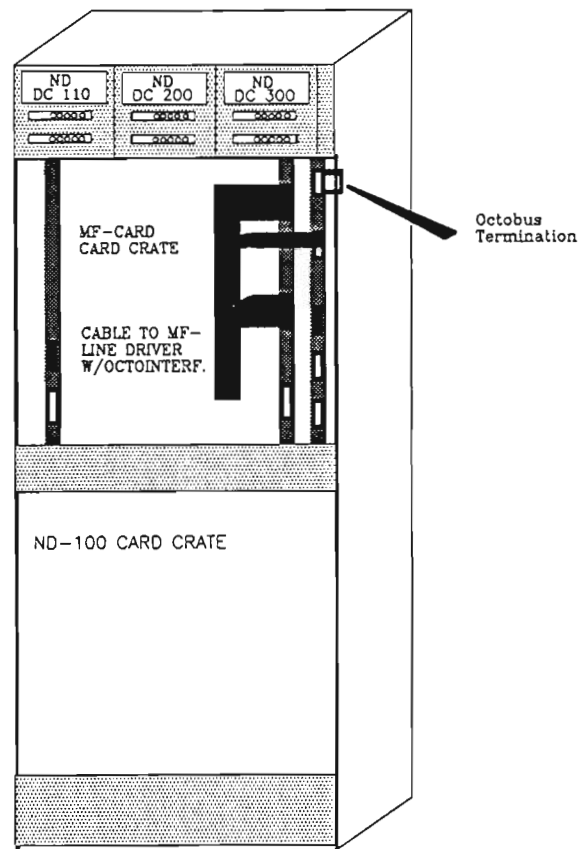


3.5 INTERNAL OCTOBUS CABLE.



Also see the figure on the next page.

ND-5000, REAR VIEW

*Figure 12. Octobus Cabling On ND-5000*

3.6 THE ACCP

During normal operation the ACCP takes care of the OCTObus communication between the ND-100 and the ND-5000. Additionally it checks for errors like timeout, memory errors, power failure, etc. These activities don't require any human interference what so ever. This means that normally, i.e. by delivery, the ACCP is not accessible by anyone.

However, when the ND-5000 is not running, the ACCP can be used to inspect the memory, control store, etc. and to act as a communication link to the ND-5000 CPU while debugging the hardware. To do this, a console must be connected in the following way:

3.6.1 CONNECTING THE ACCP CONSOLE

When eco level 4 on ACCP is done, the Samson Console print has to implemented to enable communication with the ACCP from a terminal.

In the large cabinet, the ACCP console is connected to the plug card in the backwiring of the MF-crate, located on the rear side of the crate in the same position as the ND-5000 CPU.

Print 5235, Partno. 324195 "Samson console maxson"

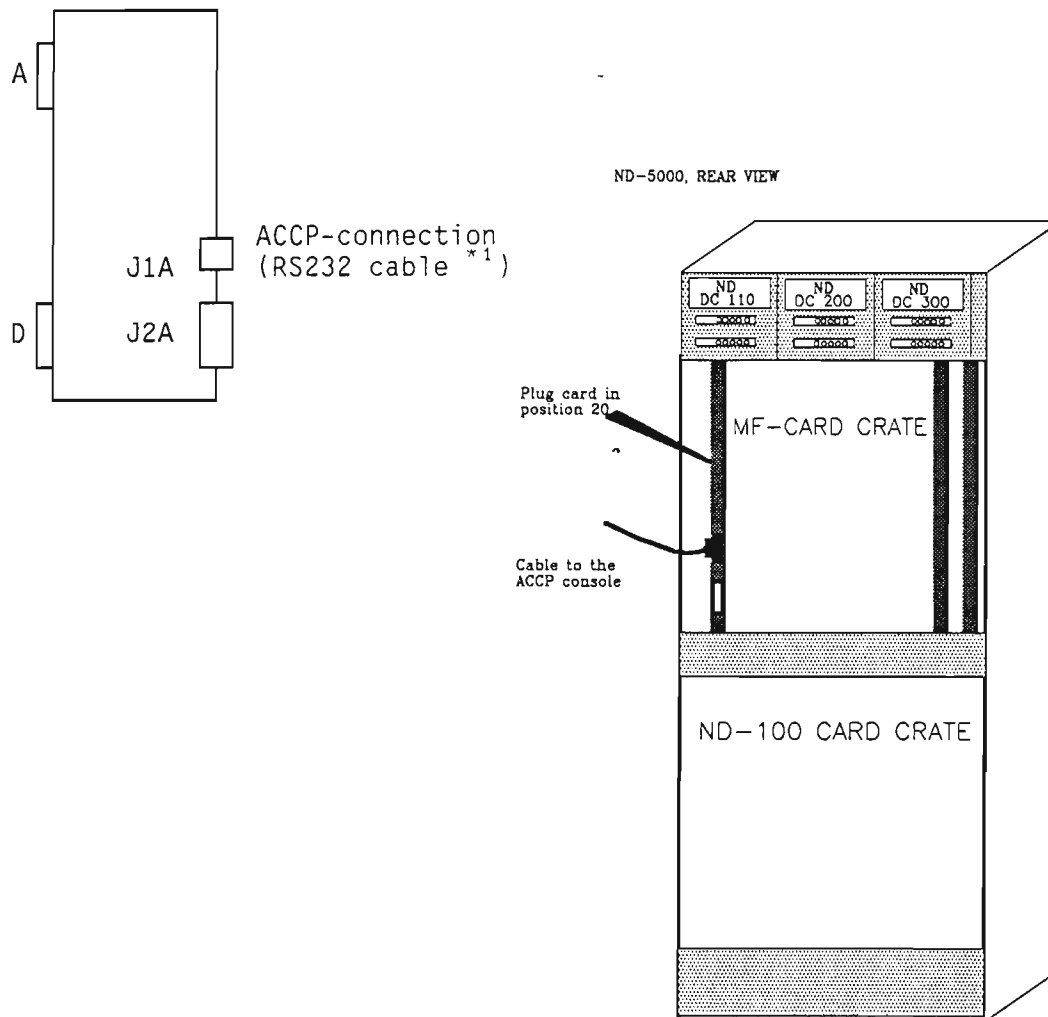


Figure 13. ACCP-Console Connection on the Large Cabinet

- *1) This cable will be delivered together with the new cabinet. Its partnumber is 325705 and it is registered in the MPS system as CABL-EXT 12/21/22xx Domino
This cable will be delivered with each cabinet.

In the ND-5000 Compact cabinet, the ACCP consol is connected to a plug card situated in the middle on the right side of the cabinet.

Print 5236, Partno. 324196 "Samson console comson"

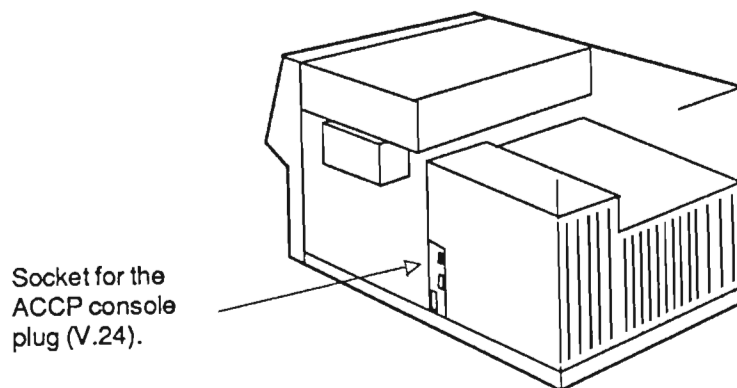
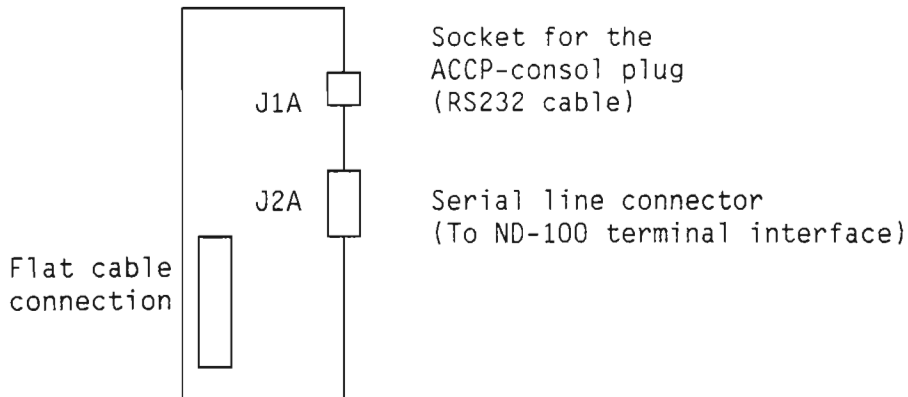


Figure 14. ACCP-console Connection on ND-5000 Compact Cabinet

Some systems has been delivered in the "old" large cabinet. A special kit is prepared for this cabinet version (kit no: 311002). In these cabinets the plug card for ND5000 Compacts (print 5236) are mounted in the rear side of the MF-crate.

NOTE

When eco level 4 on ACCP is done, the ND-5000 Console print has to implemented to enable communicatione with the ACCP from a terminal.

When you have connected the console, type <CTRL X> on the keyboard to perform a software RESET. The display will now look like this:

```
ACCP Software Reset performed
Communication ACCP-ND100 started. Version: .....
ACCP:
```

Typing HELP will display the commands available.

3.6.2 ND-5000 SELFTEST

SAMSON is equipped with a selftest feature which gives a go/nogo test of the CPU at power up/master clear. The test is executed by the ACCP and no interaction with other processors is necessary to perform the test.

After power-up or after ND-5000 CPU receives a Master-Clear command via Octobus, a selftest microprogram is loaded from ACCP proms and executed under control of the ACCP. This takes appr. 20 seconds, and the host (ND-100) must issue the ACCP-command 'Read selftest status' every 2nd-5th second after a Master Clear until the ACCP answers the message. (When the ACCP executes selftests, it does not answer Octobus commands.) This command returns a 16-bit status word where the result of each test is reflected in a particular bit. If a test was successfully completed, the status bit is 0, otherwise 1. This means that all bits should be 0 when all parts of the selftest was OK.

If any part of the selftest failed, the ACCP will toggle the Master Reset signal in order to flash the red LED on the edge of the motherboard. This means that the CPU is dead. The ACCP is still alive, so in order to debug errors all the test systems may be used. Before this is done the command RESET-CPU must be issued, or Ctrl-X typed on the consol.

Note: The selftest started at Master Clear is a short version taking appr. 20s. The full selftest must be started by the command "RUN SELFTEST". This test takes appr. 3 minutes.

THE SELFTEST STATUS

The selftest status is a 16-bit word where each bit refers to the different test. If a bit is set, this test has failed.

Bit 15 has a special meaning: If CPU type is 2, i.e. if the CPU model is ND-5400, ND-5500 or ND-5700, then this bit is set if the MFbus-controller has not been initialized with proper CPU model.

The bits are allocated as follows:

Bit 0: BUS test
" 1: MIR test
" 2: CS test
" 3: START/STOP test
" 4: ARG test
" 5: ALU test
" 6: REG test
" 7: TSB test
" 8: INSTR.CACHE test
" 9: DATA CACHE test
" 10: CONTROL CACHE test
" 11: AAP
" 12: -
" 13: -
" 14: -
" 15: CPU model not initialized
in MF-controller

THE SELFTEST RUN

After a power-fail or MASTER-CLEAR, or when the ACCP-command RUN-SHORT-SELFTEST is given, a selftest will be run in the ND-5000. If this selftest fails, see the description on page 77

The selftest comprises the following sub-tests:

ACCP local RAM test	Tests the local RAM on the ACCP module.
Bus test	Tests the main busses on the CPU
MIR test	Tests the Micro Instruction Register on the Mother Board.
Control Store sample test	Tests some random locations on the Control Store module
Start/Stop microprogram test	Tests that the microprogram can be started and stopped.
A,MARG D,AIB test	Tests mini-arguments
A,SARG D,AIB test	Tests short-arguments
A,LARG D,AIB test	Tests long-arguments
Loading Control Store with selftest	Loads Control Store with the next tests.
ALU verify test	Tests ACCP communication, AOP and FBUS, true and false ALU operations, ALU status, Q-register and FBUS shift, status and trap register, selection of testobjects, loop-counter and index-counters.
Register test	Tests WRF-registers, SRF registers, Modus, HL, LL, L, P, B and R registers and MIC registers.
TSB test	Tests TSB as memory and generates TSB entries.
Instruction CACHE test	Tests the instruction CACHE with walking zero, walking one, address in address. Tests the GTI and USED bits.

Data CACHE test	Tests the data CACHE with walking zero, walking one, address in address. Tests Dirty-directory, dirty and reserved flags.
Control CACHE sample test	Tests that some locations can be copied from Control Store to the Control CACHE.
AAP test	(Not implemented yet)
Selftest completed OK	No errors found by the selftest.

Other useful commands:

VALUE <Convert number>
 HELP <Command>
 LOOK-AT-CONTROL-STORE <CS address>
 LOOK-AT-MEMORY <Address>
 LOOP-ON-NEXT-COMMAND <Supress output text ?>
 MAIN-FORMAT <BASE (HEX,OCT,DEC)>
 READ-ACCP-STATUS
 READ-ECO-LEVELS
 RESET-CPU
 RUN-LONG-SELFTEST
 RUN-SHORT-SELFTEST
 SEND-KICK-OCTOBUS <DESTINATION><Kick value (process)>
 SEND-MULTIBYTE-OCTOBUS <Destination><Subprocess><Message>
 SEND-OCTOBUS <Data (16)>
 SET-CLOCK-SPEED <Clock speed (Slow,Normal,Fast)>
 SET-KICK-TIMEOUT <Kick timeout (ms)>
 SET-SERIAL-LINE <Enable nd100-communication via serial
 line ? (y/n)>
 START-MICROPROGRAM <CS address>
 STOP-MICROPROGRAM
 TEST-BUFFERS <ASR/AOB>
 TEST-BUSLOOP <Test-pattern>
 TEST-MEMORY <From address> <To address>
 TRACE-COMMUNICATION-DATA <Trace communication data to
 consol? (y/n)>

NOTE

The command SET-CLOCK-SPEED will be
 allowed to modify the clock speed
 according to the CPU type and CPU model.

Ref: ND-5000-Hardware Description (ND-05.020) for further
 description of the ACCP-commands.

CHAPTER 4 THE HARDWARE TRACE MODULE

This chapter gives a short description of the use of the hardware trace module.

4.1 THE TRACE MODULE MEMORY

The trace memory of ND-5000 CPU is 160 bits wide and 4K deep static RAM. It covers:

- the three most important busses, MIB, DB and AOP
- the microprogram address
- all "nano-state" identifiers
- the pipeline WAIT signal
- a wired spare signal
- the signal TRIGD.

All these signals will be stored into the memory every micro- or nano-cycle for 4 K cycles, dependent of the prespecified setup of the tracer.

The memory is accessible from microprogram, only for read, in five 32-bits partitions. The content may be read consecutively from address zero after clearing the Trace Address Counter. One of the five read actions, read C-trace, will increment the address counter. To locate the trigger on middle-trace mode, a signal TRIGD (triggered) is available in the C-trace word.

4.2 SOFTWARE CONTROL OF THE TRACE MODULE

The Trace module is controlled by software running in ND100. A set of MON60 functions are available:

Function	162B	: Initialize Trace Module
	163B	: Clear Trace Module
	164B	: Arm Trace Module
	165B	: Disarm Trace Module
	166B	: Dump Trace Module
	167B	: Clear Address Counter

The ND-500/5000 Monitor includes a set of commands to maintain the Trace module on ND-5000 CPU.

INITIATE-TRACER

To initiate the tracer the following command is available in the ND-500/5000 Monitor. This command is privileged and can only be used from user SYSTEM.

INIT-TRACER <trace cycle>,<trace mode>,<trig spec.>
 <csa>,<clear address>

- | | |
|----------------------------|---|
| <trace cycle> | = MASTER - Nanocycles are traced
MIR - Microcycles are traced. |
| <trace mode> | = START - Start at trig and stop at end
of memory.
CENTER - Start when armed, when triggered stop
in oposite memory quadrant.
(Beetween 2 and 3 K are traced after
trig)

END - Start when armed, stop when triggered. |
| <trig spec.> | = MICRO - Trig on a micro programmed trigger
(A,BM02 D,SPEC,CTRACE)
CSA - Trig when MAR = given control store
address <csa>.
ACCP - Receive trigger from ACCP.

WIRED - Trig when the signal WTRIG 0 on
connector C7 pin B27 or XTRIG_1 on
connector C7 pin B31 present.
One of these signals may be strapped
to any signal. WTRIG 0 can be used
if trig on low condition of a
signal is wanted and XTRIG_1 if trig
on high conditon. |
| <csa> | = <value> - Control store address (octal).
Default = 0. |
| <clear addr.> | = Yes - The address counter will be cleared
and the trace memory filled with
a dummy pattern. |

NOTE

The system microprogram will after been started in address 0, initiate the trace module as follows before the IDLE loop is entered.

```
Trace cycle = MASTER
Trace mode  = END
Trig spec   = MICRO
```

Then the address counter is cleared and the trace module is armed.

ARM-TRACER

When this command is given, the tracer will be armed.

DISARM-TRACER

When this command is given, the tracer will be disarmed.

DUMP-TRACE-MEMORY

The trace memory contents are dumped from ND-5000 CPU, and examination are started from the trig point. The terminal screen is filled with the content of the trace memory. Ref. Examine-Trace command.

WRITE-TRACE-FILE

WRITE-TRACE-FILE <filename><comment-line>

<filename> = Any filename. Default file type is :TRAC.

<comment-line> = Any free text within 80 characters.

The text should describe the error situation.

Write the trace data currently being examined to a file. The file may later be recovered for further studies.

Together with the trace data, information about the system, such as time and date, CPU number and ECO levels are stored.

READ-TRACE-FILE

Recover trace data and information from file.

EXAMINE-TRACE

Start examination of current trace. The picture is positioned at the trigger.

Next page contains a sample of a trace investigation picture. The sample shows a END trace with CSA trig on 100B. For navigation during trace examination the following commands may be used:

>	= Position to next page
<	= Position to previous page
Move	= Move to given (trig relative) address
Trigger	= Move to trigger address.
Print	= Print current page to file.
Get	= Search for pattern. Cntrl<P> gives back last pattern searched for. The value X will match any value.
Exit	= Exit from examination picture.

```

Rel.
Trig. MAR    MIB    DB    AOP    WAIT IDU DAC    DCC
      dec oct  hex    hex    hex    bin hex bin    MC IMM DMM MIC
      -18 00645 0000C000 00000004 00000001 1 06 000000001 0F0 87 00 001
      -17 00645 0000C000 00000004 00000001 1 06 000000001 0F0 87 00 001
      -16 00645 0000C000 00000004 00000001 1 06 000000001 0F0 87 00 001
      -15 00645 F8000103 00000004 00000001 1 06 000000001 0F0 88 00 001
      -14 00645 F8000103 00000004 00000001 1 06 000000001 0F0 97 00 001
      -13 00644 F8000103 00000004 00000001 1 08 000000001 0F0 97 00 001
      -12 00644 F8000103 00000004 00000001 1 10 000000001 0F0 97 00 001
      -11 00645 F8000103 00000004 00000001 1 38 000000001 0F0 97 00 001
      -10 00645 F8000103 00000004 00000001 1 18 000000001 0F0 97 00 001
      -9 00645 F8000103 00000000 000047FF 1 30 000000001 0F0 97 00 001
      -8 00645 F8000103 00000000 FFFFFFFF 1 40 000000001 0F0 97 00 001
      -7 00645 F8000103 00000000 FFFFFFFF 1 70 000000001 0F0 97 00 001
      -6 00101 F80000FF 00000000 080265A8 1 70 000000001 0F0 97 00 001
      -5 00101 F80000FF 00000000 080265A8 1 70 000000001 0F0 97 00 001
      -4 00101 F80000FF 00000000 00000001 0 70 000000001 0F0 97 00 001
      -3 00101 F80000FF 00000004 00000001 0 70 000000001 010 97 00 001
      -2 00100 F80000FF 00000004 00000001 1 70 000000001 010 97 00 001
      -1 10243 F80000FF 00000004 00000001 1 00 000000001 0B0 97 00 001
TRIG> 10244 F80000FF 00000004 00000001 0 00 000000001 0B0 97 00 001
Scroll (>,<),(Move,Trigger,Print,E):E

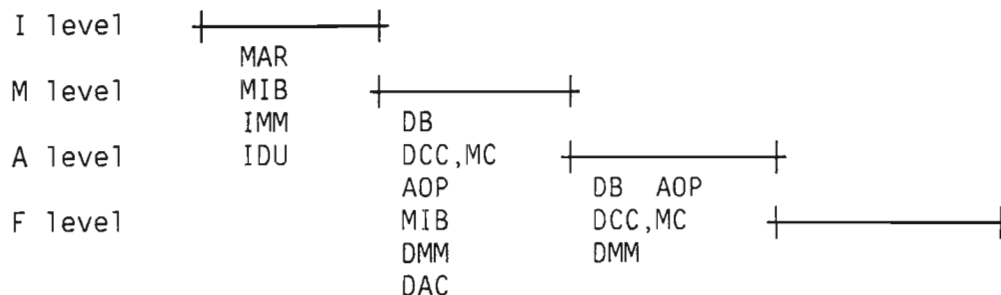
```

MAR	- Control store address. Entry points if no instruction cache hit
MIB	- Activity towards instruction memory Logical program addresses Physical memory addresses Instruction data at wordboundary
DB	- Activity towards data memory. Logical data addresses Physical memory addresses Data on word boundary (no data cache hit)
AOP	- All A operands. Data memory/cache aligned Index register (pre/post addressing mode)
DAC	- State-bit from DAC.
IDU	- State-bit from IDU.
IMM	- State-bit from instruction MM.
DMM	- State-bit from data MM.
WIRED	- Available to be strapped to any signal.
WAIT	- Pipeline wait.
DCC,MC	- State-bit form Data Cache Control and Memory Control.
MIC	- State-bit from MIC.

NOTE

All bits in one trace word are sampled in the same cycle. You therefore have to think about the pipeline structure when analyzing a trace. MAR trace will normally be 2 cycles ahead of the others, which are largely traced on M-level.

The Pipeline structure:



4.3 DUMP OF TRACE MEMORY WHEN AN ERROR SITUATION HAS OCCURED

When an error situation occurs the trace module may contain valid information. This is described in the ND-5000 error message chapter.

The procedure below can be used to dump the trace module and send it to the repair sender together with the defect ND-5000 CPU.

- Enter the ND-500/5000 Monitor from user SYSTEM.
- Dump the trace memory.

N5000:DUMP-TRACE-MEMORY↵

type E↵ to go exit from the examine-trace picture.

- Then write the trace dump to a file by using the command:

N5000:WRITE-TRACE-FILE↵
 File name:"TRACE-DUMP"↵
 Comments:Prot.viol in PED↵

- Copy this file to a floppy discette and send the discette with the defect ND-5000 CPU. In this example the file name is TRACE-DUMP:TRAC.
- Rearm the trace module.

N5000:ARM-TRACER↵

A rearming of the trace module is required if only a dump of trace memory has been done. If the ND-5000 CPU has to be exchanged this command is not required.

4.4 ND-5000 TRACE MODULE DECODING TOOLS

A ND100 program has been made to investigate the trace dump file. The program is called ND5000-Trace Module.

@N5000-TRACER↵

```
*****
**** N D 5 0 0 0   Trace module  DECEMBER 2, 1987   ****
*****
```

Command: HELP↵

```
HELP or ?           - will give you help
EXAMINE-TRACE-MEMORY - Examine trace contents
READ-TRACE-FILE      - Read a trace mem. contents from file
WRITE-TRACE-FILE      - Write contents of trace mem. to a file
EXIT                 - Leave the program
```

Command: READ-TRACE-FILE↵

From file : SAVE-TRACE↵

Date: 87.11.19 10:41:08 CPU no: 19148D Sintran K Rev: 05400
Micro program version : 11323

```
MB.2  ALU.1  AAP.4  IDAC.-  IDA.2  MM.1  CACHE.1  CS.2  MIC.2  ACCP.1
09.b   00.c   02.a   --      03.a  03.c   01.c   03.b   00.b   04.c
```

Text: PROT.VIOL

Command: EXAMINE-TRACE↵

Rel.	Trig.	MAR	MIB	DB	AOP	WI	IDU	IDU	DCC				
D	0	0	0	0	0	WA	STA	SUB	DAC	MC	IMM	DMM	MIC
D	0	0	0	0	0	B	D	0	H	H	H	H	B
TRIG>	00103	01000053113	26000116164	00000000001	00000000001	00	0	0	02F	010	00	00	111
1	14201	01000036107	00000000001	00000000001	10	0	0	0	02F	010	00	00	011
2	00104	01000036107	00000000001	00000000001	00	0	0	0	02F	000	00	00	111
3	00105	01000036107	00000000001	00000000001	00	0	0	0	02F	010	00	00	111
4	00000	01000036107	00000000001	00000000001	00	0	0	0	02F	010	00	00	111
5	14202	01000036107	00000000001	00000000001	10	0	0	0	02F	010	00	00	011
6	00026	01000036107	00000000001	00000000001	00	0	0	0	02F	000	00	00	111
7	14203	01000036107	00000000001	00000000001	00	0	0	0	02F	010	00	00	111
8	00104	01000036107	00000000001	00000024004	00	0	0	0	02F	010	00	00	111
9	00105	01000036107	00000024004	00000000000	00	0	0	0	02F	010	00	00	111
10	00000	01000036107	00000000001	00000000001	00	0	0	0	02F	010	00	00	111
11	14204	01000036107	00000000001	00000000001	10	0	0	0	02F	010	00	00	011
12	14205	01000036107	00000000001	00000000001	00	0	0	0	02F	000	00	00	111
13	14206	01000036107	00000000001	00000000001	00	0	0	0	0AB	010	00	00	111
14	14211	01000036107	00000024014	00000000000	00	0	0	0	02F	010	00	00	111
15	14212	01000036107	00000000001	00000000001	10	0	0	0	02F	010	00	00	111
16	14212	01000036107	00000024014	00000000001	11	0	0	0	02F	243	00	00	111
17	14212	01000036107	00000024014	05003000000	11	0	0	0	02F	242	00	01	111
18	14212	01000036107	00000024014	05003000000	11	0	0	0	02F	242	00	00	111
19	14212	01000036107	00000024014	05003000000	11	0	0	0	02F	242	00	00	111

Scroll (>,<), (Move,Trigger,Print,Get,Format,Backw,E):E

Command: EXIT↵

CHAPTER 5 MAINTENANCE

This chapter gives some hints and advice on different maintenance tasks.

5.1 PREVENTIVE MAINTENANCE

Make a habit of checking the fans and voltages each time you open up the ND-5000 cabinet. Some parts collect a lot of dust, so regular cleaning is good preventive maintenance and increases system uptime.

5.2 FANS

Check that all fans are running.

Inspect the fans mounted on top of the cabinet to make sure that all of them are working.

The other fan trays are of the "plug-in" type which are disconnected from the AC mains when you take them out of the cabinet. All fans are of the same type and size. The best way to find out if an individual fan is working or not is to pull out the fan tray without first turning off the mains switch. As the fans will continue to rotate for a while after they have been disconnected, you will be able to see if one or more fans go markedly slower than the others. In that case they should be replaced.

The power supplies have their own fan tray, mounted under the power supplies on the back of the cabinet.

Figure 15. The fans

5.3 VOLTAGES

Check the voltages when the computer is fully configured. Attach a voltmeter to the points marked TP (test point) on the power supply. Adjustments are made by turning the screws marked ADJ (adjustment) till the voltmeter shows the correct value. See section 1.3 for details.

When this is done on all power modules, you can check that there are no open circuits or voltage drops. This is done by measuring the voltage at the point where the bus-bar is connected to the ND-100 crate. The drop in voltage from the power supply to the card crate should be below 50 mV, and it is not necessary to do any further adjustments.

A battery pack is located inside the standby power supply DC300. The battery pack is a plug in module, fastened by two screws behind the front plate on the right hand side. If you suspect that the battery is flat, note that there is a 20 Amp fuse on the at the back of the battery module. This fuse will go if the battery plug is shortened by a mistake. The battery is supposed to last for 4-5 years without any maintenance

The power modules themselves don't need any maintenance.

5.4 LEDs

The LEDs are described in detail in chapter 8. On the mother board of the ND-5000 CPU, a yellow LED (number three, counting from the top) tells if the tracer has been triggered. The tracer consists of some memory and logic is reserved for supervising what is happening inside the CPU. If something goes wrong, all activity prior to this error condition will have been logged and stored in the trace module. It is extremely important to save this information for later use. This can be done as follows:

Copy the trace memory to the shared memory by giving the monitor command

DUMP-TRACE-MEMORY

The trace data will also be displayed on your terminal, but you can exit from the screen picture to be able to save this trace onto a file by the command

WRITE-TRACE-FILE

Now you can do a regular floppy dump, and send the floppy and a written statement of what really happened together with the defective module to the repair centre.

5.5 THE ECO-SYSTEM

It is now possible to read the current ECO status of the module from the ACCP console or a terminal. This is done by typing READ-ECO-LEVELS in the ACCP command mode, or by typing VERSION in the ND-5000 monitor. See example of the using this command in the chapter "Debugging commands in the ND-5000 monitor".

The VERSION command will also give additional information about the microprogram and so on. The ECO status may also be read from small paper labels on the module itself, as usual. New CPUs from stock will also have their ECO level printed on the packaging.

All ECO's on the ND5000 module will be carried out by the Central Repair Centre. This because of the compact layout of this unit and the need for special tools. But it is necessary to know how to find current ECO level on the ND5000 CPU.

Normally the ECO level is found on the paper label on the card edge of the mother board.

Example:

324602	S.no.	ECO:	320001	S.no.	ECO:
↑ Part no.	↑ Serial no. for the modul	↑ Eco level for the modul	↑ CPU type	↑ Serial no. for the whole modul	↑ Eco level for the whole mod

If the label is missing, it is necessary to determine the ECO level out of what the N5000:VERSION command display.

DETERMINE CURRENT ECO LEVEL

The monitor command N5000:VERSION will display the version of the microprogram and the ECO number on each baby module and the Mother Board. This is possible because of a strap-field and a print status PAL on each baby module that is coded to the current ECO level which can be read out by SW.

When we have found the current ECO numbers on each module comprised in the ND5000 CPU, we can decide the ECO level on the CPU by using the ECO-register(Ref.: Service Hand Book). To use this correctly, we need to know the rules behind this register and the procedure to determine the ECO level on the CPU.

This procedure will be demonstrated by an example:

STEP 1:

ND-5000: VERSION

Subsystem part: 88. 2.18 REV.-J00 BETA.
System part...: 88. 3. 1
Swapper.....: 88.03.04
Micro program.: 11825
Cpu type.....: 5700
Accp version...: 87.10.16A01

Module:	MB.2	ALU.1	AAP.4	IDA.2	MMS.1	CS.2	CACHE.1	MIC.1	ACCP.1
ECO no:	12a	1c	2a	5a	3c	3b	2c	0b	7c

This output has to be used together with the example of a ECO-register on the next page.

STEP 2: For each module, underline the first occurrence of the current ECO number. Read from bottom and upwards. Procedure: (Do check that you are in the right column. Two different card layout for the ACCP. In this example the actual values will be: MB -033, ALU -028, AAP -036, IDA -029, MM -015B, CS -021, CACHE -023, MIC -014B, ACCP.01 -034.

STEP 3: Look for the uppermost underlined ECO number. (In this example AAP.04 -036) If an arrow is present on this line, this means that this line is not legal, and you have to use the one above. Underline this line. With this line, the ECO level of the module is defined.

STEP 4: For each module, compare the current ECO number with the number indicated under ECO level. Any mismatch here means that you are running a combination of babies that is not tested against each other. The ND-5000 CPU should then be exchanged.

Procedure

The MB, ECO no. -033 has to be checked against eco-level -036. Here, 12a is found on both places and this is legal. The ALU -028 must be checked against -036. (This is also legal, because there is only a dummy {D} on the C-print).

If you don't have the last ECO level on your module, put attention to the field-action field. The star code means:

- No field action.
- * If symptoms detected. The ECO is to be executed only when the symptoms specified occurs.
- ** Must be done. The ECO must be executed by next P.M.
- *** Urgent! The ECO must be executed as soon as possible.

Unnecesarry ECO-level updating is waste of time and money, so please think twice before exchanging the module for ECO updating:

"Do the error really concern the ND-5000?"

Some general rules behind the ECO-register:

LINE NO	EXPLANATION
	-From one level to a higher one, only one ECO is issued.
020-020	-Introduction of a new PCB. (CS)
029-030	-Only an ECO on one of the PCBs, ALU d print. Dummy (D) on c.
021-022	-ECO has to be carried out on <u>all</u> PCB versions of a baby module. (MB.02)
018B-20	-An ECO requires another ECO on another module.(IDA,MB)

5.6 SUBSTITUTING MODULES

Field service is to replace defective parts of the ND-5000 based on trouble shooting. Note that you must always replace the whole ND-5000 CPU. The reason is that special tools are needed to disassemble this module.

Before removing or replacing cards in the card racks, you must remember to turn off the power. This is done by turning the key switch on the operator panel OFF. When you have earthed yourself using the wrist strap, you can pull the relevant card out of the crate.

After having installed a new card, you must make sure that this one is working better by running the same software that failed before.

5.7 THE CPU TRANSPORT BOX

CPUs are packed and stored in antistatic bags inside boxes specially designed for storage and transport. Inside this box there is an antistatic frame made of polystyrene. The box is marked with the CPU type, part number and ECO level. It is especially important that the CPU does not leave this packaging before it is to be installed in a machine.

NOTE

You must earth yourself to the cabinet before you touch the CPU card.

The defective CPU must be placed in the packaging together with trace-dump floppy and error messages and sent to the Central Repair Centre.

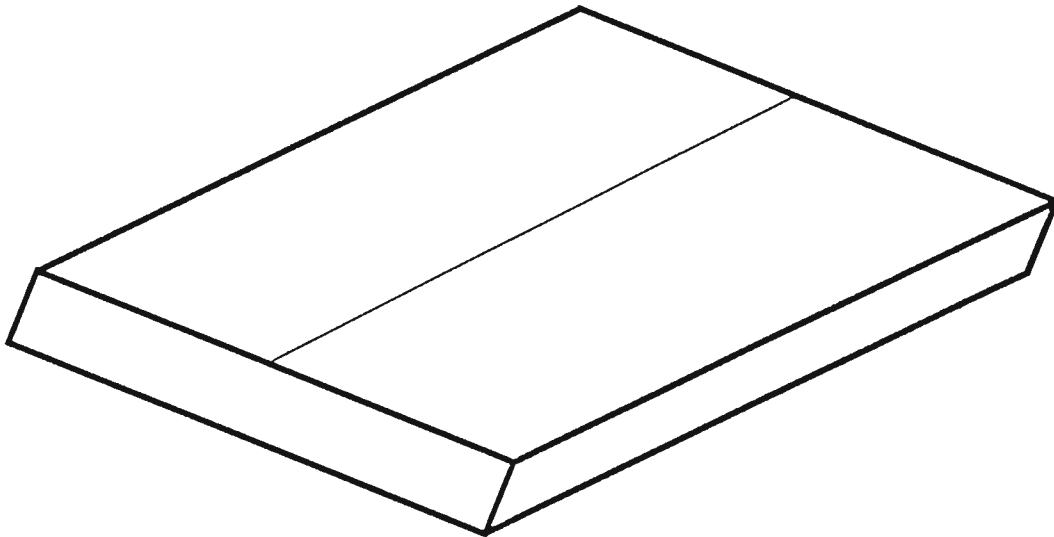


Figure 16. The CPU transport box

CHAPTER 6 TROUBLESHOOTING

6.1 ND-100 ERROR

This flow-chart refers to the proper actions to be take when SINTRAN enters different error conditions.

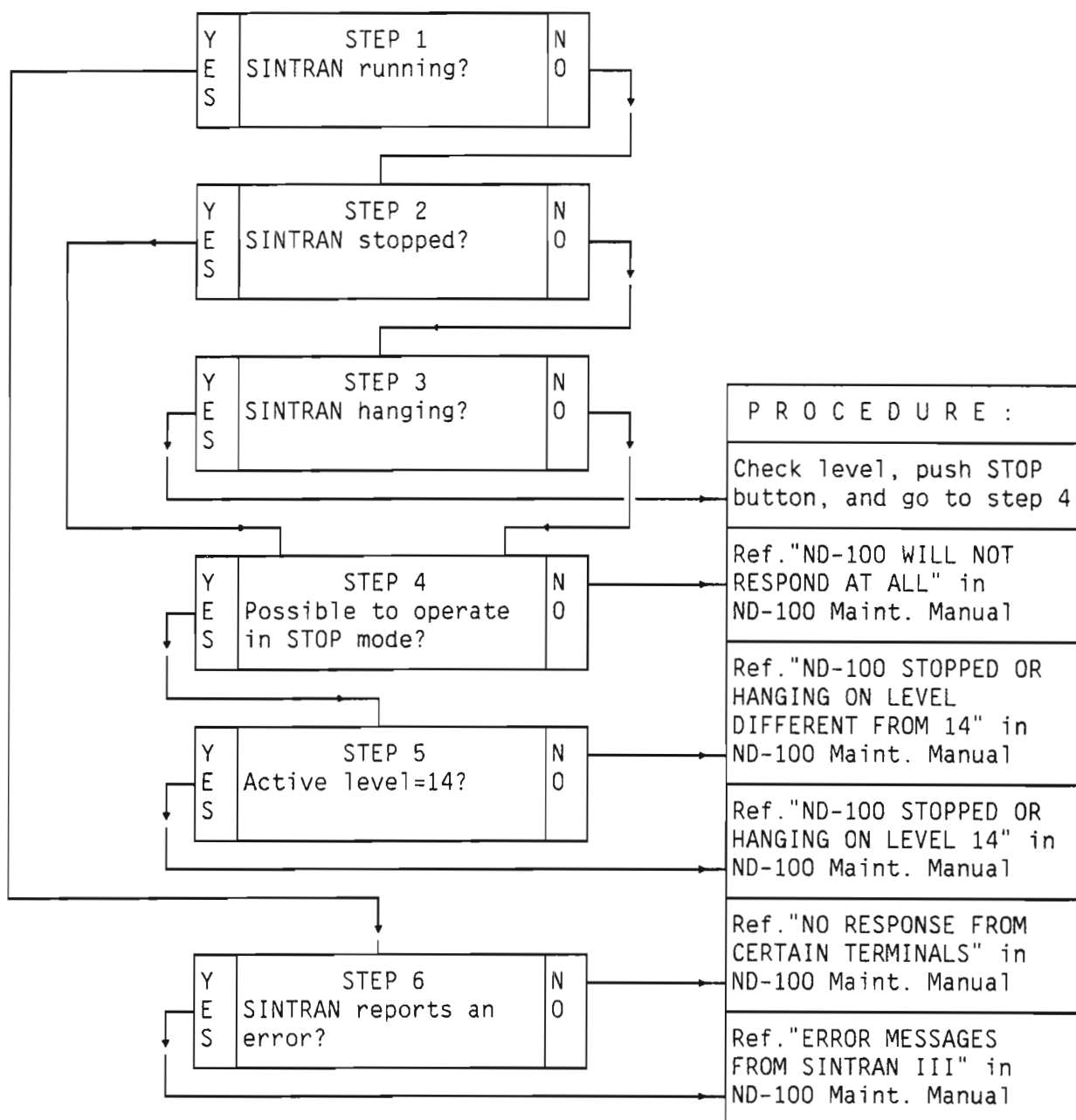
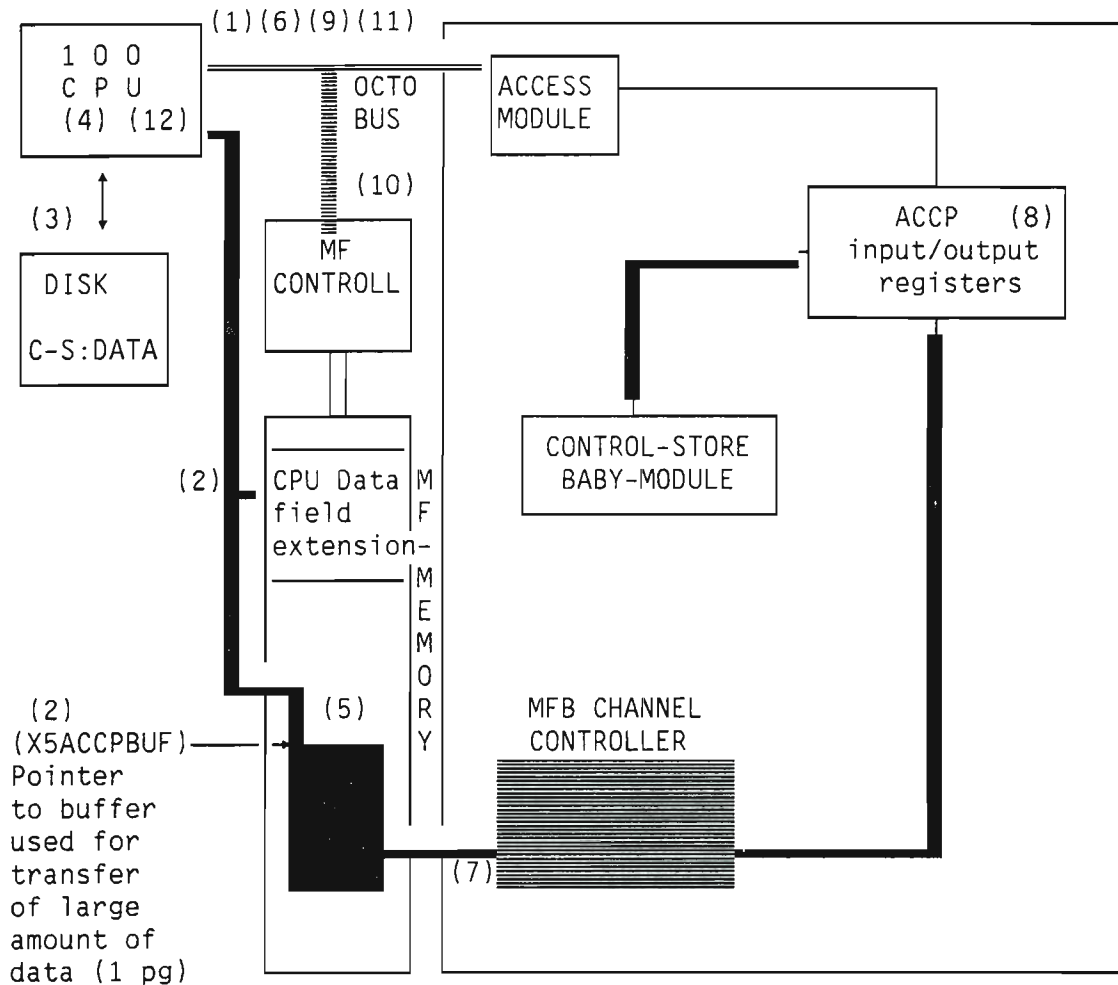


Figure 17. ND-100 Error Conditions

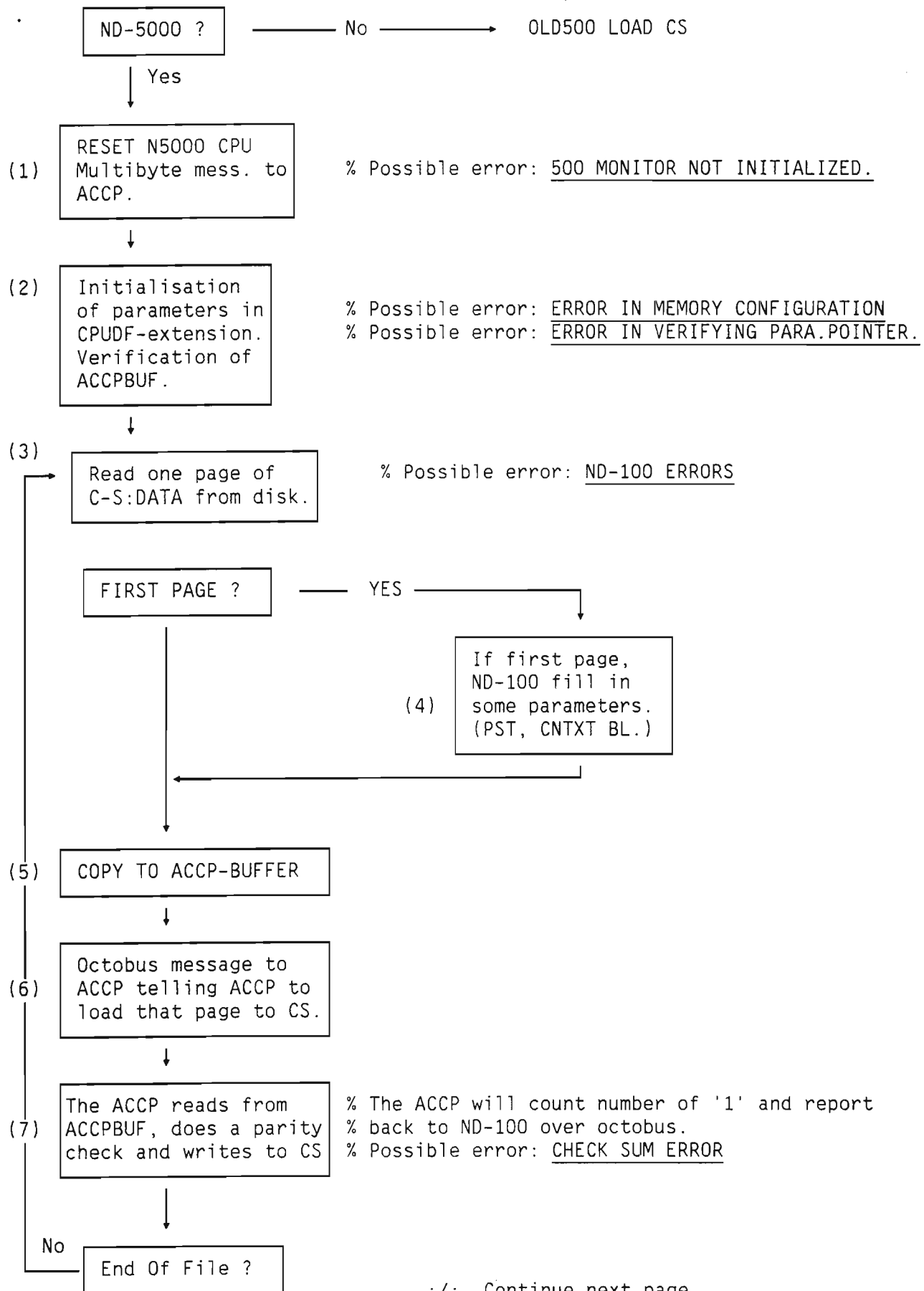
6.2 ERRORS DURING STARTUP

This section explains what what happens during startup of the ND-5000 and about possible errors that might occure.

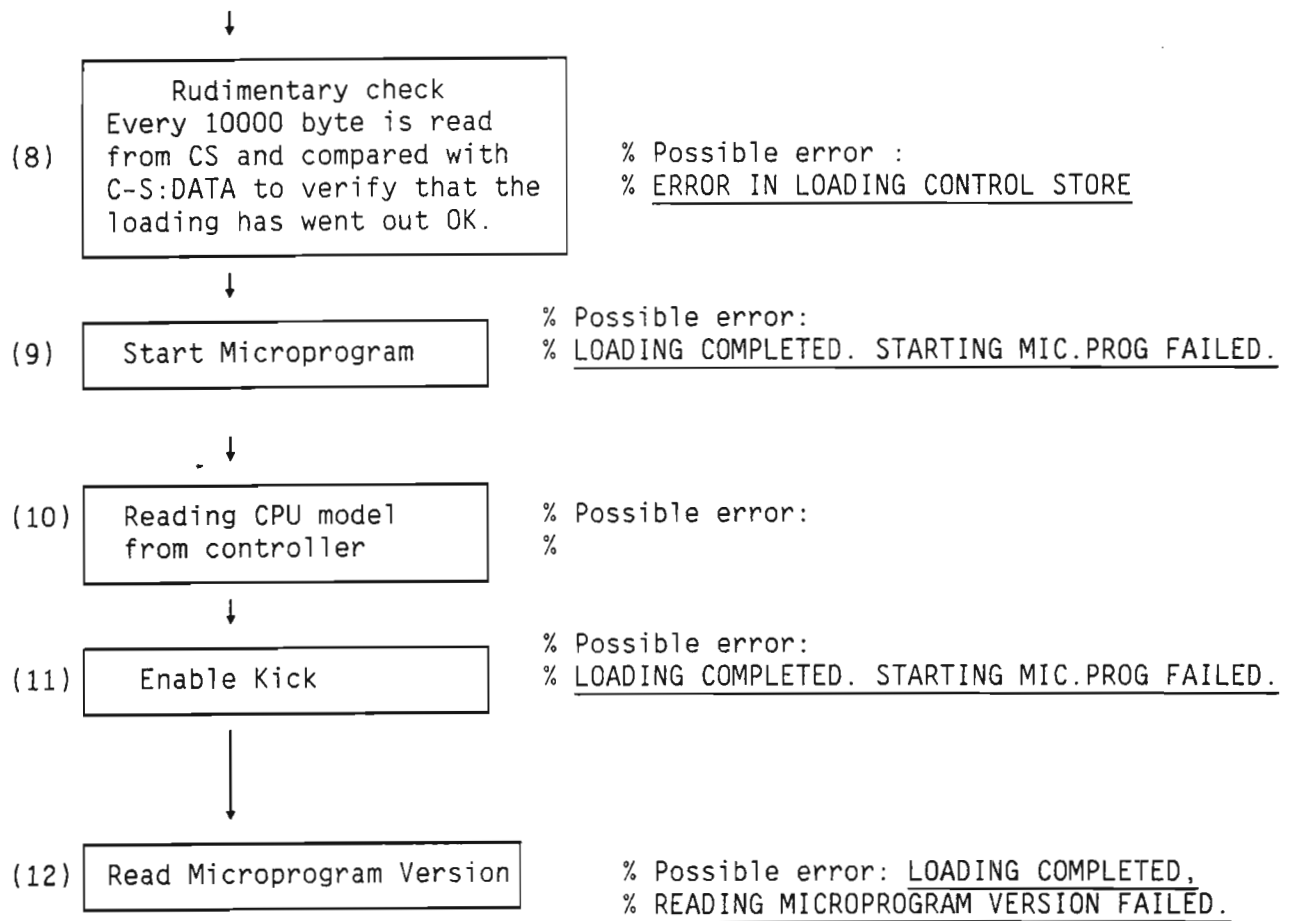
6.2.1 LOADING OF CONTROL STORE



Numbers in parenthesis refers to the different states in the flow-chart diagram on the next two pages.

Load-control-store flow:

:/: Continue next page



6.2.2 DIFFERENT ERROR MESSAGES THAT CAN OCCURE DURING START-UP OF ND-5000

ATTEMPT TO START THE ND-500-MONITOR DON'T SUCCEED

ND-100 HANG ON LEVEL 13 AND 14 WHEN ENTERING N5000 MONITOR FIRST TIME

Explanation When trying to start the ND-500-monitor, the ND-100 hang on level 13 and 14. The reason for this could be errors in accessing MF-Line-Driver (100 BUS error) or errors on the MF-Line-Driver itself.

Action Run the 100 test-program CONFIG.

ND-500-MONITOR NOT INITIALIZED

The reason for this error could be OCTOBUS errors or error on the ACCP-module in the 5000-CPU. Wrong parameters in the MF-BUS configuration could also be the reason.

Explanation During SINTRAN start-up procedure it will find out whether there is an ND-5000 or ND-500 present in the system by sending IOX 's to the different interfaces via OCTOBUS. If one or more N5000's are found, MASTER-CLEAR on N5000 and reset ACCP is performed on the identified CPU's. This will initiate a short N5000 CPU selftest. The lowest byte in CPU df variable CPUAVAILABLE is updated according to configuration. If no CPU is found, the user will be denied access to ND-500-MONITOR and an error message will be given.

Action Possible error could be on octobus. Connect a terminal to ACCP consol and MF-consol. Verify the octobus communication by the MF-consol command

>LIST-OCTOBUS-STATION.

Check that the station numbers are correct for the different slot positions.

If errors, check the octobus switches on MF line-driver and the MF-Bus Controller (Double Bus Controller). (The octobus speed should be "0" for COMSON and MAXON. "2" for beta-cabinet). Do also check any loose connections in the octobus wiring.

Check the ACCP function by running selftest from the ACCP consol.

If octobus tests still fails, exchange the failing module.

Check the MF-BUS configuration and parameters.

ND-5000 SELFTEST IS FAILING

```
BUS test failed
Result:  0000FFFFH
Expected: 12345678H
Continue this test? [Y/N]:
```

NOTE

If the selftest fails in the Instruction or Data CACHE test, check that both the Program and Data CACHE enable switches are set in the ON position. Perform a software RESET by pressing <CTRL X>.

The selftest can be started by the following ACCP commands:

ACCP:RUN-SHORT-SELFTEST↵

A short version of the selftest is started (approx. 20 sec).

ACCP:RUN-LONG-SELFTEST↵

or from ND-500/5000 Monitor by the command:

ND-5000:RUN-SELFTEST↵

Start the full selftest of the ND-5000 CPU. This test takes appr. 3 minutes, and in addition to the Master Clear selftest it includes a thorough test of control store and cache.

ND-5000:MASTER-CLEAR↵

Perform a master clear message via the Octobus to ND-5000 CPU and a short selftest is started.

NOTE

When the selftest is finished the control store has to be reloaded. The selftest will destroy the control store.

If the selftest fails, the red indicator on the edge of the motherboard starts flashing, and the following message appears on the error device:

```
23/10-13:04 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-42
ND-5000 selftest failed in:
5B...-ALU test
```

This means that the the ALU test failed.

When the user enters the ND-500/5000 Monitor he will get a message that selftest has failed.

```
ND-500/5000 MONITOR Version I01 87. 9. 1 / 87. 9.17
ND-5000 selftest failed in:
.5B...- ALU test
N5000:
```

The following command can be used to reset the the ND-5000 CPU:

ND-5000:RESET-CPU↵

This will perform a reset CPU message via the Octobus to the ND-5000. The ACCP will be reinitiated. This command will reset the ND-5000 CPU if the selftest has failed, and the red indicator on the edge of the motherboard.

NOTE

If the selftest has failed and after RESET-CPU has been performed, a new selftest can be started either by the command MASTER-CLEAR or RUN-SELFTEST. If the selftest fails again the ND-5000 CPU has to be exchanged.

In addition:

```
ND-500/5000 MONITOR Version I01 87. 9. 1 / 87. 9.17
ND-5000 selftest failed in:
17B...-
N5000:
```

Explanation

After the execution of the selftest, the ACCP sends a multibyte message to the controller asking for the CPU type. Any mismatch between the CPU model returned and the CPU-type (possible types are CPU I, CPU II and CPU III) one results in an error message mentioned above. If the ND-5000 CPU don't receive any CPU model from the controller, caused by octobus communication problems, this will be reported to the ACCP console.

Action

Connect a terminal as ACCP console. This may give more information about the reason for the trouble.

Possible error messages from ACCP console could be:

```
MF Bus controller not found on octobus station 2-7.
```

Action

Check OCTOBUS communication.

```
MF Bus controller has incorrect CPU model setting.
```

Action

-Check ECO level on the controller and the ND-5000 ACCP module

-Set correct CPU model.

ERRORS DURING LOADING OF CONTROL-STORE

The next step of initialization takes place when the first MON 60 call is executed. A routine CON50MD is called to connect octobus OMD number. This will make it possible for N100 to receive multibyte messages from ACCP. If this is OK, the CPUAVAILABLE bit 5ALIVE is set. If not OK, octobus messages will later give appropriate error messages.

The buffers for N5000-N100 communication will be allocated in shared (MF) memory, e.g. mon60 buffers, octobus buffer, message buffers, fast-UDMA buffers etc.

Normally the first MON 60 function will be RESERVE-PROCESS(15B). If this is the first access to system monitor, several local tables are initialized (e.g. swap-file-table, segment table). The result of the ND-5000 selftest will be checked, and octobus OMD and ident number will be sent to ACCP.

If the next function is place, start-standard-domain or alike, the necessary action to make the system run will be performed automatically. This sequence will contain the following :

Load control-store (func 37) including a reset on ND-5000 CPU and a verification of memory addresses in communication with ACCP (ACCP function: Verify Parameter Pointer). A 2Kb buffer in the MF- memory is used for transferring. The pointer to this page (X5ACCPBUF) is found in the negative part of the dummy message (-17 and -16). This buffer is verified in the following way:

The ND-100 writes a bit pattern to the transmission buffer which is read by the ACCP and sent over to ND-100 again over OCTOBUS. Any mismatch here will be reported as:

ERROR 2174	ERROR IN VERIFYING PARAMETER POINTER
------------	--------------------------------------

Action

Check the memory configuration.

Next step in the Loading Control Store procedure is to load the first page of CONTROL-STORE:DATA into the transmission buffer. Some system parameters are patched into this first page. Then this page is read by the ACCP and loaded into the ND-5000 Control Store memory. Every 1000 byte comparison is performed on the CONTROL-STORE:DATA file and Control Store in the ND-5000. Errors here gives:

ERROR 2154	CHECK SUM ERROR
------------	-----------------

Action

Run the long self test from the ACCP consol by the command: RUN-LONG-SELFTEST If failing, replace the 5000 CPU.

After the Control Store has been loaded, the ND-100 reads some loacations from the Control Store module to verify that CONTROL-STORE:DATA has been loaded correctly. The content of these bytes is checked against the content of the CONTROL-STORE:DATA file. Mismatch here gives:

ERROR IN LOADING CONTROL STORE

Explanation

The Control store memory is not loaded correctly.

Action

Replace the 5000 CPU.

Then the microprogram is started. The microprogram asks: "WHO AM I?"

If wrong CPU model set from the MF-consol and the correct microprogram is loaded.

- CPU I will run as a slow version of 5200.
- CPU II will run as 5400.
- CPU III will run as 5400

Action

If the correct microprogram version is used:

- Set the correct CPU model by using the updating tool.
- Perform the command: MASTER-CLEAR.
- Reload Control Store.

If correct CPU model is set on the controller:

- Copy the correct microprogram version to CONTROL-STORE:DATA.
- Reload Control Store.

Next step in the start-up procedure is to enable for kicks on octobus. Then the ND-500-Monitor reads the microprogram version.

Possible error message:

LOADING COMPLETED. READ MICROPROGRAM VERSION FAILED.

Explanation The ND-100 is not able to read the microprogram version. The reason for this could be wrong micro program version, HW fault on 5000 module or error in communication between 100 - 5000.

Action Check and reload the microprogram. If still failing, replace the 5000 CPU.

```
LOADING COMPLETED.  
FATAL SYSTEM ERROR.  
N500/5000 TIMEOUT.
```

Explanation If this error occurs during loading of control store, the error could be caused by errors in CONTROL-STORE:DATA.

The timeout error is explained under "N500/5000 TIMEOUT" in this troubleshooting chapter.

Action Priority:

- Copy a new microprogram version to CONTROL-STORE:DATA on disk.
- Exchange 5000 CPU.
- Run a memory test from ND-100.
- Test communication from ND5000 to memory.
- Run memory-test from MF console.

If the error is still not solved, look at the explanations to the message "N500/5000 TIMEOUT" later in this troubleshooting chapter.

The next step in the start-up procedure is to load and start swapper. Errors that could occur here are explained in other places in this chapter, see page 84.

6.2.3 ND-500 ERROR MESSAGES

GENERAL

This section gives an overview of the different error messages from the ND-5000 computer.

You will also find an explanation of each error message and advice for trouble-shooting procedures.

Error messages:	See page:
*** FATAL SYSTEM ERROR ***	--
*** ND5000 HARDWARE FAULT ***	--
PAGE FAULT	--
PROTECT VIOLATION	--
INDEX SCALING ERROR	--
ILLEGAL INSTRUCTION CODE	--
INSTRUCTION SEQUENCE ERROR	--
ILLEGAL OPERAND SPECIFIER	--
TRAP HANDLER MISSING	--

Table 9. Survey of ND-5000 error messages

FATAL SYSTEM ERROR MESSAGES

A group of errors are fatal to the system. These errors are detected by the N500 driver, System Monitor or by the Swapper. All processes running in ND-5000 will be aborted.

The error message always the heading (1. line):

***** FATAL SYSTEM ERROR *****

Second line of the error message
ND-500/5000 time-out Time-out, impossible to terminate ND-500/5000 Fatal error from System Monitor The Swapper Stopped Fatal error from Swapper

ND-500/5000 TIME OUT

```

*** FATAL SYSTEM ERROR ***
ND-500/5000 time-out

N100 STATUS  000000
N500 STATUS  000000
MAR 000000000000    MICRO P: 00000177777

```

An Error message will in addition to the user error message be written on the Error device:

```

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
Error code:  2000B  ND-500/5000 time-out

```

Explanation

Two different situations may be origin to this error message:

- The watchdog message (Read microprogram version) has not been processed by the microprogram in ND-5000 CPU or this is not indicated in the status of the message.
- ND-100 is unable to reserve the execution queue semaphore. This will normally indicate that the microprogram has not released it in time, which is due to ND5000 hardware problems.

Action

1. If this problem occurs during normal runtime on the ND-5000.

- Find the extended datafield (Ref: Appendix):

Look at location x5sema.

If x5sema = 0: Look at the location x5proc. If x5proc = a process number then it means that neither ND-5000 microprogram or ND-100 has reserved the execution queue and a process was executing macro code. Use the command LOOK-AT-HARDWARE to find the program counter and the process number.

N5000: LOOK-AT-HARDWARE ↵

A list of registers is displayed. Look for SRF17 and the P register. The content of SRF17 - 1 was the current process. The P register points to the instruction.

Restart ND-5000 and place the failing process and look at the failing instruction using the P register.

If x5proc = -1: The ND-5000 microprogram is in the idle loop or have just left the idle loop to scan the execution queue, but not reserved it yet. In this case the ND-5000 CPU don't

response to any kicks (external traps).

Possible error:

- ND-5000 CPU or microprogram.
- MF memory

If x5sema = -1: Look at the location xprocNo. If xprocNo = 0: ND-100 has reserved the execution queue. If this is the case may be something wrong with:

- System monitor or the ND500 driver.
- ND-100 CPU or memory.

If xprocNo = 1 to 4: The ND-5000 microprogram has reserved the execution queue. If this is the case may be something wrong with:

- ND-5000 CPU.
- System monitor or the ND500 driver.
- MF memory.

- Take a Memtof dump of the situation.

2. If this problem occurs during start up of ND-5000.

- See chapter about error when start up

TIME OUT, IMPOSSIBLE TO TERMINATE ND-500/5000

***** FATAL SYSTEM ERROR *******Time out, impossible to terminate ND-500/5000**

An Error message will in addition to the user error message be written on the Error device:

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM) TERMINAL-52
Time out, impossible to terminate ND-500/5000

Explanation

This error occurs when ND-100 sends a terminate message to ND-5000 CPU, and the ND-5000 doesn't respond within certain time limit (at present approx. 0.6 sec.).

Action

Same as for Time out message.

FATAL ERROR FROM SYSTEM MONITOR

*** FATAL SYSTEM ERROR ***

Fatal error from System Monitor Errcode: 30B

An Error message will in addition to the user error message be written on the Error device:

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
Error code: 2014B Fatal error from System Monitor

The complete error codes is in range from 2300B to 2347B. Only the displacement above 2300B will be regarded as subcodes to Fatal error from System Monitor. Ref. appendix Fatal error from System Monitor.

Calculate the complete error code: 2300B + 30B = 2330B

Explanation

If this error occurs during startup of the ND-5000, it is because the system monitor detects that something is wrong when initializing the Swapper (load Swapper or starting the Swapper).

However, if this error occurs during processing, the system monitor has found an error in the system tables.

Possible errors could be:

- The system monitor has been corrupted.
- Error in memory configuration.
- Error in memory switch settings.
- Other serious MF memory errors.
- Octobus errors.

Action

- Restart ND-5000 by a warm start or a cold start.
- Check that the SWAPPER:PSEG and SWAPPER:DSEG files are ok.
- Check that the ND-5000 selftest runs ok.
- Exchange the ND-5000 CPU and see if the problem is solved.
- Run test programs on ND-100.
Disc-tema, Memory, Instruction, Cache, Paging and Octobus test.

THE SWAPPER STOPPED

```
*** FATAL SYSTEM ERROR ***
The Swapper stopped
PAGE FAULT
At program address:      1      2242B
Logical address:         1    1024460B
Physical segment:                58D
MEMORY MANAGEMENT STATUS: 22701016000B
```

An Error message will in addition to the user error message be written on the Error device:

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: [SYSTEM] TERMINAL-52
Error code: 2070B The Swapper stopped
```

Explanation

The ND-500 driver detects a fatal trap in the message buffer. If the current process is the Swapper process (process No. 0), the trap is fatal for the system. The swapper is stopped and the error message will be written.

The reason for the stop is given in the third line of the error message and can be one of the error situations listed below:

- PAGE FAULT (As in our example)
- PROTECT VIOLATION
- HARDWARE FAULT
- ILLEGAL INSTRUCTION CODE
- ILLEGAL OPERAND SPECIFIER
- INSTRUCTION SEQUENCE ERROR
- INDEX SCALING ERROR

Action

The action depends on the error message:

PAGE FAULT:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Proceed with step 2 to find the instruction that caused the error. Also see the **PAGE FAULT** error message.

PROTECT VIOLATION:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Proceed with step 2.

Also see the **PROTECT VIOLATION** error message.

HARDWARE FAULT:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Proceed with step 2.

Also see the **HARDWARE FAULT** error message.

ILLEGAL INSTRUCTION CODE.

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Go to step 2 to find the instruction that caused the error.

Also see the **ILLEGAL INSTRUCTION CODE** error message.

ILLEGAL OPERAND SPECIFIER:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Go to step 2 to find the instruction that caused the error.

Also see the **ILLEGAL OPERAND SPECIFIER CODE** error message.

INSTRUCTION SEQUENCE ERROR:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- Go to step 2 to find the instruction that caused the error.

Also see the **INSTRUCTION SEQUENCE** error message.

INDEX SCALING ERROR:

- Check that the internal hardware tracer has triggered. If so, clear the FERROR flag and dump trace memory to a file.
- See the **INDEX SCALING** error message.

STEP 2

If you want to have a look at the instruction causing this error, you must first clear the FERROR flag. This flag is always set when FATAL SYSTEM ERRORS occur.

The flag is cleared like this:

```
@LOOK-AT S3DPIT
READY
4227/ nnnnnnn
nnnnnn-12/ yyyyyy
yyyyyy+13/ 2070 0
```

Now you can use the LOOK-AT-REGISTER command on the swapper process from the ND-500-Monitor. Log in as user SYSTEM and do as described below:

```
N500: ATTACH-PROCESS 0
N500: LOOK-AT-REGISTER P
P : xxxxxxxxxxxx/
P1 xxxxxxxxxxxx: <Failing instruction>
N500: EXIT
```

FATAL ERROR FROM SWAPPER

```
*** FATAL SYSTEM ERROR ***
Fatal Error from Swapper
Error Code : 31B
```

An Error message will in addition to the user error message be written on the Error device:

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
Error code: 2047B Fatal error from Swapper
```

Explanation

This fatal error is detected by the Swapper and reported to the driver. All processes running will be aborted.

The error could be caused by almost anything.

- SINTRAN III error
- System Monitor error
- File system error
- ND-5000 hardware error
- Internal error in the swapper
- ND-100 hardware error

A list of sub error-codes is found in the appendix.

Action

- Record all ND-5000 activity that was active when this error occurred:

N5000: LIST-ACTIVE-SEGMENT -2 ↵

All active segments for each process are printed out, even the swapper process.

- Clear the FERROR flag by doing the following:

@LOOK-AT S3DPIT↵

READY

4227/ nnnnnnn

nnnnnn-12/ yyyyyy

yyyyyy+13/ 2070 0↵

- Take a dump of the Swapper data segment by using the command DUMP-SWAPPER in the ND-500/5000 Monitor. Ref chapter "Debugging commands in the ND-5000 monitor".
- Run test on the filesystem (Filesystem investigator).
- Run test on ND100 CPU and memory.

- Exchange the ND-5000 CPU and start up the system to see if the problem is solved by exchanging the CPU.
- If not, the dump should be investigated.

HARDWARE FAULT

If a Hardware Fault trap condition is reported back to the ND-500/5000 Monitor the following error message will be written:

```
*** ND5000 HARDWARE FAULT ***

At program address:      1      31B
From CPU in slot position:      6D
Logical address:      1      466414B
MEMORY MANAGEMENT STATUS:      5B
DATA POFF read request
Physical address:      13  137776414B
Physical segment:      8D
WR:      13771B
ACCP status:      62750B
BADAP:      140B
```

An Error message will in addition to the user error message be written on the Error device:

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 51B at: 1000000031B ND-500/5000 Hardware
fault
```

An error message will also be written on the console connected to the MF-bus controller:

```
* MFBUS BUSERROR (MEMORYCYCLE) - T I M E O U T - *
MAINT.STAT: 155022B
ERRLOG 1: 3000B ERRLOG 2: 20046B ERRLOG 3: 44001B ERRLOG 4:0377B
MASTER: 0006 SLAVE: 00000 ADDRESS: 37400B SYNDROM: 0000B
```

Explanation

A Hardware Fault trap conditon occures in following situations:

- Memory error
- Index error in the Physical Segment Table

Memory error is detected by the BADAP gatearray on the ND-5000 CPU and the ACCP will be interrupted to start the microprogram in the hardware fault routine.

A memory error could be one of two conditons:

- Memory timeout
- Memory parity error

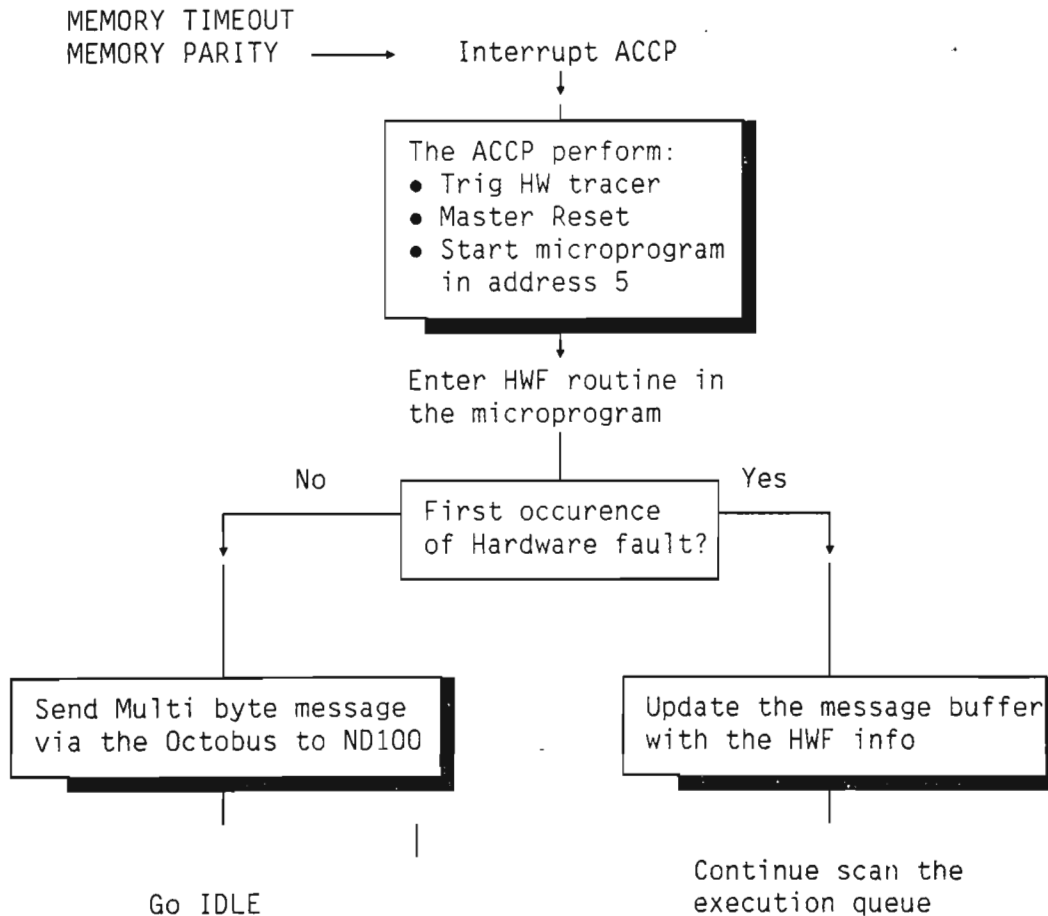
When a Hardware Fault is caused by a memory error the program address, ACCP, MMS and BADAP status register is the most interesting part of the error message. In case of a memory error during look-up in the MMS tables the WR register is holding the physical page address used. The MMS status register will also have been locked. However, if a memory error occure during the final read/write access the MMS status will not have been locked, neither the Logical address or the WR register. From these registers the following information can be found:

Program address:	What kind of instruction
ACCP status:	Data/Program reference.
BADAP status:	Memory timeout/parity error.
MMS status:	Memory timeout/parity error during look up in MMS tables.
WR register:	Failing physical page address during look up in MMS tables.

The slot number in the error message indicates in which slot position in the MF crate the trapping ND-5000 CPU is located.

In addition to the Hardware Fault error message, an errormessage will also appear on the console connected to the MFB controller. The MFB controller will when corrections is needed or if any memory error occures in the MF bus bank, report this to the error station number, usually N100 via the Octobus (MFB Contr. PROMS ver. C or later). This multibyte message will then be received by N100 and sent to the Error logger (Sintran K WM500 or later versions). The Error logger will then write an error message on the error device.

The sequence of handling a memory error may be shown like this:



In case that a multi byte message is sent, the error message that will be written on the error device and have the following layout:

```

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
Memory error detected by the ND-5000 Micro program
Process no.: 1B
Trapping P.: 10000000025B
Restart P.: 10000000025B
Trap no....: 51B
MMS-sts....: 00005516075B
Log.addr...: 10000000160B
Phys.addr...: 17777774160B
Phys.seg...: 11B
Wr.....: 0B
ASTS.....: 62750B
Badap.....: 140B
  
```

The error message appearing on the user terminal is as follows:

Memory error detected by the ND-5000 Micro program

Hardware fault situations detected by the MMS gatearray:

- Index error in the Physical Segment Table

When this situation occurs the MMS status register and logical address is the interesting part of the error message.

ActionIn case of Memory Parity error:

- Bad memory board in the MF bus.

Look at the error message appeared on the console connected to the MFB controller if any or check for any red indicators lighting on any of the memory boards in the MF bus. From the error message appeared on the console the slot number of the slave module indicates where in the MF crate the "bad" memory board is located. If any red indicators is lighting on this memory board then change this board.

- Data transfer error on the MF bus or the detection circuit on the ND5000 is bad.

In case of transfer error on the MF bus it's likely that there is other problems occurring in this MF bus as well.

However, the detection circuit on ND5000 CPU may have failed. If so, change the ND5000 CPU.

In case of Memory timeout:

- Too big physical address in the MF bus supplied by the ND5000 CPU.

This could be a hardware as well as a software problem. The link addresses in the message queue could be wrongly updated by the System monitor in Sintran. This link address to the next message is used by the microprogram to find the next message in the queue. The microprogram save this address in the SRF. If this is a hardware problem then it will occur very often.

The Swapper is also using physical addressing when accessing the Physical Segment Table.

The hardware tracer will contain the physical address and whether the microprogram was scanning the message buffer or if the Swapper was active.

In case of TSB miss the Memory Management gatearray looks up into the Physical Segment Table to find the correct page. During this lookup, physical addresses is used based on the physical page addresses found in the

PST. One of this page addresses may have been wrongly updated by the Swapper or by the System monitor. If so, the physical page address read from the PST and used in MMS can be found in WR register in the error message.

If WR contain a physical page address within the memory size in the MF bus it's likely a hardware error on the ND-5000 CPU.

If WR contain a physical page address outside the memory size in the MF bus, a hardware problem in the ND-5000 CPU may have been the reason for that the physical page address in the PST has been wrongly updated by the Swapper process earlier.

- Bad memory board in the MF bus.

By looking at the error message from the MFB controller the slave slot number will point out the memory module in the MF crate.

NOTE

The internal hardware tracer will in case of Hardware fault have been triggered by master reset from the ACCP.
The trace memory must be dumped from the ND-500/5000 Monitor from user system.
The trace module contain valid information about the error situation.
Ref. chapter ND-5000 Trace Module.

Example

```

@ND
ND-500/5000 MONITOR Version I REV.-I01
N5000: BM2

*** ND5000 HARDWARE FAULT ***

At program address:      1          31B
From CPU in slot position:      6D
Logical address:      1          466414B
MEMORY MANAGEMENT STATUS:      5B
DATA POFF read request
Physical address:      13  137776414B
Physical segment:      8D
WR:      13771B
ACCP status:      62750
BADAP:      140

```

N5000:

In this case N500-MESSAGE can be used to decode this message further if necessary. If so, start N500-MESSAGE on another terminal from user SYSTEM.

@ND500-MESSAGE

```

*****
***  N D 5 0 0 X - MESSAGE DECODER  pre.21.08.87***
*****

```

Status on CPU type..: 5800 CPU number..: 16408

```

Operating syst.: SINTRAN III VSX/500 - K
Revision.....: 144205
Local CPU.....: ND110/CX-16PITS - 32 Fp
Mic.program ver: 7D
Main CPU.....: ND5000
Mic.program ver: 14213D
System part....: 87. 5.29 Rev. K04
Swapper.....: 87.07.03
Local memory...: 5632D Kbytes.
Shared memory..: 16384D Kbytes.
Register block.: 00000444000B ==> phys.ND5000 addr.
Phys.Seg.Table.: 00000644000B ==> phys.ND5000 addr.

```

>>READ

Give process number.(-1=SW):1

Dump of Message buffer for process: 1 CPU in slot position: 6D

```

Link.....: 177777B
Link.....: 177777B
Status....: 000003B          ==> Answer/N500 finished
Sender....: 000001B          ==> Process no.
Receiver...: 000001B          ==> Process no.
Prev.link.: 000001B          ==> Previous message
Micfunc....: 000025B          ==> Restart after trap

```

** TRAP MESSAGE.....: Hardware fault on Data channel

Trapping P.....: 01000000031B
Restart P.....: 01000000031B
Trap number.....: 051B ==> 41D
Logical Address...: 01000466414B
MMS Status reg...: 00000000005B
 No trap indicated in MMSTS!!
 POFF read request
Physical Address.: 13137776414B >>13884 Kbyte
 Phys.page: 15436B
Phys.Segment no...: 000010B
Physical page-WR.: 013771B
ACCP status(ASTS): 062750B
BADAP status.....: 140B ==> Memory timeout
General Buffer pointer...: 054 032000B

PAGE FAULT

If a Page Fault trap condition is reported back to the ND-500-MONITOR the following error message will be written:

```
PAGE FAULT
At program address:      1      2242B
Logical address:         1    1024460B
Physical segment:                58D
MEMORY MANAGEMENT STATUS: 22701016000B
```

Explanation; When a PAGE FAULT is issued by a process, this process is set to idle, and the swapper is started to handle the page fault.

The swapper checks whether the logical page number (bit 26-11 from the logical address) is greater than or equal to the number of pages in the segment.

If that is the case then it depends if the process that issued the Page Fault trap has a local trap handler to take care of a Programmed trap condition or not when further action is going to be taken.

If the a local trap handler for Programmed trap condition exist and is locally enabled, the Swapper will set the Programmed trap bit in the ST1 register for trapping process. This will cause a Programmed trap condition when the process is restarted.

The error code will in this case be written into the context block for the process and can be read by the process by using the monitor call GERRCOD.

The message appearing on the user terminal will usually have the following layout:

```
PROGRAMMED TRAP
```

The process will continue after handling this trap condition.

If the process is not locally enabled for Programmed trap, the error message will be issued by the ND500 Monitor. The process will not be restarted and the error message below will appear on the user terminal.

ADDRESS OUTSIDE DATA SEGMENT

PAGE FAULT

At program address: 1 2242B
 Logical address: 1 1024460B
 Physical segment: 58D
MEMORY MANAGEMENT STATUS: 22701016000B
CPU trap: *)Zero in PST

*)This message depends on trap-type.

or

ADDRESS OUTSIDE PROGRAM SEGMENT

PAGE FAULT

At program address: 1 2257B
 Logical address: 1 2003531B
 Physical segment: 43D
MEMORY MANAGEMENT STATUS: 22700006100B
CPU trap: *)Zero in PST

*)This message depends on trap-type.

The following can be read from the error message above:

A Page Fault trap condition has occurred at program address 1'2257B from ND5000 CPU when reference to logical program address 1'2003531B was attempted. The logical page number (2003531B / 4000B) 400B was outside the program segment. The physical segment number used for accessing the program segment was 43D.

When examine the program at the failing address this can be done as follows:

```
ND-5000: LOOK-AT-PROGRAM 1'2257↵
P 1      2257B: CALL 1'3531B,0B ↵
P 1      2265B: IF K RET EX↵
```

Now we can see that the correct logical program address should be 1'3531B. This is an access to the first logical page on the program segment.

In this case it is an error on ND-5000 CPU.

Action

1. If the trap has been reported to the monitor as example above then
 - Dump of the internal hardware trace module to a file may only be done if the error message is a fatal system error and the Swapper stopped, because of a page fault. In other cases, the internal tracer does not contain any valid information.
 - Write down or get a hardcopy of the error message.
 - Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in procedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to connect to the failing process. Then the following procedure can be used:

```

N500: LOOK-AT-PROGRAM <program address-200B>~
Px xxxxxxxxxB: <instruction> ~
.
.
.
Px xxxxxxxxxB: <instruction causing the trap>~
Px xxxxxxxxxB: <instruction> EXIT~

```

- Then dump the following registers for the failing process:

```

N500: LOOK-AT-REGISTER~
P  : xxxxxxxxxB ~
L  : xxxxxxxxxB ~
R  : xxxxxxxxxB ~
B  : xxxxxxxxxB ~
I1 : xxxxxxxxxB ~
I2 : xxxxxxxxxB ~
I3 : xxxxxxxxxB ~
I4 : xxxxxxxxxB EXIT~

```

- Use the commands LOOK-AT-DATA or LOOK-AT-RELATIV to the B-, R- or Index-register to get the indirect address if any.
- Copy the trace file to a floppy if a dump has been taken.
- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the special designed box together with the floppy containing the trace dump and a hardcopy of the error message and the information described above.

- Run the same program on the new ND-5000 CPU to see if the problem has been solved or not.

2. If the process has handled the trap condition locally in the process a less detailed error description is written.

To investigate further may be rather complicated, but a procedure to follow is described below. The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

- If the error message is PROGRAMMED TRAP, you should rerun the program with programmed trap disabled:

```
N500: DEBUG-PLACE <program>↵
N500: LOCAL-TRAP-DISABLE PROGR-TRAP↵
N500: RUN↵
```

Now, one of the error messages mentioned above should appear, like:

```
ADDRESS OUTSIDE DATA SEGMENT

PAGE FAULT
At program address:      1      2242B
Logical address:        1  1024460B
Physical segment:              58D
MEMORY MANAGEMENT STATUS: 22701016000B
```

- If you want to have a look at the instruction that caused the error, you can do the following:

```
ND-5000: LOOK-AT-PROGRAM 1'2242↵
P 1    2242B: W MOVE 1'24460B,R.30B:S ↵
P 1    2251B: W MOVE B.34B:S,R.34B:S EX↵
```

When we compare the failing logical data address in the error message above with the logical data address in the instruction, one bit has been added in the error message (bit 18).

If the program initializes its own local trap handlers, which is very common for the most of the subsystems like NOTIS, PED, SIBAS etc. then it may be complicated to get more detail about the trap condition.

Usually the local trap handlers are set up in one of the first subroutines in the program.

Take a debug place on the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be reported to the ND-500/5000 Monitor.

NOTE

Product as NOTIS, PED, LED, LINKAGE-LOADER, LINKER and other editors which uses file as segment and need to expand its segment, the process a PROGRAM-TRAP locally enabled.

Example

@ND ND-500/5000 MONITOR Version I REV.-I01

N500: TEST ↵

*** 1987-03-10 08:36:48 ND-500 TRAP: (7635B)

PROGRAMMED TRAP

AT ADDRESS 1000000045B

N500: DEBUG-PLACE TEST ↵

N500: LOCAL-TRAP-DISABLE ALL ↵

N500: RUN ↵

ADDRESS OUTSIDE DATA SEGMENT

PAGE FAULT

At program address: 1 2237B

Logical address: 1 40000074B

Physical segment: 69D

MEMORY MANAGEMENT STATUS: 26707217000B

N500: WHO ↵

1 USED BY SYSTEM

2 USED BY SYSTEM

3 USED BY TEST

==> 4 USED BY TEST

The failing process is process 4.

N500: LOOK-AT-REGISTER P ↵

P : 1000002237B /↵

P 1 2237B W STZ B.030B(W1) REG I1 ↵

I1 : 10000000B B ↵

B : 1000000044B EX ↵

N500:

The effective address in the instruction is:

B +30 +(I1*4)

↓

1'44+30B+(10000000B*4)

↓

1000000044B+30B+40000000B

↓

Eff. address:1040000074B

This logical address is too big. Compare the logical address calculated with the logical address found in the message buffer.

In this example register I1 has been wrongly updated earlier in the program.

NOTE

If the logical address found in the message buffer is not equal to the effective address, the error is probably caused by a hardware fault. You should then suspect the ND-5000 CPU.

PROTECT VIOLATION

If a Protect Violation trap condition is reported back to the ND-500/5000 Monitor the following error message will be written:

```
PROTECT VIOLATION
At program address:      1      2237B
From CPU in slot position:      6D
Logical address:         0      24B
MEMORY MANAGEMENT STATUS: 25007017472B
CPU trap: Zero in capability table (DMM and write PV)
DATA Write request
Physical address:        0 57750024B
Physical segment:              0D
WR:                          540B
```

An Error message will in addition to the user error message be written on the Error device:

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 44B at: 1000002237B ND-500/5000 Protect
Violation
```

Explanation

When trying to access a non-existing logical segment in the domain for a process, a Protect Violation trap condition occurs.

If writing to a logical segment is attempted without having the write permitted bit set in the capability for this logical segment, a Protect Violation trap condition occurs.

If ALT prefix is used in accessing a logical segment on one domain from another is attempted without the parameter access bit set in the capability for this logical segment.

The program address in the error message points to the instruction that caused the error.

NOTE

A Protect Violation trap condition can be provoked by a hardware error on the ND-5000 CPU as well as an error in the program. If this program has run through without any error before it is probably a hardware fault on the ND-5000 CPU.

Usually the PROTECT VIOLATION trap has a local traphandler

routine in the program which will be started when the trap condition occurs. The most user defined traphandlers don't handle this trap condition in different way. The error message displayed is normally not detail enough to be used for debugging purposes.

To be able to debug the error, you must make the program stop at the point where the error occurs. This procedure below may be used for programs written in Fortran.

```
N5000:LOCAL-TRAP-DISABLE PROT-VIOLATION~
N5000: RUN~
```

The local traphandler to take care of the Protect Violation trap condition will not be started. If this trap condition occurs now the trap will be reported to the ND-500/5000 Monitor and the error message above will be written. In addition the microprogram will trig the internal hardware tracer. The tracer will contain valid information about the trap condition.

From the error message below we can read:

A Protect violation has occurred at program address 1'2237B from ND5000 CPU in MF crate position 6. The protect violation trap occurred on a data write access to logical segment number 0 , address 24B (logical address 0'24B). The content of WR register in case of protect violation is the physical page address to the process segment.

```
PROTECT VIOLATION
At program address:      1      2237B
From CPU in slot position:      6D
Logical address:         0      24B
MEMORY MANAGEMENT STATUS: 25007017472B
CPU trap: Zero in capability table (DMM and write PV)
           |
           | from MMSTS bit 3:0 trap code.
DATA  Write request
  |   |
  |   | From MMSTS bit 31:29
  |   | or PROGRAM (MMSTS bit 6)
  |   |
Physical address:      0  57750024B  % Valid only if MMSTS
                                   % bit 11 (TSB miss)=0
Physical segment:      0D
WR:                    540B
```



```

PROTECT VIOLATION
At program address:      1      2237B
From CPU in slot position:      6D
Logical address:         0      24B
MEMORY MANAGEMENT STATUS: 25007017472B
CPU trap: Zero in capability table (DMM and write PV)
DATA Write request
Physical address:        0 57750024B
Physical segment:        0D
WR:                      540B

```

Action

1. If the trap has been reported to the monitor as in the example above:
 - Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
 - Write down or get a hardcopy of the error message.
 - Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in procedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to connect to the failing process. Then the following procedure can be used:

N500: LOOK-AT-PROGRAM <program address-200B>

Px xxxxxxxxxB: <instruction>

.

.

.

Px xxxxxxxxxB: <instruction causing the trap>

Px xxxxxxxxxB: <instruction> EXIT

- Then dump the following registers for the failing process:

N500: LOOK-AT-REGISTER

P : xxxxxxxxxB

L : xxxxxxxxxB

R : xxxxxxxxxB

B : xxxxxxxxxB

I1 : xxxxxxxxxB

I2 : xxxxxxxxxB

I3 : xxxxxxxxxB

I4 : xxxxxxxxxB

EXIT

- Use the commands LOOK-AT-DATA or LOOK-AT-RELATIV to B register to get the indirect address if any.
 - Copy the trace file to a floppy.
 - Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate incase there is a ND-5900 system with more than one ND-5000 CPU.
 - Run the same program on the new ND-5000 CPU to see if the problem has been solved or not.
2. If the process has handled the trap condition locally in the process a less detailed error description is written.

To investigate further may be rather complicated, but a procedure to follow is described below.

The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

```
N5000:DEBUG-PLACE <failing program>~
N5000:LOCAL-TRAP-DISABLE PROT-VIOLATION~
N5000:RUN~
```

When the Protect Violation trap condition occurs it will be reported to the ND-500/5000 Monitor as described above. The internal hardware tracer will also have triggered.

If however, the program initializes its own local traphandlers, which is very common for the most of the subsystems like NOTIS, PED, SIBAS etc. then it may be complicated to get more detail about the trap condition.

Usually the local traphandlers are set up in one of the first subroutines in the program.

Debug place the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be

reported to the ND-500/5000 Monitor and the internal tracer will trig.

The action to be taken is the same as done in point 1.

3. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message is will be written on the error device.
The internal hardware tracer will have been triggered.

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM) TERMINAL-52
 ND-500/5000 trap number: 44B at: 1000002237B ND-500/5000 Protect
 Violation

If further investigation is needed:

The following information can be found from the error message:

Protect violation on the program or data channel?
 MF crate position?
 What is the logical address?
 What is the MMS status?
 What is the program address?

If Protect Violation on the data channel:

- Check MMSTS register in the error message. If the capability is zero then proceed, but if it is Write or Alt Protect Violation then the capability may be set wrongly by Sintran.
- Check what kind of addressing mode is used.
- Get the contents of the involved registers:

B-register
 R-register
 Pre index register
 Post index register

If indirect addressing, get the contents of the location holding the indirect address.

- Compare the failing logical address in the error message with the calculated address in the instruction. There may be some differences in bit 27-31. If there is more than one operand in the instruction, the failing address can be compared with every operand to see which one is failing.

- If there is a mismatch between the calculated address and the logical address in the error message then it is a hardware fault on the ND-5000 CPU. However, if the addresses are equal the location or register used in the address calculation has been wrongly updated before in the program. It could still be a hardware fault on the ND-5000 CPU.

If Protect Violation on the program channel:

Get the failing process and the instruction causing the error.

If the instruction is one of the following:

- CALL
- CALLG
- JUMPG
- GO
- RETx
- IF<cond>GO

the calling or jump address is wrongly calculated. If RETx instruction the return address on the local stack may be wrong.

Example

@ND ND-500/5000 Monitor version I REV.I01
 N5000: TEST ↵

*** 1986-03-10 08:36:48 ND-500 TRAP: (7644B)
 PROTECT VIOLATION
 AT ADDRESS 1000000037B

--- EXCEPTION STATISTICS

OCCURRENCES	EXCNO	EXCEPTION TYPE
1	7644B	*** ND-500 TRAP PROTECT VIOLATION

*** JOB ABORTED ***

N5000: DEBUG-PLACE TEST ↵
 N5000: LOCAL-TRAP-DISABLE PROTECT-VIOLATION ↵
 N5000: RUN ↵

PROTECT VIOLATION

At program address: 1 37B
 From CPU in slot position: 6D
 Logical address: 3 5400B
 MEMORY MANAGEMENT STATUS: 25007017472B
 CPU trap: Zero in capability table (DMM and write PV)
 DATA Write request
 Physical address: 0 57755400B
 Physical segment: 0D
 WR: 540B

N5000: LOOK-AT-PROGRAM 1'37 ↵
 P1 37B: D1 =: IND(B.024B) REG B ↵
 B : 1000000004B EX ↵

Now you can calculate the effective address to the location where the indirect address is stored:

$B+24B=1'4+24=1'30$

Get the indirect address:

N5000: LOOK-AT-DATA 1'30 ↵
 D 1 30B: 1000005400B EX ↵

In this case the indirect address is :

1'5400B

Now you can compare the indirect address with the logical address found in the message buffer:

Logical address: 030000005400B
 Indirect address: 010000005400B

In the logical address in the message buffer bit 28 is set.

The logical address in this case has been wrongly calculated in the ND-5000 CPU.

INDEX SCALING ERROR

If a Index Scaling Error trap condition is reported back to the ND-500/5000 Monitor the following error message will be written:

```

INDEX SCALING ERROR
At program address:      1      1062B
From CPU in slot position:      6D
  
```

An Error message will in addition to the user error message be written on the Error device. This error message will also appear when this process is a RT program or standard domain started up directly from Sintran.

```

20/08-12:35 Nd-500/5000 CPU 1: process 1: [SYSTEM]TERMINAL-52
ND-500/5000 trap number:  40B at: 1000001062B ND-500/5000 Index
Scaling error
  
```

Explanation

In post index addressing mode, one of the index registers contains the post index:

H1:=B.20B(R2)

In this instruction INDEX REGISTER 2 (I2) contains the index.

The scaling of the post index is dependent of the data type of the instruction. In the instruction above the effective address (Ea) is calculated like this:

$$Ea = (B) + disp + (R2) * p$$

p is the scaling factor to be used.

This overview shows the relation between the data type and p:

DATA TYPE:	p:
BI	1/8
BY	1
H	2
W	4
F	4
D	8

The content of the post index register could be positive or negative.

If after scaling with correct scaling factor, the result exceeds 32 bit, an INDEX SCALING ERROR trap is generated.

If this trap condition is not handled by a local traphandler in the process the internal HW tracer will trig.

Action

1. If the trap has been reported to the monitor as in the example above:

- Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
- Write down or get a hardcopy of the error message.
- Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in procedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to connect to the failing process. Then the following procedure can be used:

```
N500:LOOK-AT-PROGRAM <program address-200B>↵
Px xxxxxxxxxB:<instruction> ↵
.
.
.
Px xxxxxxxxxB:<instruction causing the trap>↵
Px xxxxxxxxxB:<instruction> EXIT↵
```

- Then dump the following registers for the failing process:

```
N500: LOOK-AT-REGISTER↵
P : xxxxxxxxxB ↵
L : xxxxxxxxxB ↵
R : xxxxxxxxxB ↵
B : xxxxxxxxxB ↵
I1 : xxxxxxxxxB ↵
I2 : xxxxxxxxxB ↵
I3 : xxxxxxxxxB ↵
I4 : xxxxxxxxxB EXIT↵
```

- Use the commands LOOK-AT-DATA or LOOK-AT-RELATIV to B register to get the indirect address if any.
- Copy the trace file to a floppy.
- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate incase there is

a ND-5900 system with more than one ND-5000 CPU.

- Run the same program on the new ND-5000 CPU to see whether the problem has been solved or not.
2. If the process has handled the trap condition locally in the process, a less detailed error description is written.

To investigate further may be rather complicated, but a procedure to follow is described below.

The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

```
N5000:DEBUG-PLACE <failing program>↵
N5000:LOCAL-TRAP-DIS INDEX-SCALING-ERR↵
N5000:RUN↵
```

When the Index Scaling Error trap condition occurs it will be reported to the ND-500/5000 Monitor as described above. The internal hardware tracer will also have triggered.

If however, the program initializes its own local trap handlers, which is very common for the most of the subsystems like NOTIS, PED, SIBAS etc. then it may be complicated to get more detail about the trap condition.

Usually the local trap handlers are set up in one of the first subroutines in the program.

Debug place the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be reported to the ND-500/5000 Monitor and the internal tracer will trigger.

The action to be taken is the same as done in point 1.

3. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message will be written on the error device. The internal hardware tracer will have been triggered.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 40B at: 1000001062B ND-500/5000 Index
Scaling error
```

Example

The following error message may occur:

```
INDEX SCALING ERROR
At program address:          1   1062B
From CPU in slot position:    6D
```

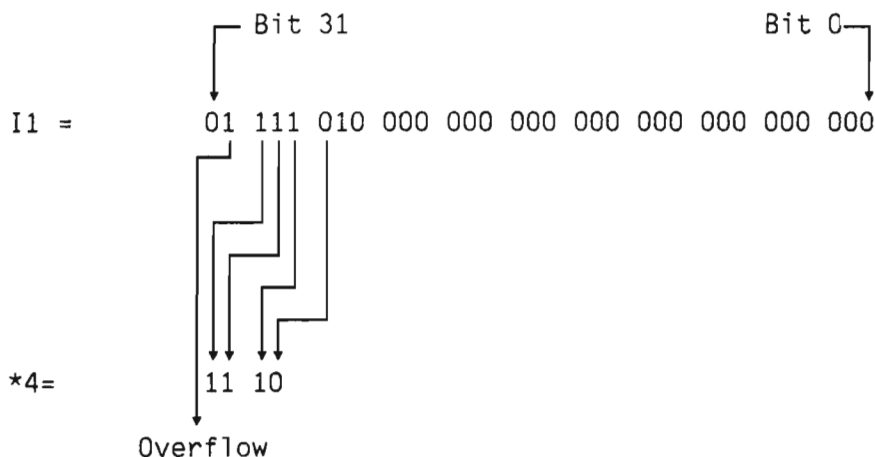
- Use the LOOK-AT-PROGRAM command to get the instruction.

```
N500: LOOK-AT-PROGRAM 1'1062 ↵
P 1   1062B:W MOVE B.034B:S,B.124B(W1) EX↵
```

- Use the LOOK-AT-REGISTER command to find the post index register. If the content of this register is OK, the error condition is caused by a hardware error.

```
N500: LOOK-AT-REGISTER I1 ↵
I1    : 17200000000B EX ↵
```

The data type is word (W), so the contents of I1 must be multiplied with 4. That means shifting two positions to the left.



Exceeding 32 bits in index makes INDEX SCALING ERROR

ILLEGAL INSTRUCTION CODE

If a Illegal Instruction Code trap condition is reported back to the ND-500-MONITOR the following error message will be written:

```
ILLEGAL INSTRUCTION CODE
At program address:      1      1062B
From CPU in slot position: 6D
```

An Error message will in addition to the user error message be written on the Error device. This error message will also appear when this process is a RT program or standard domain started up directly from Sintran.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: {SYSTEM}TERMINAL-52
ND-500/5000 trap number: 41B at: 1000001062B ND-500/5000 Illegal
Instruction Code
```

Explanation

The microprogram will issue an ILLEGAL INSTRUCTION CODE if:

- it is a privileged instruction and the PIA bit in Status register 1 is not set.
- it is a BP instruction and the BPT trap is disabled.
- mapping to a entry which is not defined.

Action

1. If the trap has been reported to the monitor as example above then
 - Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
 - Write down or get a hardcopy of the error message.
 - Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in prosedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to connect to the failing process.

Then the following procedure can be used:

```

N500: LOOK-AT-PROGRAM <program address-200B>↵
Px xxxxxxxxB: <instruction> ↵
.
.
.
Px xxxxxxxxB: <instruction causing the trap>↵
Px xxxxxxxxB: <instruction> EXIT↵

```

- Then dump the following registers for the failing process: fm,n;

```

N500: LOOK-AT-REGISTER↵ P : xxxxxxxxB ↵ L :
xxxxxxxB ↵ R : xxxxxxxxB ↵ B :
xxxxxxxB ↵ I1 : xxxxxxxxB ↵ I2 :
xxxxxxxB ↵ I3 : xxxxxxxxB ↵ I4 :
xxxxxxxB EXIT↵

```

- Copy the trace file to a floppy.
- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate in case there is a ND-5900 system with more than one ND-5000 CPU.
- Run the same program on the new ND-5000 CPU to see if the problem is solved or not.

If the exchange of ND-5000 CPU does not solve the problem then it may be the dataway from the memory to the ND-5000 CPU which is the problem.

The failing module could then be:

- A memory board
- MFB controller
- Double Bus Controller (If ND-5000 Compact)
- Disc corruption

2. If the process has handled the trap condition locally in the process a less detailed error description is written.

To investigate further may be rather complicated, but a procedure to follow is described below.

The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

```
N5000:DEBUG-PLACE <failing program>↵
N5000:LOCAL-TRAP-DISAB ILLEGAL-INSTR-COD↵
N5000:RUN↵
```

When the Illegal Instruction Code trap condition occurs it will be reported to the ND-500/5000 Monitor as described above. The internal hardware tracer will also have triggered.

If however, the program initializes its own local trap handlers, which is very common for the most of the subsystems like NOTIS, PED, SIBAS etc. then it may be complicated to get more detail about the trap condition.

Usually the local trap handlers are set up in one of the first subroutines in the program.

Debug place the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be reported to the ND-500/5000 Monitor and the internal tracer will trigger.

The action to be taken is the same as done in point 1.

3. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message is will be written on the error device.
The internal hardware tracer will have been triggered.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 41B at: 1000001062B ND-500/5000 Illegal
Instruction Code
```

Example

The following error message may occur:

ILLEGAL INSTRUCTION CODE		
At program address:	1	1062B
From CPU in slot position:		6D

Use the LOOK-AT-PROGRAM command to get the instruction.

```
N500: LOOK-AT-PROGRAM 1'1062 ↵  
P 1    1062B:0B EX↵
```

In this case the program address has been wrongly calculated. It is most likely that it is a hardware fault on the ND-5000 CPU.

INSTRUCTION SEQUENCE ERROR

If a Instruction Sequence error trap condition is reported back to the ND-500-MONITOR the following error message will be written:

```
INSTRUCTION SEQUENCE ERROR
At program address:      1      1062B
From CPU in slot position:      6D
```

An Error message will in addition to the user error message be written on the Error device. This error message will also appear when this process is a RT program or standard domain started up directly from Sintran.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 43B at: 1000001062B ND-500/5000
Instruction Sequence Error
```

Explanation

The CALL and ENTx instructions are treated as one instruction. A Instruction Sequence Error trap is issued by the ND-5000 CPU in two cases.

1. If a CALL or CALLG instruction is decoded by the ND-5000 CPU, the next instruction has to be an ENTx instruction else an ISE trap condition ocure.
2. If the ND-5000 CPU decodes an ENTx instruction and the previous instruction was not a CALL or CALLG instruction an ISE trap condition ocure.

If this trap condition is not handled by a local traphandler in the process the internal HW tracer will trig.

Action

1. If the trap has been reported to the monitor as example abowe:
 - Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
 - Write down or get a hardcopy of the error message.
 - Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in.prosedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to

connect to the failing process. Then the following procedure can be used:

N500: LOOK-AT-PROGRAM <program address-200B>↵

Px xxxxxxxxxB: <instruction> ↵

.

.

.

Px xxxxxxxxxB: <instruction causing the trap>↵

Px xxxxxxxxxB: <instruction> EXIT↵

- Then dump the following registers for the failing process:

N500: LOOK-AT-REGISTER↵

P : xxxxxxxxxxB ↵

L : xxxxxxxxxxB ↵

R : xxxxxxxxxxB ↵

B : xxxxxxxxxxB ↵

I1 : xxxxxxxxxxB ↵

I2 : xxxxxxxxxxB ↵

I3 : xxxxxxxxxxB ↵

I4 : xxxxxxxxxxB EXIT↵

- Use the commands LOOK-AT-DATA or LOOK-AT-RELATIV to B register to get the indirect address if any.
 - Copy the trace file to a floppy.
 - Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate incase there is a ND-5900 system with more than one ND-5000 CPU.
 - Run the same program on the new ND-5000 CPU to see whether the problem has been solved or not.
2. If the process has handled the trap condition locally in the process a less detailed error description is written.

To investigate further may be rather complicated, but a procedure to follow is described below.

The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

```
N5000:DEBUG-PLACE <failing program>↵
N5000:LOCAL-TRAP-DISAB INSTR-SEQUENC-ERR↵
N5000:RUN↵
```

When the Instruction Sequence Error trap condition occurs it will be reported to the ND-500/5000 Monitor as described above. The internal hardware tracer will also have triggered.

If however, the program initializes its own local trap handlers, which is very common for the most of the subsystems like NOTIS, PED, LED, etc. then it may be complicated to get more detail about the trap condition.

Usually the local trap handlers are set up in one of the first subroutines in the program.

Debug place the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be reported to the ND-500/5000 Monitor and the internal tracer will trigger.

The action to be taken is the same as done in point 1.

3. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message is will be written on the error device.
The internal hardware tracer will have been triggered.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 43B at: 1000001062B ND-500/5000
Instruction Sequence Error
```

Example

The following error message may occur:

INSTRUCTION SEQUENCE ERROR		
At program address:	1	1062B
From CPU in slot position:		6D

Use the LOOK-AT-PROGRAM command to get the instruction.

```
N500: LOOK-AT-PROGRAM 1'1062 ↵
P 1   1062B:CALL 1'20020B,0B 1'20020/↵
P 1   20020B:W1=:R.20B EX↵
```

↑
Instruction in address 1'20020 should
be an ENTX.

ILLEGAL OPERAND SPECIFIER

If a Illegal Operand Specifier trap condition is reported back to the ND-500-MONITOR the following error message will be written:

```

ILLEGAL OPERAND SPECIFIER
At program address:      1      1062B
From CPU in slot position:      6D
  
```

An Error message will in addition to the user error message be written on the Error device. This error message will also appear when this process is a RT program or standard domain started up directly from Sintran.

```

20/08-12:35 Nd-500/5000 CPU 1: process 1: {SYSTEM}TERMINAL-52
ND-500/5000 trap number:  42B at: 1000001062B ND-500/5000 Illegal
Operand Specifier
  
```

Explanation

An ILLEGAL OPERAND SPECIFIER trap is issued if one of these situations occurs:

- Constant operands as destination.
- ALT prefix on routine argument.
- Type conflict between instr. and operands.
- non-constant number of arguments to CALL and polynomial instructions.
- If register or constant operands in TSET and RDUS instructions.

Action

1. If the trap has been reported to the monitor as example above then
 - Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
 - Write down or get a hardcopy of the error message.
 - Make a dump on a hardcopy of the area around the failing program address. From failing program address - 200B up to the failing program address. A terminal with a hardcopy printer may be used for this purpose. If another terminal have to be used for this purpose, then after log in prosedure and entered the ND-500/5000 Monitor, the ATTACH-PROCESS command can be used to connect to the failing process. Then the following prosedure can be used:

```

N500:LOOK-AT-PROGRAM <program address-200B>↵
Px xxxxxxxxxxB:<instruction> ↵
  
```

```
Px xxxxxxxxxB:<instruction causing the trap>~
Px xxxxxxxxxB:<instruction> EXIT~
```

- Then dump the following registers for the failing process:

```
N500: LOOK-AT-REGISTER~
P   : xxxxxxxxxB |
L   : xxxxxxxxxB |
R   : xxxxxxxxxB |
B   : xxxxxxxxxB |
I1  : xxxxxxxxxB |
I2  : xxxxxxxxxB |
I3  : xxxxxxxxxB |
I4  : xxxxxxxxxB EXIT~
```

- Copy the trace file to a floppy.
- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate incase there is a ND-5900 system with more than one ND-5000 CPU.
- Run the same program on the new ND-5000 CPU to see if the problem is solved or not. If the exchange of ND-5000 CPU does not solve the problem it might be something wrong with the dataway to the ND-5000 CPU.

The failing module could then be:

- A memory board
- MFB controller
- Double Bus Controller (If ND-5000 Compact)
- Disc corruption

2. If the process has handled the trap condition locally in the process a less detailed error description is written.

To investigate further may be rather complicated, but a prosedure to follow is described below.

The main purpose is to provoke the same trap condition to be reported to and decoded by the ND-500/5000 Monitor.

If the failing program is a Fortran program then local handling of the trap condition in the program may be omitted by the following procedure:

```
N5000:DEBUG-PLACE <failing program>↵
N5000:LOCAL-TRAP-DISAB ILLEGAL-OPER-SPEC↵
N5000:RUN↵
```

When the Illegal Operand Specifier trap condition occurs it will be reported to the ND-500/5000 Monitor as described above. The internal hardware tracer will also have triggered.

If however, the program initializes its own local trap handlers, which is very common for the most of the subsystems like NOTIS, PED, SIBAS etc. then it may be complicated to get more detail about the trap condition.

Usually the local trap handlers are set up in one of the first subroutines in the program.

Debug place the failing program.

Find the start address of the program and look at the program from the start address and search for the first call to a subroutine.

Put a break point just after this call. Start the program. The program will then stop at the breakpoint. Then disable the local trap handler and reset the breakpoint and continue the process. When the trap condition occurs the trap will be reported to the ND-500/5000 Monitor and the internal tracer will trigger.

The action to be taken is the same as done in point 1.

3. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message will be written on the error device. The internal hardware tracer will have been triggered.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 42B at: 1000001062B ND-500/5000 Illegal
Operand Specifier
```

Example

The following error message may occur:

ILLEGAL OPERAND SPECIFIER	
At program address:	'1 1062B
From CPU in slot position:	6D

Use the LOOK-AT-PROGRAM command to get the instruction.

```
N500: LOOK-AT-PROGRAM 1'1062 ↵  
P 1 1062B:W MOVE B.20,+200B EX↵
```

In this situation an attempt to write into a constant is done. This could be a program error or the program address has been wrongly calculated. It is most likely that it is a hardware fault on the ND-5000 CPU if the program has run through ok before.

TRAP HANDLER MISSING

If a Traphandler Missing trap condition is reported back to the ND-500/5000 Monitor the following error message will be written:

```

TRAPHANDLER MISSING
At program address:      1      2237B
From CPU in slot position:      6D
  
```

An Error message will in addition to the user error message be written on the Error device. This error message will also appear when this process is a RT program or standard domain started up directly from Sintran.

```

20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number:  45B at: 1000001062B ND-500/5000
Traphandler missing
  
```

Explanation

The Trap Handler Address register (THA) points to the base of an array of 64 elements (32-bits) in the data memory. This register is normally initiated in the registerblock together with P and PS register during place of the program.

Each element contains the start address of a trap handler routine in the program memory.

The Nth element of this array must hold the start address of the trap routine to handle the Nth trap condition.

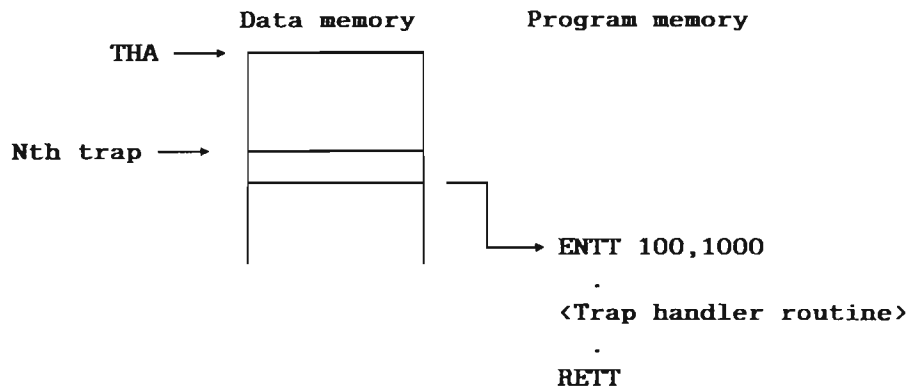
If the contents of the THA register + Trap $N*4 = 0$, the microprogram will issue a Trap Handler Missing message to the ND500/5000-Monitor. Trap N is in the range 9-360 (ref: (ND-05.009) N500-Reference Manual).

If the location pointed to by the Nth element does not contain an ENT instruction, the microprogram will issue a Trap Handler Missing trap message to the ND-500/5000 Monitor.

To be able to access this array, the Nth trap has to be locally enabled. The Nth bit in OTE1 or OTE2 has to be set. If not, the trap no. 9-29 will be ignored or for trap no. 30-36 the microprogram will report the trap to the ND-500/5000 Monitor.

Trap no. 30-36 will always be taken care of, either by a local trap handler or by the ND-500/5000 Monitor.

Trap no. 37-41 are fatal traps and will always be reported directly to the ND-100.

**NOTE**

This error message could appear if you have a linking problem on your domain.

Normally the traphandler vector table and the traphandler stack is located behind the system stack. If the program attempt to use more space than it is on the system stack, the traphandler vector can be overwritten. So, when a trap condition occurs and the trap is going to be handled by the process itself a trap handler missing trap may occur dependent of the value in the trap handler vector entry.

The trap leading to trap handler missing can be found in the message buffer location 628 relative byte address to the link location.

Action

1. If the trap has been reported to the monitor as example above then
 - Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
 - Write down or get a hardcopy of the error message.
 - Have this program run through before without this error, and no relinking or any modification has been done since then with this program?

If YES:

- Use the N500/5000 message decoder to look at the message buffer for this process. Write down which trap has lead to trap handler missing. The rest of the information in the message buffer will belong to this trap.
- Relink the domain by the Linkage Loader and rerun the program. If the domain is linked to library segments and these library segments are replaced with new revisions this trap may occur.

If the problem is still present:

- Reload the library segments used by this process and relink the program.

If the problem is still present:

- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above. The slot position from the error message can be used to locate the failing ND-5000 CPU in the MF crate incase there is a ND-5900 system with more than one ND-5000 CPU.
- Run the same program on the new ND-5000 CPU to see if the problem has been solved or not.

If the problem is still present:

- The Trace dump should be investigated.

If NO:

- There could be a program error or a linking problem.
- Try to relink the domain by the Linkage Loader. Rerun the program. If the domain is linked to library segments and these library segments are replaced with new revisions this trap may occur.

If the problem is still present:

- Reload the library segments used by this process and relink the program.

If the problem is still present:

- There is something wrong with the program. If the program overwrites the trapvector table which is located behind the system stack, this error situation may occur.

2. If the process is a RT program or has been started as a standard domain direct from Sintran and not from the ND-500/5000 Monitor only an error message is will be written on the error device.
The internal hardware tracer will have been triggered.

```
20/08-12:35 Nd-500/5000 CPU 1: process 1: (SYSTEM)TERMINAL-52
ND-500/5000 trap number: 45B at: 1000002237B ND-500/5000 Trap
Handler Missing
```

- Dump the internal hardware trace module to a file. Ref. ND-5000 Trace Module chapter.
- Write down or get a hardcopy of the error message.
- Have this program run through before without this error, and no relinking or any modification has been done since then with this program ?

If YES:

- Relink the domain by the Linkage Loader and rerun the program. If the domain is linked to library segments and these library segments are replaced with new revisions this trap may occur.

If the problem is still present:

- Reload the library segments used by this process and relink the program.

If the problem is still present:

- Exchange the ND-5000 CPU with a new one and put the failing ND-5000 CPU into the box together with the floppy with the trace dump and a hardcopy of the error message and the information described above.
- Run the same program on the new ND-5000 CPU to see if the problem has been solved or not.

If the problem is still present:

- The Trace dump should be investigated.

If NO:

- There could be a program error or a linking problem.
- Try to relink the domain by the Linkage Loader. Rerun the program. If the domain is linked to library segments and these library segments are replaced with new revisions this trap may occur.

If the problem is still present:

- Reload the library segments used by this process and relink the program.

CHAPTER 7 SOME USEFUL DEBUGGING COMMANDS IN THE ND-5000 MONITOR

All these commands are described in detail in the manual:

ND-500 LOADER/MONITOR

ND-60.136.04

Before any debugging commands are used, the program must be moved into the user's virtual memory. This is done by the commands:

PLACE-DOMAIN or DEBUG-PLACE.

If patches to the program segment are to be done, the DEBUG-PLACE must be used.

N5000:DEBUG-PLACE <domain name>↵

By using the LOOK-AT commands, it is possible to display and modify registers and locations in program and data memory.

An address in the current domain is specified as:

<segment no><segment relative address>

7.1 DEBUGGING COMMANDS

7.2 LOOK-AT-PROGRAM

- LOOK-AT-PROGRAM <address>,<domain>

This command displays and modifies program memory or program segments. The display is started at the specified address.

If domain is specified, the program segment file is displayed and may be modified. (default :PSEG)

7.3 LOOK-AT-DATA

- LOOK-AT-DATA <address>,<domain>

This command is similar to the LOOK-AT-PROGRAM, except that the data memory or data segment is involved.

7.4 LOOK-AT-FILE

- LOOK-AT-FILE <address>,<file name>

This command is similar to the LOOK-AT-PROGRAM and LOOK-AT-DATA, except that the segment file name can be :PSEG or :DSEG or any file. The patches are done direct on the file. This command should be used if pathing on the segment file is required.

7.5 LOOK-AT-STACK

- LOOK-AT-STACK

The current local datafield is displayed. This is the memory area pointed to by the current B-register and contains the subroutine call information. The sub-commands PREV and NEXT can be used to display previous stack or next stack.

7.6 LOOK-AT-RELATIVE

- LOOK-AT-RELATIVE <relative to>

Starts listing of data memory relative to either the contents of the R-register, B-register, I1-register I2-register, I3-register, I4-register, or an absolute address.

7.7 LOOK-AT-REGISTER

- LOOK-AT-REGISTER <register name>

The specified register in the context block for this process is displayed. If ↵ is typed, the next register is displayed. Registers identified as MIC are used by the microprogram.

7.8 LOOK-AT-SRF

- LOOK-AT-SRF <SRF-address>

Examine location in the Scratch Register File (4 K x 32 bit memory) on the ND-5000 cpu from the specified address. Ref. Appendix Allocation of SRF register.

7.9 LOOK-AT-RESIDENT-MEMORY

- LOOK-AT-RESIDENT-MEMORY <address>

Examine physical location in shared memory from the specified address. Address 0 is the start of shared memory. This command is privilege and can only be done from user SYSTEM.

7.10 LIST-ACTIVE-SEGMENT

- LIST-ACTIVE-SEGMENT <process no>

ND-5000:LIST-ACTIV-SEGMENT↵

Process number:↵

Process no.: 1B : (N5000-MAINT-MAN)TERMINAL-55
Process seg: Phys seg: 54B: (N5000-MAINT-MAN)TERMINAL-55
Instr.seg 26B: Phys seg: 7B: (PACK-TWO:DOMAIN-USER)LINKAGE-LOAD-H00:PSEG
Data seg 26B: Phys seg: 62B: (PACK-TWO:DOMAIN-USER)LINKAGE-LOAD-H00:DSEG

If process number is -2 then all segments for all processes are displayed. This may only be done from user SYSTEM.

7.11 SET BREAKPOINTS

- BREAK <address>,<count>

This command sets a breakpoint at the specified address. If a positive number is specified for the count argument, the breakpoint will be passed (count-1) times before reaction.

When the breakpoint is reached, the execution will be terminated.

A breakepoint instruction (BP) will be written in to the specified address in program memory and breakpoint trap condition will be enabled in MTE register.

- RESET-BREAKS <break number>

The specified breakpoint is reset. If no parameters are given, all breakpoints will be reset.

7.12 ADDRESS TRACE

- TRACE <address>,<data type>

Whenever the location at the specified address is modified during program execution, the new value is displayed. This command uses the LL and HL registers located in the IDU.

NOTE:

This command enable for Address Trap Write (ATW). When this command is used the ND-5000 cpu will be slowed down (Slow1) during execution of this process as long as Address Traps are enabled.

- GUARD <address>,<data type>,<limit1>,<limit2>

If no limits are given, any modification of the location specified causes a guard violation error and the program is terminated.

If one or two limits are specified, the value of the specified location is checked against this range. If the value is outside this range, there is a conditional guard violation and the program is terminated.

This command uses the LL and HL registers located in the IDU.

NOTE:

This command enable for Address Trap Write (ATW). When this command is used the ND-5000 cpu will be slowed down (Slow1) during execution of this process as long as Address Traps are enabled.

- EXHIBIT-ADDRESS <prog.address>,<data address>,<type>

With this command a breakpoint is set in the specified program address. When the execution reaches this breakpoint, the specified data address and its content is written to the output device.

7.13 RESET ALL DEBUGGINGS ACTIVITY

- RESET-DEBUG

This command clears the effect of all previous used commands.

7.14 TRAP HANDLING

- ENABLED-TRAPS

This command lists the content of the OTE (Own Trap Enable) register of the current domain and the MTE (Mother Trap Enable) register.

- LOCAL-TRAP-DISABLE <trap condition>

This command clears the bit in the OTE register corresponding to the specified trap condition, thereby disabling the trap handling for that trap condition.

If ALL is specified, all traps will be locally disabled. The OTE register will be cleared.

7.15 PROGRAM EXECUTION CONTROL

- RUN

This command will restart the program from the beginning.

- CONTINUE

This command restarts the execution of the program at the current program counter.

- STEP <step start>,<execut. start>,<count>

Single step. If no parameter is given, the instruction pointed to by the program counter is disassembled and shown on the output device. By typing ↵, this instruction will be executed.

7.16 DISPLAY ERROR MESSAGES FROM MONITOR CALLS

- AUTOMATIC-ERROR-MESSAGE

On typing this command, errors caused by monitor calls will automatically be written to the communication device.

7.17 RESIDENT PLACE

- RESIDENT-PLACE <domain name>

This command will place the domain permanently in memory. The command is used to avoid swapping and is only allowed for user SYSTEM.

7.18 LIST ECO LEVEL AND VERSION OF BASIC ND-5000 HARDWARE/SOFTWARE

- VERSION

This command will make a printout of the version of:

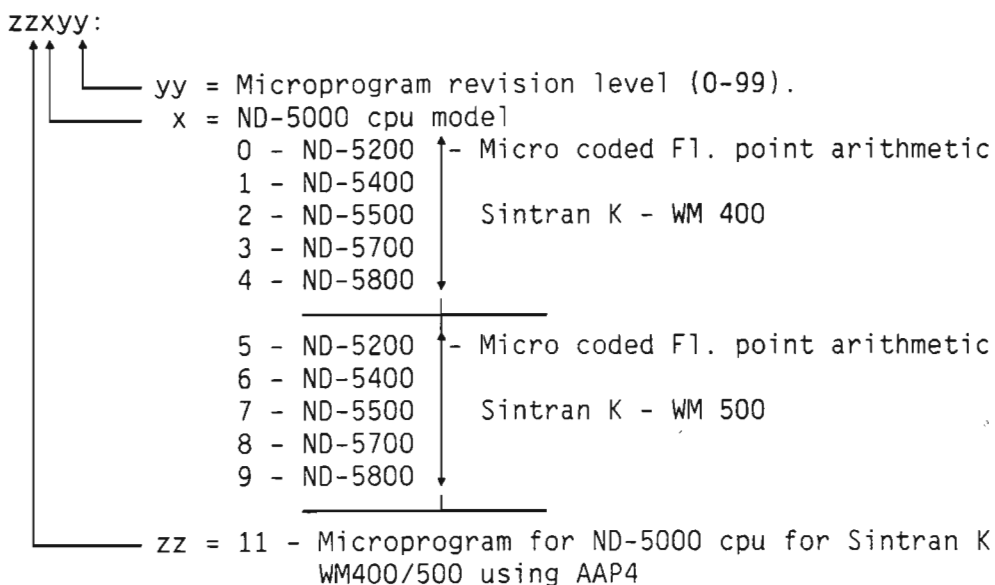
- the current active subsystem (background part of the monitor)
- the system part of the monitor
- the swapper
- the microprogram
- the ACCP program
- the ECO level on every "baby" module on ND5000 CPU.

ND-5000: VERSION↵

```
SUBSYSTEM PART: 87. 9. 1 REV.-I01
SYSTEM PART...: 87. 9.17
SWAPPER.....: REV.-I03
MICRO PROGRAM.: 11323
ACCP VERSION..: 87.11.27 E0
```

```
Module: MB.2 ALU.1 AAP.4 IDA.2 MMS.1 CS.2 CACHE.1 MIC.1 ACCP.1
ECO no: 12a 2c 2a 5a 30c 3b 2b 0d 5c
```

From the microprogram version the following may be decoded:



7.19 LIST MEMORY CONFIGURATION

• MEMORY-CONFIGURATION

List the memory configuration specified in the ND-5000 system.

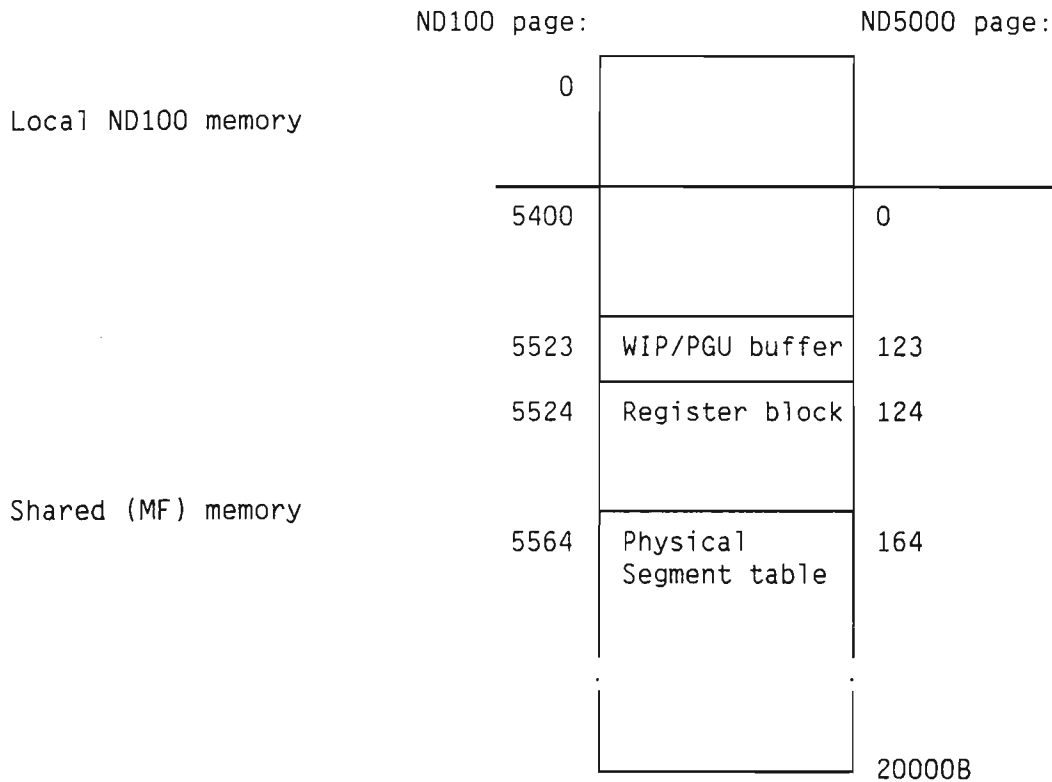
Example

Memory configuration of a ND-5000 system with 16 Mbyte shared memory and 5,5 Mbyte of local ND100 memory.

ND-5000: MEMORY-CONFIGURATION↵

PART	WIDTH	N100	N500P	N500D
OB-	17777B	Y	Y	Y

	PAGE		WORD	BYTE
	ND-100	ND-500	ND-100	ND-500
ND-500 address zero:	005400	000000	00013000000	00000000000
ND-500 register block:	005524	000124	00013250000	00000520000
Physical segment table:	005564	000164	00013350000	00000720000
WIP/PGU table:	005523	000123	00013246000	00000514000



7.20 FORCED STOP ON THE ND-5000

• STOP-ND-500

This command will stop the ND-5000 and log out all users logged into ND-5000. This command should only be used if a stop is required, and must be used from user SYSTEM. If someone attempts to start a ND-5000 process after this

command, a "warm start" of the ND-5000 will be issued.

7.21 DUMP OF THE SWAPPER DATASEGMENTS

- DUMP-SWAPPER <file name>

The Swapper data segments are dumped to a file. This dump may be investigated later by the command INSPECT-DUMP. The command is restricted to user SYSTEM only. This command intended to be used after a fatal error from Swapper. If so, the FERROR flag must be reset before the command is given. The FERROR flag is reset by giving the two commands: DEBUG-SWAPPER ON followed by DEBUG-SWAPPER OFF.

7.22 DUMP ALL HARDWARE REGISTERS

- LOOK-AT-HARDWARE <hardware register>

ND-5000: LOOK-AT-HARDWARE↵
Register name: HELP↵

MMS
MIC
IDU
IAC
DAC
SPEC

ND-5000: LOOK-AT-HARDWARE↵
Register name:↵

P	01000072036	B	01000463604	L	01000072036	R	01000224210
X1	01000224210	X2	01000250374	X3	00000000034	X4	01000224374
A1	00000000000	A2	00000000000	A3	00000000000	A4	00000000000
E1	00000000000	E2	00000000000	E3	00000000000	E4	00000000000
SC1	01000463634	SC2	00000000054	SC3	00000000000	SC4	00000000000
SC5	01000463604	SC6	01000224210	SC7	00000000017	SC10	00000000003
SC11	00000177400	SC12	00000000000	SC13	00000020034	SC14	00000000000
Q	00000000040	STS	00000000000	TE	00000000000	LC	00000000140
RFA1	00000002001	RFA2	00000007771	MOD	37700000107	EA1	37777777777
EA2	01000464343	EA3	00000450400	IAR	01000070046	DACR	00000024474
SC2	00000000054	SRF0	00000000000	SRF1	14631463146	SRF2	14000000000
SRF3	00017777777	SRF4	17760000000	SRF5	30000000000	SRF6	17777777777
SRF7	02104210420	SRF10	00000000000	SRF11	37777777777	SRF12	01000503244
SRF13	11022000003	SRF14	00000000000	SRF15	00000000000	SRF16	00530160200
SRF17	00000000040						

This command will stop the microprogram and the microprogram is stored in the hardware register dump routine. This command is privilege and must be done from user SYSTEM and all processes in the ND-5000 must have been logged out.

After this command has been issued the microprogram has to be started in address 0 and the Swapper has to be started.

7.23 OPERATIONS AGAINST THE CONTROL STORE

- LOOK-AT-CONTROL-STORE <address>

This command is used to examine and modify the ND-5000 microprogram.

- COMPARE-CONTROLSTORE <filename>,<start address>,<no of words>,
<max no of fault>

```
N5000:COMPARE-CONTROL↵
File name:↵
Start address:↵
Number of words:↵
Max number of faults:↵
```

```
UNEQUALITY FOUND AT: 000003:
CONT STORE: 100030 000000 000000 000000 000000 050000 100000 020000
FILE:       100030 000000 000000 000000 000000 050000 100000 001340
```

```
UNEQUALITY FOUND AT: 000016:
CONT STORE: 100030 000000 000000 000000 000000 050000 100000 020000
FILE:       100030 000000 000000 000000 000000 050000 100000 001340
```

NOTE

This command doesn't function with ND-500/5000 Monitor version I.

- MICRO-START <address>

This command starts the execution of the ND-5000 microprogram at the specified address.

7.24 RESET ND-5000 CPU

- RESET-CPU

- The octobus message RESET-CPU is sent to the ACCP and a Master Clear on ND-5000 cpu is performed
- then the FERROR flag is cleared
- the microprogram is stopped and an echo test on the octobus between the ND-100 and the ACCP is run.
- then the memory configuration is tested by sending the address of the ACCP buffer located in the multi port memory to the

ACCP and verify that the System monitor and the ACCP "agrees" on the location of the buffer.

NOTE

This command brings the ND-5000 out of a "hangup" situation.

- MASTER-CLEAR

- The octobus emergency message MASTER CLEAR is sent to the ACCP. A Master Clear on the ND-5000 cpu is performed and a short selftest is started. The selftest status is read and verified.
- then the FERROR flag is cleared
- the microprogram is stopped and an echo test on the octobus between the ND-100 and the ACCP is run.
- then the memory configuration is tested by sending the address of the ACCP buffer located in the multiport memory to the ACCP and verify that the System monitor and the ACCP "agrees" on the location of the buffer.

NOTE

This command brings the ND-5000 out of a "hangup" situation.

After this command has been issued the control store has to be loaded.

- RUN-SELFTEST

The long selftest will started on the ND-5000 cpu. The comand is privilege and all processes has to be logged out of ND-5000. A long version of the selftest will take approx. 3 min.

NOTE

This command doesn't function on Sintran K - WM406.

- Internal Trace Module maintenance commands.

```
INIT-TRACER
ARM-TRACER
DISARM-TRACER
DUMP-TRACE-MEMORY
CLEAR-TRACE-MEMORY
CLEAR-TRACE-ADDRESS
EXAMINE-TRACE
READ-TRACE-FILE
WRITE-TRACE-FILE
```

Ref. the chapter about the trace module for further explanation.

7.25 SET CACHE MODE

- CACHE-MODE <program cache mode>, <data cache mode>

```
ND-5000: CACHE-MODE↵  
Program cache mode:   ↵  
Data cache mode:   ↵
```

Default value is NORMAL-WICO.

NORMAL-WICO	- Use both memory and cache. Use Write In Cache Only mode on Data cache.
MEMORY	- Use memory only. The cache will be inhibited.
WRITE-THROUGH-DATA	- Use Write Through Data cache. Inhibit WICO mode.
SMART-IF-GO-PROGRAM	- Applies to the instruction cache

7.26 ATTACH ANOTHER PROCESS

- ATTACH-PROCESS <Process no.>

The PROCESS NO. is the number of the process with which communication is desired. Default is the current process that is connected to the terminal.

The commands LOOK-AT and RUN will be routed to the specified process. The process should not be connected to any other terminals. The command is used for debugging purposes, attaching to the swapper process.

CHAPTER 8 TEST AND UTILITY PROGRAMS

This chapter describes the test and utility programs for ND-5000.

8.1 ND-5000 TEST MICRO PROGRAMS

SEMICS is a collection of microprograms for test purposes for ND-5000 and runs under TPE (the Test Program Environment monitor). SEMICS runs under the operating system as a background program.

These tests are grouped in "test-systems", each consisting of one or several micro-tests (max 256). The purpose for SEMICS is to detect errors in the ND-5000 CPU. The commands are made as similar as possible to the commands for test microprograms for ND-500 (TEMICS).

8.1.1 DEFINITIONS

Testsystem is a collection of 1-256 **tests**. A testsystem is started by the command 'RUN'.

Test is the smallest unit which is visible to the user. A test is started by the command 'TEST'.

Subtest is a part of the test (some tests consist of several subtests). The subtest is invisible to the user.

8.1.2 SWITCHES ON THE CACHE

SEMICS is able to detect whether the cache module is present or not. When running microtests using the cache or demanding cache present, the switches for both data and instruction cache should be in the 'ON' position. Ensure that this is the case before starting cache dependent micro tests.

8.1.3 LOADING AND STARTING SEMICS

SEMICS is delivered on two floppy discs.

1. The floppy disc 250247-XX-01D with :
 - TPE-monitor
 - SEMICS overlay files
 - Extended Error Information

2. The floppy disc 250247-XX-02D with the collection of the microtests.

You can run the microtests in two different ways:

1. Running microtests from the floppy.
2. Running microtests from the harddisc.

In both cases you must load the TPE monitor and the SEMICS overlay files to the harddisc.

If you want extended error information, the microtests must also be run from the harddisc. The PD sheets describe how to perform the loading.

STARTING THE TPE MONITOR

First you must start the monitor:

1. Log in as user SYSTEM.
2. Type the following:

```
@(ND5000-SEMICS)TPE-MON-100-:PROG↵
```

When the TPE monitor has started, it will identify itself with this heading:

```
TPE Monitor, ND-100 / ND-110 - Version: lnn - yyyy-mm-dd
```

The TPE monitor is now ready to accept your commands.

STARTING SEMICS

When the TPE monitor is running, you can load the testprogram:

```
>Load (ND5000-SEMICS)SEMICS↵
```

The testprogram will identify itself, and the program commands will be available.

NOTE

If you are running the microtests from the floppy disc: When you have entered the floppy disc with microtests (250247-XX-02D), you must specify directory and user name when you run the first test.

Example:

```
TPE>RUN↵  
Test system: (250247-XX-02D:FLOPPY-USER)<Test name>↵
```

8.1.4 USING SEMICS

This section describes how to use SEMICS.

PREPARE FOR TESTING

The first step in a normal test sequence is to get the configurant dependent parameters into the program and reserve some memory. This is done with the command:

`>INVESTIGATE-AND-RESERVE`

This command investigates the system by reading some parameters about the system. After this, the operator is asked whether or not to reserve memory for test purposes.

TEST NUMBERING

Each testsystem contains one or several tests (max. 256). These tests are numbered from 1 to <n>, where <n> is the number of the last test in the testsystem.

THE INITIAL PARAMETER VALUES

SEMICS uses several parameters to control the testing. Some of these parameters must be set according to the hardware configuration of the system. The command

`>INVESTIGATE-AND-RESERVE`

will find the ND-5000 configuration and update the corresponding parameters.

NOTE: The ECO fields must be strapped correctly.

The system parameters found by the `>INVESTIGATE-AND-RESERVE` command can at any time be suppressed by using the `>SET-PARAMETER` command.

ACCESSING FILES THROUGH SEMICS

As SEMICS is running as an background program from user SYSTEM, all files will normally be owned by user SYSTEM. This applies to files accessed through commands in the TPE monitor (i.e. all commands that the command >MONITOR-HELP gives you).

For instance, if you are using the TPE monitor command SET-PRINTER-FILE with a filename without the user/directory prefix, the file will be searched under user SYSTEM.

When using some SEMICS commands that require a filename as parameter, please note that:

1. Files accessed through SEMICS commands WITHOUT the user/directory prefix will be searched under the user that SEMICS is taken from. This is usually user ND5000-SEMICS.
2. Files accessed through TPE commands such as MODE will be searched under user SYSTEM.

RUNNING MICROTTESTS

The microtests are divided into testsystems. Each testsystem consists of one or several tests. Each test can be executed with one, two, three or four different types of input data. When running a test, several repetitions can be executed for each input type. Different sets of inputs will be used in each repetition.

The input types and the number of repetitions for a test are defined and described on the file :TEST.

The tests can be run in one of three modes :

1. All tests in all testsystems in sequence
COMMAND : COMPLETE-RUN.
2. All tests in one testsystem
COMMAND : RUN.
3. One test in one testsystem
COMMAND : TEST.

You will be asked if you want to interrupt the execution when a test fails. If you answer YES, or you press the escape key when a test is running, you will be asked about the next action.

The next action can be :

- REPETITION: Continue the execution on the next repetition.
- INPUT-TYPE: Continue the execution on the next inputtype.
- TEST-ABORT: Abort the execution of the current test. Continue on the next test if it is defined.
- ABORT: Abort the execution of the microtests.
- DUMP: Dump the last input/output parameter values.

The parameter ERRMAX indicates how many errors which can occur in a test before the test aborts.

ERRMAX can be changed by the command SET-PARAMETER.

ERROR IN MEMORY CONFIGURATION

When reserving memory or investigating the system, the MON 60 (ND-5000 monitor call) can result in the error message "ERROR IN MEMORY CONFIGURATION". When this happens, the reason often is an inconsistency between the memory configuration information given to the ND-5000 background monitor and the real physical memory configuration (including also switch settings). In some cases there could of course also be a real HW-problem on the boards involved.

When you get the message ERROR IN MEMORY CONFIGURATION, try the following hints:

- Restart the ND-5000 MONITOR and give the command:

ND-5000: MEMORY-CONFIGURATION↵

A list of parameters will be displayed. Compare these parameters with your PHYSICAL local memory size and multiport size. Make sure that the parameter values and HW-switch values are consistent.

- Update the parameters using the command:

ND-5000: DEFINE-MEMORY-CONFIGURATION↵

(Remember to update the HENT-MODE file with these values as well).

- Reload SEMICS and use the command

>LIST-PARAMETERS↵

to check the current testparameters. Update the values by giving the command

>SET-PARAMETER↵

and start specify thje parameters you wish. (Remember that ALL your test parameters must now be set manually by you with the command >SET-PARAMETER before testing).

TIME CONSUMPTION

8.1.5 LOGGING ERRORS

If you detect any errors on the CPU when running SEMICS, you must return the CPU to the ND repair center. Then please include a file containing the error messages from SEMICS where the errors were detected. This file is called LOG-FILE:SYMB.

Output from SEMICS can be assembled on the file by using the TPE commands :

```
>SET-PRINTER-FILE  
>SET-PRINTER-MODE
```

The following example illustrates how to direct output from SEMICS to the terminal and to the file LOG-FILE:SYMB at the same time.

```
>SET-PRINTER-FILE LOG-FILE:SYMB ↵  
>SET-PRINTER-MODE DUPLICATED ↵
```

8.2 ND-5000 TEST MACRO PROGRAMS

8.2.1 TEST PROGRAMS

8.2.1.1 DESMODUR

The ND-500/ND-5000 verification system is designed to verify and check the ND-500 and ND-5000 instructions and operation. Three types of tests are performed:

- Bootstrap; only limited aspects of an instruction or operation are tested.
- Full; this is intended to be a test of the hardware and of the microcode used in the instruction or operation.
- Microcode; this is mainly to check the microcode. It is assumed that the hardware used in the instruction or operation works correctly.

The testrun starts by executing simple instructions. These instructions are then used for more complex tests. However, the first simple instructions must themselves be tested, and this is done by a test-module called bootstrap.

Instructions are executed, and status registers and memory cells affected by the instruction or operation are compared against expected data. If they match, the instruction or operation is assumed to work correctly and the test proceeds. If any error is detected, all relevant error information is saved and can be displayed upon request.

A Warning

The verification system is mainly intended to verify the logical functions of the macro-instruction set. It verifies that the macro instructions are executed logically correct and gives the correct status and result. It is not a specific hardware-design test.

Because of the extensive testing, it is possible that the verification test will discover marginal or hard hardware errors with no connection to the instruction currently being tested. Hardware errors may occur in the test result/parameter setup, or in the checking afterwards. This may lead to confusing error information, and it will be necessary to check what really caused the error message.

The verification test will NOT test the different instructions under all conditions, such as:

- page fault
- TSB-miss
- cache-miss
- PXING etc.

Therefore, an instruction which is tested by the verification test, may fail in another program where some of the external conditions are different.

HOW TO GET STARTED

ND-500

Verification

System User

The ND-500 verification system runs under the user N500-VERIFY-DOM. This user has all test programs linked together ready for execution, and all the files necessary to run the verification system.

System Initiation

Log in as user N500-VERIFY-DOM:

```
13.24.24      9 JUNE  1987
SINTRAN III - VSX/500  VERSION K
ENTER N500-VERIFY-DOM ↵
PASSWORD:  ↵
OK
@
```

Enter the ND-500/ND-5000 monitor:

```
@ND ↵
ND-5000 MONITOR  VERSION XXX
ND-5000:
```

Activate the verification system DESMODUR:

ND-5000: DESMODUR ↵


```
*****  
**** N D - 5 0 0 0 V E R I F I C A T I O N S Y S T E M ****  
****                                VERSION: 87.01.27                                ****  
*****
```

NOTE: Now you must specify some testrun parameters (CPU type etc.) by answering the following questions:

```
DO YOU WANT DEFAULT TESTRUN(Y/N/H=HELP).....(Y)? : N
SPECIFY MONITOR TYPE(M=MULTI-USER,S=SINGLE-USER,"/")....(M)? :
DO YOU WANT ALL MODULE NAMES LISTED(Y/N/"").....(N)? :
TESTRUN-MODE(D=DISPLAY,C=CONTINUOUS,S=STOP,"/").....(S)? :
TESTRUN SPEED(C=COMPLETE,Q=QUICK,"/").....(C)? :
MAXIMUM(377B) NUMBER OF ERRORS ALLOWED(OCTAL"/").....(377)? :
CPU TYPE=ND-500(G),ND-5000(S) or ND-5000 w/ 570 FP (X)...(G)? : S
TEST MODULE RUN (A=ALL, O=ONE MODULE, "/").....(A)? :
SPECIFY NAME OF FIRST MODULE IN TESTRUN.....(IFGOBOOT)? :
```

NOTE: Default values of directives can be specified by CR; the default directive is shown on the right in parentheses. The "/" character can be used to correct any incorrect directive given; it deletes all directives given and restarts the request sequence.

Numerical input must be ended with CR. Non-numerical input must not be ended with CR, but the module name must be ended with CR. If an improper directive is given, an error message is printed and the request is repeated.

The Testrun Parameters

DO YOU WANT DEFAULT TESTRUN (Y/N/H=HELP).....(Y)? :

Default testrun means that default testrun parameters will be substituted by the verification system and the testrun initiated.

H (HELP) gives a list of the default directives:

Default testrun is:

MONITOR TYPE:	M=MULTI-USER
TESTRUN MODE:	S=STOP
TEST MODULE RUN:	A=ALL MODULES
TESTRUN SPEED:	C=COMPLETE TEST
MAXIMUM NUMBER OF ERRORS ALLOWED:	377B
CPU-TYPE:	G=ND-500

SPECIFY MONITOR TYPE(M=MULTI-USER,S=SINGLE-USER,"/").(M)?:

Incorrect monitor type causes error messages to be printed which are not relevant to the instruction or operation being tested.

DO YOU WANT ALL MODULE NAMES LISTED(Y/N"/").(N)?:

Y (YES) gives a list of the test-modules in the system:

```

.....P A R T 1:SIMPLE INTEGER INSTRUCTIONS.....
IFGOBOOT      LOADWORD      WORDADD      TYPELOAD
ADDRESSING     SIARITH       LOGICAL       DESCADDR
.....P A R T 2:ARITHMETIC AND CONTROL INSTRUCTIONS.....
MIARITH-1     MIARITH-2     FLOAT-COMP   FLOAT-ARIT   FLOAT-FUNC
FLOAT-CONV-W  FLOAT-CONV-F  FLOAT-CONR  LOOPI       LOOPD
LOOPST        INDEX        STBITS      BITFICKLE    SHIFTOP
LOADADDR-1    LOADADDR-2    LOADADDR-3   LOADADDR-4   LOADADDR-5
LOADADDR-6    LOADADDR-7    LOADADDR-8   LOADADDR-9   LOADADDR-10
LOADADDR-11   LOADADDR-12   JUMPER      ERBE-INIT    ERBE-ENTM
ERBE-ENTF     ERBE-ENTFN   ERBE-ENTS   ERBE-ENTSN   ERBE-ENTT
ERBE-ENTB     BUDDY        I-F-REG
.....P A R T 3:STRING INSTRUCTIONS.....
BMOVEFILL     STRINGS-1     STRINGS-2     STRINGS-3     STRINGS-4
STRINGS-5     STRINGS-6     STRINGS-7     STRINGS-8     STRINGS-9
STRINGS-10    STRINGS-11    STRINGS-12    STRINGS-13    STRINGS-14
STRINGS-15    STRINGS-16    STRINGS-17    STRINGS-18    STRINGS-19
.....P A R T 4:TRAP HANDLING.....
SPEC-REG      TRAP-1        TRAP-2        TRAP-3        SPEC-INS
TRAP-N-1      TRAP-N-2      TRAP-N-3      TRAP-N-4      TRAP-N-5
TRAP-N-6      TRAP-N-7      AD-HOC
.....P A R T 5: MODULE REQUIRING SEPARATE TESTRUN.....
SSMOV-1       SELFTEST

```

TESTRUN-MODE(D=DISPLAY,C=CONTINUOUS,S=STOP,"/").(S)?:

D (DISPLAY): In this mode, no error checking and handling is performed. Apart from those modules testing string instructions, all test-modules eject this mode, i.e., they act as if no testing is to be performed. Modules testing string instructions display the entire test conditions for the first string instruction in the module; then, control is given to the operator.

C (CONTINUOUS): If an error is detected during the testrun, a brief error message is printed. Then, the testrun is automatically resumed.

S (STOP): If an error is detected during the testrun, an extensive error message is printed. Then, control is given to the operator. The operator can now use the testrun commands.

TESTRUN SPEED(C=COMPLETE,Q=QUICK,"/").....(C)?:

C (COMPLETE): The testprogram uses a complete set of test conditions during the testrun, i.e., the instruction or operation is exhaustively tested.

Q (QUICK): The testprogram uses a quick set of test conditions during the testrun, i.e., the instruction or operation is not thoroughly tested. Apart from those modules testing string instructions, all test-modules ignore the testrun speed and always execute with a complete set of test conditions.

MAXIMUM(377B) NUMBER OF ERRORS ALLOWED(OCTAL"/").....(377)?:

Total number of errors that is accepted during the testrun. If exceeded, the testrun is aborted and a message is printed on the terminal.

CPU TYPE=ND-500(G),ND-5000(S) or ND-5000 w/ 570 FP(X)..(G)?:

Selects the CPU-type to get correct data and results for testing.

TEST MODULE RUN (A=ALL, O=ONE MODULE, "/").....(A)?:

"A" means that all test modules will be executed in sequence.

"O" means that one test module will be executed in loop mode.

SPECIFY NAME OF FIRST MODULE IN TESTRUN.....(IFGOBOOT)?:

This is the first module in the testrun. The entire module name must be given. Abbreviations are not recognized. The verification system places the specified module and starts execution.

MESSAGES THAT OCCUR DURING A TESTRUN

Test-modules are placed and executed in sequence, starting and ending with the modules specified by the operator. Execution of test-modules continues as long as no errors are detected. When an error occurs, the testrun prints information about the error. At this point the testrun may stop or it may continue, depending on the value of the testrun parameters:

**TESTRUN-MODE(D=DISPLAY,C=CONTINUOUS,S=STOP,"/").(S)?:
MAXIMUM(377B) NUMBER OF ERRORS ALLOWED(OCTAL,"/").(377)?:**

If testrun stops, the operator may use the testrun commands to display additional error information given by the testprogram. The operator may continue execution of the testprogram or restart the current test-module. The operator can continue in the current module as long as the maximum number of errors is not exceeded or the end of the module reached. In these cases he can only restart the module. He can place and start the next test-module in the testrun by typing RESUME-MACRO.

LOGGING ERRORS

When an error has occurred, it is important collect as much as information about the error condition as possible. This can be done by checking registers etc.:

- Create a file where you specify all the commands and parameters you want to use. Use the file as a input-file in a mode-job.
- Dump the output onto a file

The example below shows how to log error information on a file called LOG-FILE:SYMB:

@MODE <input-file>,LOG-FILE:SYMB ↵

When you send the defect module to the ND Repair Center, please include a hardcopy of the log-file.

8.2.1.2 FLOTILJE

This section gives a description on the dataprogram FLOTILJE and explains how to use it.

GENERAL

FLOTILJE is a verification program for the floating instruction set.

There are also other verification programs used for this purpose (SUPER and LIBTEST), but they have some weak points:

- only a limited set of test data can be used
- only parts of the addressing facilities of the ND-500/ND-5000 instruction set is used
- difficult to implement execution of instructions in a controlled environment, such as:
 - Cache miss/hit
 - TSB miss/hit
 - Pagefaults/restarting
 - Word aligning
 - PXING
 - etc.
- Lack of testing with CONSTANTS. Because these will become a part of the instruction code, these operands will be routed through other parts of the hardware, and will, therefore, test other parts of the system.

The purpose of FLOTILJE is to fill this gap in the testing of the floating instruction set.

**A description of
what FLOTILJE can
do:**

- Generate RANDOM test data - no check on legal values.
- Instructions are using a wide spectrum of addressing modes, including CONSTANTS.
- Execute instructions with the following situations on operands:
 - Cache hit/miss
 - Dump dirty cache
 - TSB hit/miss
 - Pagefaults
- Operands are put in odd addresses (WORD CROSSING)

- Depending on the : LOW ADDRESS xxxxB command in the load file, one of the operands will cross the page limits and generate PXING.
- For each instruction type, it will be checked that the same data set will give the same result and status bits (ST1) for all addressing modes.
(Because of the random generation of data, it is difficult to have an exact facit result. Instead an expected result and status is made, using only register operands, and it will be checked that both result and status is the same for all addressing modes.)
- In some instructions (SIN, COS, ASIN, ACOS, ATAN, ATAN2 and SQRT), the results should lay between certain limits, and it will be checked that the result really lays between these limits.

OPTIONS OF TESTRUN MADE BEFORE STARTING

Command="L"	List the tested instructions with their addresses and operands.
Command="S"	Specify fixed test data. If you want to loop on certain test data, you may specify these as 3 * 2 octal values of 11 digits each. See example of the use of this command under the section "Running FLOTILJE".
Command="P"	Pass instructions. You may skip instructions that you don't want to test. This may be done if you notice that a certain instruction fails, and you want to loop and check the other instructions without being stopped each time the failing instruction is tested.
Command="N"	Save on a file the test data and the result/status from the test data. This file may be used as an input to a later test-run (maybe on another CPU) to compare the results exactly from one run to another. The file name is FLOTSETUP:DATA.
Command="O"	Use the test data and result/status from a file as input (see command="N"). The test data is used, and the result/status is checked against the result/status on the input file (expected result).
Command="C"	The command changes bit 23 in OTE1 from 0 to 1 or vica versa. If changed to 1, the ADDRESS-TRAP-WRITE trap will be enabled, and this will lead to a disabling of the AC-cache on the ND-5000, i.e., AC-miss will allways occur.

OPTIONS IF ERROR IS DISCOVERED

- Stop on error [S]** Stop on the first error. The error information will be displayed and the monitor prompt will occur.
You may continue with the command: CONTINUE.
- Continue on error [C]** Continue on error. The error information will be displayed, and the test run continues, generating new test data.
- Looping on failing test pattern [L]** The error information will be displayed. The test run will continue, using the failing data set as test pattern.

OPTIONS FOR ENVIRONMENT

The "HIT-TEST-FACTOR" parameter can be set from 0 to 4:

- 0 = The operands in the data memory will allways be in the cache.
- 1 = The operands will first be tested with CACHE HIT, then with CACHE MISS.
- 2 = The operands will be tested with CACHE HIT, CACHE MISS and DUMP-DIRTY on cache.
- 3 = The operands will be tested with CACHE HIT, CACHE MISS, DUMP-DIRTY on cache and MISS in TSB.
- 4 = The operands will be tested with CACHE HIT, CACHE MISS, DUMP-DIRTY on cache, MISS in TSB and PAGEFAULT.

RUNNING FLOTILJE

This section shows an example on how to run FLOTILJE using default parameters.

```
ENTER:ND5000-USER-TEST↵
PASSWORD:↵
```

```
@ND↵
```

```
ND-5000/5000 MONITOR Version I00 (Preliminary)87. 5. 6 / 87. 5.29
ND-5000: D-PLA FLOTILJE↵
```

```
ND-5000: RUN↵
```

```
=====
==          F L O T I L J E - FLOATING TEST (DHB 87.07.14)          ==
== -----                                                         ==
== TESTS DOUBLE AND SINGLE FLOAT +,*,-,/,ADD3,MUL3,SUB3.DIV3,ADD2,MUL2==
== SUB2,DIV2,MULADD,PSUM,SQRT,SIN,ASIN,COS,ACOS,TAN,ATAN,ATAN2,EXP,ALOG, ==
== ALOG2,ALOG10,REM,INT,INTR,AXI AND POLY.                        ==
== USING MOST ADDR. MODES AND CONSTANT OPERANDS. DATA IS GENERATED RANDOM =
== , RES.,ST1 AND RESULT LIMITS ARE TESTED. IT IS POSSIBLE TO SET THE ==
== "HIT TEST FACTOR" FROM 0 TO 4, WITH THE FOLLOWING MEANINGS:      ==
== 0=CACHE/TSB HIT NO PGF, 1=CACHE MISS,TSB HIT,NO PGF, 2=DUMP DIRTY ON ==
== CACHE, HIT ON TSB, NO PGF, 3=CACHE/TSB MISS, NO PGF, 4=CACHE/TSB MISS+==
== PGF.                                                             ==
== EX: IF 4 IS GIVEN, THE TESTING WILL GO FROM 0 THROUGH 4 USING THE ==
== SAME DATA PATTERNS FOR TEST WITH THE FIVE SITUATIONS DESCRIBED.==
== NB ! TSB MISS AND PAGEFAULT APPLIES TO THE FIRST OPERAND ACCESSED ==
== ONLY. (OPERANDS ARE ON THE SME PAGE (DEPENDING ON LOADING))=
=====
RUN/LIST TESTED INSTR./SPECIFY TESTDATA/PASS INSTRUCTION (R/L/S/P) /R/:↵
STOP / CONTINUE / LOOP ON FOUND ERROR PATTERN (S/C/L) /S/:↵
WILL GENERATE RANDOM TEST DATA, STOPS ON ERROR.
GIVE HIT TEST FACTOR (0-4) DEF = 3 :↵
HOW MANY LOOPS SHOULD BE DONE (DEC) (0=FOREVER) :↵
```

```
NUMBER OF INSTRUCTIONS TESTED : 2592D * 4D = 10368D
```

```
LOOPING ; HITTESTFACTOR : 0D
LOOPING ; HITTESTFACTOR : 1D
LOOPING ; HITTESTFACTOR : 2D
LOOPING ; HITTESTFACTOR : 3D
```

LOGGING ERRORS

It is possible to log errors to a file by using the @MODE command in sintran. The startup procedure and the parameters must of course be specified on the input-file. How to log errors to the file LOG-FILE:SYMB, is explained below:

@MODE <input-file>,LOG-FILE:SYMB ↵

8.2.1.3 PIPELINE-EXERCISER

This section gives a description of the Pipeline Exerciser and shows how to run it.

GENERAL

The pipeline-exerciser consist of four different fragments:

Fragment 1 uses simple macro instruction, executed in a way that creates predefined pipeline situations.

Fragment 2 exercises different addressing modes in different pipeline situations.

Fragment 3 tests instructions before and after exceptions (trap situations are used).

Fragment 4 combines a set of "heavy"-instructions and addr. modes.

HOW TO RUN THE TEST

Log into the user ND5000-PIPELINE, start the ND-5000 Monitor, and enter the following commands:

ND-5000: PIPELINE-EXERC↵

----- P I P E L I N E E X E R C I S E R -----

COMMENT

You must define two parameters before the Pipeline Exerciser can start:

TESTRUN-MODE:

E: If an error occurs, an error message will be printed out.

B: If an error occurs, the program will stop, and you have the possibility of examining registers.

LOOP COUNTER : Total number of loops in a testrun.

TESTRUN-MODE (E=ERROR-MESSAGE,B=BREAK-POINT,"/"..... (E)?"↵

TOTAL NUMBER OF LOOPS IN TESTRUN (OCTAL, "/")..... (1)"7↵

COMMENT

During the test-run, the program counts the number of loops, in this case 7 (oct).

***** E N D O F T E S T R U N *****

ND-5000:

LOGGING ERRORS

It is possible to log errors to a file by using the @MODE command in sintran. The startup procedure and the parameters must of course be specified on the input-file. How to log errors to the file LOG-FILE:SYMB, is explained below:

@MODE <input-file>,LOG-FILE:SYMB ↵

8.2.1.4 MMS-TEST

The memory management (MMS) test program is a stand-alone test program. It is designed to test the different operations performed by the memory- management-system, and the micro-code.

Its basic task is to provoke trap situations in MMS. These are:

- protect violation
- page-fault
- index-error (a type of hardware fault), on both data- and program MMS.
- Instruction sequence error (tested in conjunction with multi-domain calls)

To create these traps, the program modifies both the domain-information-tables (DIT) and the physical segment table (PST) during execution. The traps arising will be handled locally, and different tests to evaluate the trap will be performed..

Since the MMS-test creates fatal traps by altering contents in PST and DIT, it is important that no other program can access the actual table entries. Therefore, fixing of pages used by the test, has to be done before starting up. (" MMS-TEST:MACR" will do this.)

In the present version, it's necessary to know the start address of PST (i.e. PSTP). Use the comand

MEMORY-CONFIGURATION

to get this, and give it as an input parameter to the program. You must be a privileged user to be able to do that.

If errors are detected, error information will be printed to the terminal if specified.

Files

- MMS-TEST-1:xxx ,is the main program.
- MMS-TEST-2:xxx / MMS-TEST-3:xxx, are program code placed on segment 2 and 3. This code is accessed from the main program, after the segment's capability or mapping tables has been altered. Access to code on one of these segments will cause the wanted trap condition.
- MMS-TEST:MODE, will link and load the three programs to the domain "MMS-TEST".

- MMS-FIX:MACR, is a macro for preparing the test for run. It will place the program, and fix the wanted pages into memory. Before giving the command "RUN", you must make yourself a privileged user (this macro must be started from user System). This is done by patching the "PIA" bit into the domain info table for domain 0.

The MMS-test is separated into different fragments, classified by the type of trap.

RUNNING THE MMS TEST

Starting the MMS test:

- Log onto user SYSTEM
- Start the ND-500/5000 monitor
- Type the command "(N500-MMS-TEST)MMS-FIX".

This will cause a :MACR file to be executed, which will perform several patches, a MEMORY-CONFIGURATION command, and a LOOK-AT-REGISTER PS command.

When you see the "ND-5000:" prompt, you must enter the command "LOOK-AT-PHYSICAL-SEGMENT 310 x", where x is the last digit of the PS register. You must change this location to a "1", as shown in the example below.

NOTE

After entering the RUN command, you will be asked for the value of the pointer to the physical segment table. This value appeared in the output from the MEMORY-COMMAND. Since the screen is cleared when you enter the RUN command, you must take note of this number prior to giving the RUN-command.

@ND

ND-500/5000 MONITOR Version I00 (preliminary) 87. 6.16 / 87. 9. 1

ND-5000: (N5000-MMS-TEST)MMS-FIX↵

N500: D-PLACE (N-MMS)MMS-TEST↵

N500: LOOK-AT-REG ST1↵

ST1 : OB PER

ST1 : OB 2

ST2 : OB .

N500: FIX-S-S 1 P 0 217777B↵

N500: FIX-S-S 1 D 0 133777B↵

N500: FIX-S-S 2 P 10000000B 10007777B↵

N500: FIX-S-S 2 D 10000000B 10007777B

N500: FIX-S-S 3 P 10000000B 10007777B

N500: FIX-S-S 3 D 10000000B 10007777B

N500: MEM

PART WIDTH	N100	N500P	N500D
OB- 17777B	Y	Y	Y

	PAGE		WORD	BYTE
	ND-100	ND-500	ND-100	ND-500
ND-500 address zero:	004000	000000	000100000000	000000000000
ND-500 register block:	004112	000112	00010224000	00000450000
Physical segment table:	004152	000152	00010324000	00000yyyyyy
WIP/PGU table:	004111	000111	00010222000	00000444000

N500: LOOK-AT-REGISTER P

PS : 1142200000xB

ND-5000: LOOK-AT-PHYSICAL-SEGMENT 310 X

PHSEG 0	310B:	OB PERMIT-DEPOSIT
PHSEG 0	310B:	OB <u>BYTE</u>
PHSEG 0	310B:	OB <u>1</u>
PHSEG 0	311B:	OB <u>.</u>

ND-5000: RUN

```

*****
*****
****
****      M E M O R Y   -   M A N A G E M E N T   -   T E S T      ****
****
****                      V E R S I O N   :   870618                      ****
****
*****
*****

```

POINTER TO PHYSICAL SEGMENT TABLE (OCTAL,"/").....(-)?:y

IS THE PROGRAM RUNNING ON A 5000 CPU (Y/N ,"/").....(Y)?:

TESTRUN-MODE(E=ERROR-MESSAGE,B=BREAK-POINT,"/").....(E)?:

TOTAL NUMBER OF LOOPS IN TESTRUN (OCTAL,"/").....(1)?:

FRAGMENT TO RUN (1,2,3,4,5,6,7,10,11,12,13 H=HELP,"/")..(ALL EXPT 13)?:

FRAGMENT 12 IS RUNNING -- LOOP NUMBER : 00000000000B

END OF TESTRUN

ND-5000: ex

LOGGING ERRORS

To save error messages reported you can use the philosophy:

@MODE <input file>,<err-mess-file>*

where <input file> is a mode-file starting up the test with default parameter set and <err-mess-file> is an output file to store the error printouts. The only thing you should remember is that you have to set the priv. instructions allowed first.

8.2.1.5 PAGE FAULT EXERCISER

The PAGE FAULT EXERCISER is a macro written test program that supplies test instructions running in a page-fault environment. The PAGE FAULT EXERCISER test runs test sequences in which instructions deliberately overlap page boundaries. This is done to insure that the hardware handles the page fault correctly. The best way to get familiar with the test is to start it, and then enter the INFORMATION command.

FIRST THE SWAP-FILE SELECTION

You must first decide whether you want to use the current swap file for swapping or directly your DSEG, used as original segment. Using the standard swap-file is much faster for starting and exiting, but will steal about 30000 pages to the available swapping space. the DSEG option does not take any swap-file resources, but whenever you leave the program, the swapper will rewrite the 29 segments back to the DSEG, which will take about 1 MN...

Two mode-files are available for changing the swapping path, logged under PAGE-FAULT-TEST:

@Mode USE-SWAP-FILE:Mode,,

and

@Mode USE-DSEG-FILE:Mode,,

The default installation option is USE-DSEG-FILE. You can change it whenever you want, just in running one of the two mode-files, providing nobody is using PF-exerciser at the same moment.

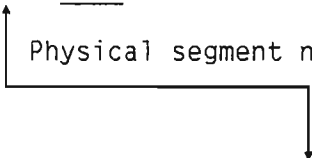
PREPARE FOR TESTING.

The second step is to get privileged. you must login to user SYSTEM , and prepare the following macro, or type directly in the console, the commands:

```
N5000: PLACE (PAGE-FAULT-TEST)PF-EXERCISER↵
N5000: LOOK-AT-REGISTER PS↵
```

```
PS : 45454000006B EXIT↵
```

Physical segment number



```
N5000: LOOK-AT-PHYSICAL-SEGMENT,310B,6B↵
PHSEG 0 310B: 0B PERMIT↵
PHSEG 0 310B: 0B BYTE↵
PHSEG 0 310B: 0B 1↵
PHSEG 0 311B: 0B EXIT↵
N5000: RUN↵
```

PRECAUTIONS.

There are quite obvious: since you will be running as a privileged process, you are allowed to execute any fatal instruction for the system. Remember always to check the SWAP-FILE size before running the program: there should be at least 3000 pages available for other processes, unless you want to run alone. If you get the message:

```
NO MORE AVAILABLE SWAPPING SPACE
```

try to place the domain using DSEG files as swapping files.

A SESSION EXAMPLE

Since you are running under TPE control, you can change your terminal type, print the date, display the program status, or even preparing some mode-files for later use. Here is one example:

@ (Logged under user SYSTEM)

@

@ND-500-MONITOR↵

N500: PF-Exerciser↵ (Macro doing all the patches)

TPE Monitor, ND-500 - Prerelease: A03 - 1987-02-10

PAGE-FAULT Exerciser for ND-500/5000 - Version A03 - 1987-12.20

-INFO- The command INFORMATIONS gives you a full program description.

Number of data segment used for the test routines : 29

You are running on a ND-5000 CPU.

Data patterns initialization.wait...

- Segment 30

Finished.

The command HELP gives you the full list of available commands

TPE>PROGRAM-STATUS↵

TPE PROGRAM STATUS: 1987.11.30 12:45:11

TPE Version.....: A03 - 1987.02.10 (Prerelease)

Console device.....: 36D / 44B

Printer device/mode.....: 36D / 44B / normal

Stop on full page.....: Off

Test program file name.....: PF-Exerciser-A03

CPU type used.....: ND-5000

Data segments available.....: 2D:30D

Data segments used for test.....: 2D:30D

Program segments used for test.....: 2D:30D

Data pattern number.....: 1

Data pattern regeneration count.....: 50000

Histogram function is turned.....: Off

TPE>RUN-ALL-TESTS↵

Sequential run is started at 1987.11.30 12:52:05

- Program segment 30

COMMENT

Counting from 2 to 30D

Sequential run finished.

Data pattern verification. Wait...

- Segment 30

Finished.

COMMENT

No error.

Random run is started at 1987.11.30 14.05.43

Test counter: 50000

COMMENT

Increments every 1000 loops.

Data pattern verification. Wait...

- Segment 30

Finished

COMMENT

No error.

*** Data patterns regeneration with next pattern number : 2

Data pattern initialisation. Wait...

- Segment 30

Finished.

Test counter: 12500

COMMENT

Here I pressed the ESCAPE-key.

Run aborted by ESCAPE at 1987.11.30 16:11:54

Number of executed test sequences : 12576

== ESCAPE ==

TPE>SERVICE-PROGRAM

PF-Serv>LOOK-AT-ESCAPE-REGISTER

Register name: P

P 02000000011

L 01000142751

PF-Serv>LOOK-AT-PROGRAM

Address: 2'11

02000000011: PCTSB

02000000013: DCTSB

02000000015: RETD

PF-Serv>VERIFY-DATA-PATTERN

COMMENT

Always verify memory before exiting.

Data pattern verification. Wait...

- Segment 30

COMMENT

No error detected

Finished.

PF-Serv>EXIT

TPE>EXIT ↵

ERROR REPORTING

Most of the traps are enabled during the RUN. If a failure deal to a trap, this one is reported and the RUN is aborted. Some informations are displayed, and the SERVICE-PROGRAM may be entered to do more investigations if needed. But, it may occur that the program or the whole system crashes, making errors impossible to be reported. Anyway, in such a case, it is demonstrated that a potential problem exists in this machine.

THE ERROR MESSAGE

When a trap is detected and reported to the test program, informations are written to the selected PRINTER device:

- The place where the trap occurred: either in the test sequence presently active, or in the main program loop (the loop which extracts the test routine parameters from the table where they are defined and which activates it).
- The test table index. (Can be used later to reproduce the error)
- The trap number and name.
- The program address (P register) when the trap occurred.

Then, it is asked if you want more information. If your answer is YES, the following is also printed:

- The expected instruction/adressing/operands type, from the table.
- The disassembled failing instruction found in program memory.
- Some registers dump from the TRAP stack: P, B.aux, L, B, R, I1, I2, I3, I4.

EXAMPLE ONE:

You just forgot to patch the physical segment to get privileged, this what will happen after some seconds:

*** ERROR, Trap has been detected in test sequence :

```
+-----+
+ Test table index.... : 00032D                                     +
+ Trap number/name.... : 33 Illegal instruction code               +
+ Program address..... : 02000000011B                             +
+ Base address (B).... : 00000000000B                             +
+-----+
```

Do you want more informations (YES or NO): YES ↵

TRAP definition: 33 Illegal instruction code

C U R R E N T L Y U S E D T E S T T A B L E :

```
Test routine address....: 02000000010B
Test instruction.....: CTSB
Addressing modes.....: Not applicable,Not applicable,Not applicable
Data type.....: Not applicable
Base address B register.: 02000000000B
```

F A I L U R E I N F O R M A T I O N S :

Instruction failing.....: PCTSB

COMMENT

Privileged one!

```
P register.....: 02000000011B
B.aux.....: 00000000000B
L register.....: 01000064704B
B register.....: 00000000000B
R register.....: 00000000000B
I1 register.....: 00000000000B
I2 register.....: 00000000000B
I3 register.....: 00000004001B
I4 register.....: 17332004000B
```

Do you want to continue the test run (YES or NO): NO ↵

TPE>

REMEMBER: Those messages will appear ONLY if the local trap handlers are enable.(Default when starting)

(See comands >DISABLE-LOCAL-TRAPS and >ENABLE-LOCAL-TRAPS.)

EXAMPLE TWO:

```
...
TPE>RUN-ALL-TESTS ↵
```

Sequential run is started at 1987.10.30 10:12:05

- Program segment 30

Sequential run finished.

Data pattern verification. Wait...

- Segment 30

Finished.

COMMENT

No error.

Random run is started at 1987.11.30 11.05.43

Test counter: 50000

Data pattern verification. Wait...

- Segment 30

Finished.

COMMENT

No error.

*** Data patterns regeneration with next pattern number : 2

Data pattern initialisation. Wait...

- Segment 30

Finished

Test counter: 12500

*** ERROR, Trap has been detected in test sequence :

```
+-----+
+ Test table index.... : 00182D                                     +
+ Trap number/name.... : 36 Protect violation                       +
+ Program address..... : 33000223764B                             +
+ Base address (B).... : 05000033776B                             +
+-----+
```

Do you want more informations (YES or NO): YES ↵

TRAP definition: 36 Protect violation

C U R R E N T L Y U S E D T E S T T A B L E :

Test routine address....: 33000223763B

Test instruction.....: DIV4

Addressing modes.....: Local Indirect, Local Indirect, Descriptor

Data type.....: Double

Base address B register.: 05000033776B

F A I L U R E I N F O R M A T I O N S :

Instruction failing.....: DIV4

P register.....: 33000223764B

COMMENT

Test routine addr. + size of ENT D, ok.

B.aux.....: 000000000000B

L register.....: 01000064704B

B register.....: 05000033776B

COMMENT

Base address B, ok.

R register.....: 000000000000B

I1 register.....: 00000004500B

I2 register.....: 00002230000B

I3 register.....: 00000004001B

I4 register.....: 17332004000B

Do you want to continue the test run (YES or NO): NO ↵

TPE>SERVICE-PROGRAM ↵

PF-Serv>VERIFY-DATA-PATTERNS ↵

Data patterns verification. Wait...

- Segment 5

Address	Expected value	Found value	Data type
0500034206	24572425424	000000000000	Source operand data

↑

COMMENT

A source pointer area has been over-written. This should never happen, and it means that something is wrong.

....

....

Finished.

8.2.1.6 FLOATING POINT TEST

GENERAL

FLOAT-TEST is a diagnostic program designed to test four arithmetic boards on ND-500/2, or ND-5000 computers. All instructions that are tested are simulated in software, i.e., the expected result computation does not use the external arithmetic on the ND-500/5000 computers. Following the computation of the expected results, the actual instruction is executed, producing the actual results. The two results (i.e., the expected and actual results) are now compared, and, in the case of unequal results, the program flags the error by giving an error message. Note, however, that only the execution of the instruction (i.e., the computation of the actual result) uses external arithmetic on the ND-500/5000.

GETTING STARTED

In addition to the program files, FLOAT-TEST comprises a data file called FUNCTEST-DATA:DATA. This data file contains a set of carefully selected operands and precomputed expected results.

To run FLOAT-TEST do the following:

Log in as user FLOAT-TEST

Sintran will respond with the following:

```
13.23.59      6 OCTOBER 1987
SINTRAN III - VSX/500  VERSION K
ENTER FLOAT-TEST↵
PASSWORD:  ↵
OK
@
```

Now, enter the ND-500/5000 monitor:

```
@ND ↵
ND-5000 MONITOR Version XXX
ND-5000:
```

Activate FLOAT-TEST by typing:

```
ND-5000: FLOAT-TEST ↵
```

TPE Monitor, ND-5000 - Prerelease: XXXXX

```
---          FLOAT-TEST          ---
--- floating hardware diagnostics ---
```

The command HELP gives you the full list of available commands
TPE>

RUNNING FLOAT-TEST

The command FUNCTION-TEST is used whenever the user wants to perform a comprehensive hardware test. The command is recommended for field testing and runs continuously until the Escape key is pressed. In the first two passes FUNCTION-TEST uses input data from the file FUNCTEST-DATA:DATA. This file contains, in addition to operands, precomputed expected results. Consequently, these passes will be very fast, and will flag severe hardware problems. From the third pass on, FUNCTION-TEST generates data using random number algorithms. System time is used to generate the random numbers.

The command is given as:

```
TPE> FUNCTION-TEST ↵
```

The program will respond by displaying the "Pass Count" number, i.e., the current pass number, and the "Test Number." The "Test Number" gives the number of the currently executing test.

All detected errors are sent to the error log (i.e., the terminal or a file), which is set to the terminal by default.

At the end of every pass, the program will display the number of completed passes and the total number of errors found during the current function test.

LOGGING ERRORS

It is possible to log errors to the terminal, to a file, or to both the terminal and a file. The SET-ERROR-LOG command is used for this purpose.

The command is given as:

```
TPE> SET-ERROR-LOG ↵  
Log Type < Term / Disk / Both / None >: Term ↵
```

If the "Disk" or "Both" option is chosen, the program will prompt for the error log file name.

```
Error Log File: FUNCTEST-ERROR:DATA ↵
```

Note that the error log file is not an ASCII file and

should, therefore, be opened using the program only. The error patterns will have pattern numbers starting from 1000.

Each failing data will be logged along with the nine immediately preceding patterns (if as many as nine preceding patterns exist). Each failing data will create one pattern in the error log file.

READING THE ERROR LOG FILE

The procedure to read and examine the error results is as follows: Take a directory listing (i.e., LIST-PATTERN-DIRECTORY) of the error log file you have chosen. The number of pattern entries will be equal to the number of errors that occurred during the last run. Read the desired pattern into memory using the READ-PATTERN-INTO-MEMORY command. Look at the header using the EDIT command. This will tell you the test number. Examine the buffer using the SHOW-PATTERN-IN-MEMORY command. It is also possible to run the buffer to regenerate the error. If required, change the formats for integer and real variables and examine the buffer again.

MISCELLANEOUS

FLOAT-TEST also gives the user the flexibility of executing specific tests. The RUN command computes the simulated results and performs a hardware test on the data existing in the memory buffer. If the memory buffer is not initialized, or if no pattern is read from the specified file, RUN will give the following message:

No valid data in the buffer ..

The RUN command will display the test number of the currently executing test. In the case of errors, appropriate error messages will be displayed.

Test number XXXD

If a test ends without errors, the program prompt will be redisplayed.

For further information on FLOAT-TEST, the user is advised to look in the FLOAT-TEST user's guide.

8.2.1.7 OCTOBUS

The Octobus Test Program runs stand alone in the ND-100, controlled by the TPE-monitor. The program must be downloaded from a floppy. Its basic functions are:

- Tests the Octobus controller in the ND-100 Line Driver.
- Finds the Octobus configuration.

OCTOBUS TEST COMMANDS

The commands available for the user are:

- SET-PARAMETERS <loop> <abort> <supress> <error report level>
- LIST-HARDWARE-CONFIGURATION
- RUN <test sequence>

SET-PARAMETERS

The user may set parameters which decide the behaviour of the RUN command. The parameters are listed below, and their default values are shown in paranthesis.

- Loop mode (No)
- Abort mode (Yes)
- After how many errors (10)
- Supress error messages (No)

The user may also specify the appearance of the error messages. This is done by answering 'Yes' to the question 'Define error reporting level'. The following questions must then be answered (default values in paranthesis):

- Controller number (Yes)
The number of the failing Octobus Controller.
- Hardware device number (Yes)
The hardware device number for the failing Octobus controller.
- Type of error (Yes)
Specify what is wrong.

- Error information (Yes)
This information depends on type of error. It may be register contents, or found and expected values.
- Decoding of status (Yes)
Decoding of register contents shown under 'Error information'.

LIST-HARDWARE-CONFIGURATION

This command will find present Octobus controllers, and all the Octobus stations present for each controller. These stations may be the MF bus controller, the ACCP and/or the Domino modules. Before returning to TPE, a configuration table is shown. This table consists of, from left to right:

- Octobus controller number.
- Octobus hardware device number.
- Receive ident code (level 13).
- Transmit ident code (level 13).
- The Octobus controller's station number.
- Stations seen by the Octobus controller.

RUN

Using this command, you can run all tests (default), only one test or a sequence of tests. The available tests are:

1. Check transmit - receive loop.
2. Loop all possible patterns.
3. Check receive FIFO length.

Test 1:

The controller will send one byte to itself. The transmit and receive parts are tested.

Test 2:

The controller will send all possible bit patterns to itself and compare the transmitted and received patterns. It is tested whether the controller is able to transmit and receive all possible bit patterns.

Test 3:

The controller will send several bytes to itself, and detect when the receive FIFO is full. The size of the

receive FIFO is checked.

The program will always test the following items, without user intervention.

- Status registers.
- Interrupt and ident codes.
- The combination RFT (ready for transfer), IE (interrupt enabled) and ID (interrupt detected).

LOGGING ERRORS

If you should detect any errors on the CPU and decide to return the CPU to the ND repair center, please include a file containing the error messages from the Octobus test where the errors were detected.

Output from the Octobus test can be assembled on a file by using the TPE commands :

```
>SET-PRINTER-FILE  
>SET-PRINTER-MODE
```

The following example illustrates how to direct output from The Octobus test to the terminal and to the file LOG-FILE:SYMB at the same time.

```
>SET-PRINTER-FILE LOG-FILE:SYMB  
>SET-PRINTER-MODE DUPLICATED
```

8.2.2 VERIFICATION PROGRAMS

8.2.2.1 SUPER

This section describes SUPER and how to run it.

TEST DESCRIPTION

This test verifies that the ND-5000 CPU calculates correctly. The SUPER test uses input from several files.

The test verifies the Floating Arithmetic and the Slice.

The SUPER source files can also be used as input files to the Fortran compiler to verify the compilation.

RUNNING SUPER

The results are compared against a facit file. If the result is correct, "OK" is written after every printout. In case of an incorrect result, "ERR" and the actual and expected result is written.

The results are printed out as octal numbers (22 digits).

ENTER nd5000-supertest↵

PASSWORD: ↵

OK

@nd↵

ND-500/5000 MONITOR Version I00 (preliminary) 87. 6.16 / 87. 9.17

ND-5000: run-super↵

SPECIFY FIRST SUPER TESTRUN..(ETHYL,HEXAT,BR-60)..(ETHYL)?ETHYL↵

SPECIFY CPU TYPE.(S = ND-5000, X = ND-5000 W/570 FP)..(S)?S↵

N500: CC *****

N500: CC * SUPER TESTRUN E T H Y L *

N500: CC * RUNNING ON ND-5000 *

N500: CC *****

N500: PLACE SUPER

N500: RUN

THE POINT GROUP OF THE MOLECULE IS ...CN

THE ORDER OF THE PRINCIPAL AXIS IS ... 2

NUMBER OF SUBGROUPS : 2

NUMBER OF SUBGROUPS : 2

ITERATION NO.: 2, TOTAL ENERGY : -78.36878 1407162746421037707250B OK

ITERATION NO.: 3, TOTAL ENERGY : -77.92968 1407157337755332721463B OK

ITERATION NO.: 4, TOTAL ENERGY : -77.94551 1407157440624367307073B OK

ITERATION NO.: 5, TOTAL ENERGY : -77.94641 1407157444370167223431B OK

ITERATION NO.: 6, TOTAL ENERGY : -77.94647 1407157444604203305325B OK

ITERATION NO.: 7, TOTAL ENERGY : -77.94648 1407157444620103310474B OK

ITERATION NO.: 8, TOTAL ENERGY : -77.94648 1407157444621160511455B OK

STOP 0

N500: TIME-USED

Time and date: 10.43.14 14 October 1987

Entered ND-500/5000 at 10.38.26 14 October 1987

Total time logged on ND-500/5000 monitor:..... 4 min 48.0 s

ND-500/5000 CPU time used in last run:..... 4 min 16.3 s

ND-100 CPU time used by ND-500/5000 process in last run: 1.3 s

Total ND-500/5000 CPU time used:..... 4 min 16.3 s

Total ND-100 CPU time used by ND-500/5000 process:..... 1.3 s

Total ND-100 CPU time used:..... 3.4 s

N500: PLACE SUPER

N500: RUN

THE POINT GROUP OF THE MOLECULE IS ...CN

THE ORDER OF THE PRINCIPAL AXIS IS ... 2

NUMBER OF SUBGROUPS : 2

NUMBER OF SUBGROUPS : 2

ITERATION NO.:	2, TOTAL ENERGY :	-233.71204	1410646662204174606140B	OK
ITERATION NO.:	3, TOTAL ENERGY :	-231.75253	1410637005133470723014B	OK
ITERATION NO.:	4, TOTAL ENERGY :	-231.55939	1410636171507464034650B	OK
ITERATION NO.:	5, TOTAL ENERGY :	-231.54454	1410636133162237160532B	OK
ITERATION NO.:	6, TOTAL ENERGY :	-231.54566	1410636135412742014150B	OK
ITERATION NO.:	7, TOTAL ENERGY :	-231.54691	1410636140042734114506B	OK
ITERATION NO.:	8, TOTAL ENERGY :	-231.54746	1410636141152101374126B	OK
ITERATION NO.:	9, TOTAL ENERGY :	-231.54767	1410636141506166540142B	OK
ITERATION NO.:	10, TOTAL ENERGY :	-231.54775	1410636141623773156274B	OK
ITERATION NO.:	11, TOTAL ENERGY :	-231.54777	1410636141656431475040B	OK
ITERATION NO.:	12, TOTAL ENERGY :	-231.54778	1410636141667355261060B	OK
ITERATION NO.:	13, TOTAL ENERGY :	-231.54779	1410636141672332465572B	OK
ITERATION NO.:	14, TOTAL ENERGY :	-231.54779	1410636141673320666142B	OK

STOP 0

N500: TIME-USED

Time and date:	13.32.42	14 October 1987
Entered ND-500/5000 at	10.38.26	14 October 1987
Total time logged on ND-500/5000 monitor:.....	2 h 54 min	15.5 s
ND-500/5000 CPU time used in last run:.....	2 h 48 min	47.0 s
ND-100 CPU time used by ND-500/5000 process in last run:		11.3 s
Total ND-500/5000 CPU time used:.....	2 h 53 min	3.4 s
Total ND-100 CPU time used by ND-500/5000 process:.....		12.7 s
Total ND-100 CPU time used:.....		15.6 s

N500: PLACE SUPER

N500: RUN

THE POINT GROUP OF THE MOLECULE IS ...C1
 THE ORDER OF THE PRINCIPAL AXIS IS ... 0
 NUMBER OF SUBGROUPS : 1

ITERATION NO.:	2, TOTAL ENERGY :	-2779.00049	1414266600037670724601B	OK
ITERATION NO.:	3, TOTAL ENERGY :	-2779.00056	1414266600044777211317B	OK
ITERATION NO.:	4, TOTAL ENERGY :	-2779.00059	1414266600046713400555B	OK
ITERATION NO.:	5, TOTAL ENERGY :	-2779.00061	1414266600047630474610B	OK
ITERATION NO.:	6, TOTAL ENERGY :	-2779.00062	1414266600050255310256B	OK
ITERATION NO.:	7, TOTAL ENERGY :	-2779.00062	1414266600050545302307B	OK

```

ITERATION NO.: 8, TOTAL ENERGY :    -2779.00063  1414266600050753572716B  OK
ITERATION NO.: 9, TOTAL ENERGY :    -2779.00063  1414266600051114777655B  OK
ITERATION NO.: 10, TOTAL ENERGY :    -2779.00063  1414266600051227022437B  OK
ITERATION NO.: 11, TOTAL ENERGY :    -2779.00063  1414266600051316126034B  OK
ITERATION NO.: 12, TOTAL ENERGY :    -2779.00063  1414266600051370063465B  OK
ITERATION NO.: 13, TOTAL ENERGY :    -2779.00063  1414266600051427411761B  OK
ITERATION NO.: 14, TOTAL ENERGY :    -2779.00063  1414266600051457335247B  OK
STOP 0
N500: TIME-USED

```

```

Time and date:    14.53.50   16 October 1987
Entered ND-500/5000 at 10.38.26   14 October 1987
Total time logged on ND-500/5000 monitor:..... 52 h 15 min 24.5 s
ND-500/5000 CPU time used in last run:..... 49 h 16 min 2.9 s
ND-100 CPU time used by ND-500/5000 process in last run: 24.0 s
Total ND-500/5000 CPU time used:..... 52 h 9 min 6.3 s
Total ND-100 CPU time used by ND-500/5000 process:..... 36.7 s
Total ND-100 CPU time used:..... 40.4 s

```

ND-5000: exit

LOGGING ERRORS

If you need to save the test output in a disc file, do the following:

- Log into user TELEFIX. Type "TELEFIX" to start TELEFIX-LOCAL.
- Type the following commands:

```

CONNECT,,
OPEN-LOG-FILE SUPER
MANUAL-MODE

```

The effect of these three commands is to connect you to a virtual terminal, and all traffic on this terminal will be in the file SUPER:LOGS.

- Type the <esc> key, and log into user SYSTEM.
- Type the commands described earlier to start the ND-5000 monitor and SUPER. When the test is finished, log out.
- Type <ctrl>@ to return to TELEFIX. You can now exit from TELEFIX, or type LOG-ANALYZER to look at the log file.

8.2.2.2 SIBAS TEST

The SIBAS test is a FORTRAN program, named SIBBIG, which exercises various SIBAS functions. It is used to verify that the SOLO and TUTTI instructions are functioning properly.

To run the test without using TELEFIX, log in as user SYSTEM. Type "MODE (SIBAS-TEST)INIT-SIBAS:MODE,," to initialize the files. Then type "ND" to start the ND-5000 Monitor. Type "SIBBIG" to start the test program. You will see the following output:

```
=====
==== S I B A S - T E S T ====
=====
```

```
This program  SIBBIG  loops and
find, delete, modify persons, jobbs
and connected reports.
```

```
THE PROGRAM WILL TERMINATE
AND THE DATABASE BE CLOSED
IF YOU TYPE  B  !
```

You must then answer the following questions:

```

Give max-time in minutes:nn↵
SIBAS system no(0-23): 1↵
Remote-line no( 0-4): 0↵
REDUSED PRINT-OUT= 0/ FULL = 1: 0↵

DATABASE FORDB OPENED AT 7 50 6 11 2 3 1987
ERASED 0 RECORDS FROM PERSON
ERASED 251 RECORDS FROM JOBB
REALM JOBB FILLED UP
STORED 1208 RECORDS INTO JOBB
486 RECORDS STORED AT 35 27 8 11 2 3 1987
DATABASE FORDB CLOSED AT 38 27 8 11 2 3 1987
DATABASE FORDB OPENED AT 47 36 8 11 2 3 1987
REALM PERSON OPENED
MODIFIED 486 PERSONS AT 10 12 9 11 2 3 1987
DATABASE FORDB CLOSED AT 11 12 9 11 2 3 1987
MODIFIED 486 PERSONS AT 20 53 9 11 2 3 1987
DATABASE FORDB CLOSED AT 21 53 9 11 2 3 1987
MODIFIED 486 PERSONS AT 0 50 10 11 2 3 1987
DATABASE FORDB CLOSED AT 2 50 10 11 2 3 1987

.
.
.
MODIFIED 486 PERSONS AT 6 59 37 12 2 3 1987
DATABASE FORDB CLOSED AT 8 59 37 12 2 3 1987
MAX TIME REACHED!!!
NUMBER OF SOPDB= 119 SCLDB= 119
NUMBER OF SRRLM= 120 SFRLM= 1
NUMBER OF STORE= 2181 SGET = 114814
NUMBER OF SMDFY= 57348 SRASE= 251
NUMBER OF SRFSM= 118 SRNSM= 57348
NUMBER OF SRNIS= 57348 SFEBL= 118
NUMBER OF SFTCH 118 TOTAL = 290003
PROGRAM STOPPED AT 7 1 38 12 2 3 1987
PROGRAM STARTED AT 8 50 6 11 2 3 1987

```

You can then type "EXIT" to leave the ND-500 Monitor.

LOGGING ERRORS

If you need to save the test output in a disc file, do the following:

- Log into user TELEFIX. Type "TELEFIX" to start TELEFIX-LOCAL.
- Type the following commands:

```

CONNECT,,
OPEN-LOG-FILE SIBAS
MANUAL-MODE

```

The effect of these three commands is to connect

you to a virtual terminal, and all traffic on this terminal will be in the file SIBAS:LOGS.

- Type the <esc> key, and log into user SYSTEM.
- Type the commands described earlier to start the ND-5000 monitor and SIBBIG. When the test is finished, log out.
- Type <ctrl>@ to return to TELEFIX. You can now exit from TELEFIX, or type LOG-ANALYZER to look at the log file.

8.2.2.3 CXTEST

CXTEST is a COBOL program which tests BCD arithmetic.

CXTEST requires approximately five minutes to run. To run it, log into user N500-USER-TEST, and enter the following commands:

```
@nd
ND-500/5000 MONITOR  Version I00 (preliminary) 87. 6.16 / 87. 9. 1
ND-5000: cxtest

MODIFIED FOR ND-500CX          FS-TSNS 7/4-86
NORSK DATA TEST-PROGRAM CHECKING ADD/MULTIPLY
DIVIDE AND SUBTRACT IN FOLLOWING FORMATS:
  DISPLAY ,COMP-1(=BINARY) AND COMP-3 (=BCD).
THE LOOPS START WITH NEGATIVE NUMBERS
THAT ARE STEPPED TO POSITIVE NUMBERS
MULTIPLY/ DIVIDE IS BY 2 FOR EASY CHECKING
THE PROGRAM ENDS BY RUNNING MOVE AND IF-TEST
USING CHARACTER-STRINGS. NOTE! THE PROGRAM
WILL ABORT AFTER ILLEGAL INSTRUCTION IF
YOUR COMPUTER HAS NO COMM. INSTRUCTION-SET!
START-DATE: 870923 AT 1438 0,CLOCK
MULTIPLY- TEST STARTED
DIVIDE- TEST STARTED
SUBTRACT TEST STARTED
TEST MOVE AND IF-STATEMENT STARTED

ND-5000:
```

LOGGING ERRORS

If you need to save the output from this test to a disc file, following the directions given at the end of the section on the SIBAS test, but specify USER as the log file instead of SIBAS.

8.2.2.4 INVERSE-MATRIX, WHETSTONE, DHRYSTONE AND LACOURT TESTS

These tests are FORTRAN number crunchers, which can fail on certain types of faulty hardware.

The Inverse Matrix test requires approximately twenty minutes to run. The Lacourt test requires approximately 3 minutes to run. The Whetstone and Dhrystone tests each require less than one minute.

To run each of these tests, log in as user ND5000-USER-TEST, and enter the following commands:

```
@nd
ND-500/5000 MONITOR Version I00 (preliminary) 87. 6.16 / 87. 9. 1
ND-5000: matrix↵
NO OF LOOPS BEFORE BREAK: 20↵
LOOP IF FAILS PR. 1000 FADDS GREATER THAN: ↵
```

```
RELATIVE PRECISION = .100000000000000D-09
```

ITERATION	1000	PRECISION OF MATRIX =	.19549851684598D-12
ITERATION	2000	PRECISION OF MATRIX =	.48272985080155D-12
ITERATION	3000	PRECISION OF MATRIX =	.72126665662958D-12
ITERATION	4000	PRECISION OF MATRIX =	.80071571595397D-12
ITERATION	5000	PRECISION OF MATRIX =	.93555013471641D-12
ITERATION	6000	PRECISION OF MATRIX =	.11567864831814D-11
ITERATION	7000	PRECISION OF MATRIX =	.12109209190925D-11
ITERATION	8000	PRECISION OF MATRIX =	.11433552182628D-11
ITERATION	9000	PRECISION OF MATRIX =	.10520891512486D-11
ITERATION	10000	PRECISION OF MATRIX =	.93997919324727D-12
ITERATION	11000	PRECISION OF MATRIX =	.85242659051339D-12
ITERATION	12000	PRECISION OF MATRIX =	.90775583786997D-12
ITERATION	13000	PRECISION OF MATRIX =	.10523613832210D-11
ITERATION	14000	PRECISION OF MATRIX =	.11755466855677D-11
ITERATION	15000	PRECISION OF MATRIX =	.12983343148383D-11
ITERATION	16000	PRECISION OF MATRIX =	.14130482489572D-11
ITERATION	17000	PRECISION OF MATRIX =	.15680331301471D-11
ITERATION	18000	PRECISION OF MATRIX =	.18678268808180D-11
ITERATION	19000	PRECISION OF MATRIX =	.22009239157814D-11
ITERATION	20000	PRECISION OF MATRIX =	.26060834777664D-11

```
END OF RUN
```



```
ND-5000: whetstone↵
3125.0 WHETSTONE KIPS
```

```
ND-5000: dhrystone↵
```

```
*****
```

```
-- Start Timer --
```

```
*****
```

```
*****
```

```
-- Stop Timer --
```

```
*****
```

```
Dhrystone time for 100000 passes = 15
```

```
This machine benchmarks at 6553 dhrystones/second
```

```
ND-5000: lacourt↵
```

```
BEGINNING OF JOB TEST1
```

```
START ONLY RETURN END
```

```
STOP ONLY RETURN END
```

```
START TANSFER A BUFFER
```

```
1 2 3 4 5 6 7 8 9 10
```

```
11 12 13 14 15 16 17 18 19 20
```

```
21 22 23 24 25 26 27 28 29 30
```

```
.
```

```
.
```

```
.
```

```
START MOMENTUM
```

```
-.194922409058E+02
```

```
STOP MOMENTUM
```

```
END OF JOB TEST3
```

```
8.68 SECONDS
```

```
16.08 SECONDS
```

```
12.92 SECONDS
```

```
13.90 SECONDS
```

```
ND-5000: ex↵
```

LOGGING ERRORS

If you need to save the output from these tests to a disc file, follow the directions given at the end of the section on CXTEST.

8.2.2.5 LIBTEST

LIBTEST is a program designed to test the mathematical library functions, but is also relevant as a verification program for the hardware. The functions are tested against their inverse function using the fact that $\text{SIN}(\text{ASIN}(X))=X$, $\text{SQRT}(X)*\text{SQRT}(X)=X$, $\text{EXP}(\text{LOG}(X))=X$, etc.

The function being tested is listed under column F in the output. The inverse function is under column G. The INTERVAL and STEP columns specify the values for which the functions are tested.

The maximum value of the errors found appear in the MAX ERROR column. The value of X for which this value occurred is in the CORR. X column,

The value in the ERROR RMS column is the mean root square calculated for all of the errors on X values.

The right-most column is the execution time for the function, in microseconds. Note that this value will be different each time the test is executed. The total execution time for this test is less than one minute.

To run LIBTEST, log in as user N500-USER-TEST, and enter the following commands:

@nd

ND-500/5000 MONITOR Version I00 (preliminary) 87. 6.16 / 87. 9. 1

ND-5000: libtest

----- S I N G L E P R E C I S I O N -----

F	G	INTERVAL	STEP	MAX ERROR	CORR. X	ERROR RMS	EX.TIME FOR
SIN	ASIN	-1.0 1.0	.1E-02	.2E-06	.5E+00	.1E-08	23
ASIN	SIN	-1.6 1.6	.2E-02	.1E-04	.2E+01	.1E-06	21
COS	ACOS	-1.0 1.0	.1E-02	.5E-06	-.6E+00	.3E-08	29
ACOS	COS	.0 3.1	.2E-02	.6E-05	.3E+01	.9E-08	22
TAN	ATAN	-9.010.0	.1E-01	.2E-05	.1E+02	.1E-07	25
ATAN	TAN	-1.6 1.6	.2E-02	.2E-06	.1E+01	.9E-09	20
SQRT	X**2	.010.0	.5E-02	.0E+00	.1E+01	.0E+00	9
X**2	SQRT	.010.0	.5E-02	.2E-06	.8E+01	.3E-08	2
EXP	LOG	.010.0	.5E-02	.3E-06	.3E+01	.2E-08	22
LOG	EXP	-9.010.0	.1E-01	.2E-06	.1E+01	.5E-09	30

----- D O U B L E P R E C I S I O N -----

F	G	INTERVAL	STEP	MAX ERROR	CORR. X	ERROR RMS	EX.TIME FOR
SIN	ASIN	-1.0 1.0	.1E-02	.3E-16	-.9E+00	.2E-18	70
ASIN	SIN	-1.6 1.6	.2E-02	.8E-14	.2E+01	.5E-17	82
COS	ACOS	-1.0 1.0	.1E-02	.8E-16	-.6E+00	.5E-18	84
ACOS	COS	.0 3.1	.2E-02	.4E-14	.3E+01	.3E-17	83
TAN	ATAN	-9.010.0	.1E-01	.3E-15	.9E+01	.2E-17	74
ATAN	TAN	-1.6 1.6	.2E-02	.3E-16	-.1E+01	.2E-18	69
SQRT	X**2	.010.0	.5E-02	.0E+00	-.1E+01	.0E+00	23
X**2	SQRT	.010.0	.5E-02	.6E-16	.8E+01	.6E-18	6
EXP	LOG	.010.0	.5E-02	.6E-16	.7E+01	.6E-18	59
LOG	EXP	-9.010.0	.1E-01	.5E-16	.1E+01	.1E-18	93

ND-5000: ex

LOGGING ERRORS

If you need to save the output from these tests to a disc file, follow the directions given at the end of the section on CXTEST.

8.3 MULTIJOB TELEFIX

How to use the Multijob Telefix is described in ND-5000 Test Macro Program Manual.

Following is a very short descriptin of only the startup of the disc pack..(not implented yet in this manual)..

8.4 UTILITY PROGRAMS

8.4.1 N500X-MESSAGE

As the error messages from the ND-5000 Monitor (using also S III WM500) has been revised to give more complete messages this program is no longer that useful as earlier on the ND-500 disc test pack, however a short start up procedure is given below: (not implented yet in this manual)..

PROGRAM DESCRIPTION

When using this utility program, you can read and decode the message buffer for a given ND-5000 process.

Each message buffer contains 145B (101D) 16 bits entries. The first 7 (0-6) of these are the message header and these locations are always decoded.

Entry no. 6 contains which function (MICFUNC) to perform and entry no. 2 contains the STATUS. Depending on the contents of these two locations, the rest of the message is either decoded or written as octal 16 bits values. The program has a set of commands:

>>EXIT:

Closes the file and exits the program.

>>HELP:

Prints a list of the commands.

>>LIST-EXECUTION-QUEUE:

Lists the execution queue.

>>OCTAL-DUMP:

Prints the message buffer as octal values, except for the message header.

>>READ-MESSAGE:

Decodes the specified message buffer. If there is no trap or Monitor Call, the program will print the message as octal values.

>>SAVE-MESSAGE:

Saves the specified message buffer to the file SAVE-MESSAGE:DATA.

>>WHO:

Lists out the users currently logged on the ND-5000.

The file SAVE-MESSAGE:DATA is not closed before the command EXIT is performed. Several message buffers can therefore be saved on the file.

RUNNING N500X-MESSAGE

```

ENTER system↵
PASSWORD:.....↵
OK
@n500x-message↵

```

```

*****
***  N D 5 0 0 X - MESSAGE DECODER  pre.. 20.05 ***
*****

```

Status on CPU type..: 5200 CPU number..: 6155

```

Operating syst.: SINTRAN III VSX/500 - K
Revision.....: 105000
Local CPU.....: ND110/CX-16PITS - 32 Fp
Mic.program ver: 11D
Main CPU.....: ND5000
Mic.program ver: 13213D
System part....: 87. 9.17 Rev. K05
Swapper.....: 87.07.03
Local memory...: 4096D Kbytes.
Shared memory..: 16384D Kbytes.
Register block.: 00000444000B ==> phys.ND5000 addr.
Phys.Seg.Table.: 00000644000B ==> phys.ND5000 addr.

```

```

>>read↵
Give process number.(-1=SW):8↵

```

Dump of Message buffer for process: 8

```

Link.....: 177777B
Link.....: 177777B
Status....: 000003B          ==> Answer/N500 finished
Sender....: 000010B          ==> Process no.
Receiver..: 000010B          ==> Process no.
Prev.link.: 000001B          ==> Previous message
Micfunc...: 000023B          ==> Start ND500/Trap or mon.call

```

** TRAP MESSAGE.....: Instruction Sequence Error

```

Trapping P.....: 01000065610B
Restart P.....: 01000065610B
Trap number.....: 043B ==> 35D
General Buffer pointer..: 040 050000B

```

```

>>ex↵
--> Exit

```

8.4.2 TEST FUNCTIONS IN ND-5000/MF FIRMWARE

8.5 MF BUS TEST AND MAINTENANCE PROGRAM

All the registers on the Port module, the RAM module and the Controller module are programmed from the MFB Test and Maintenance Program. This program appears at the console terminal, which can be connected to the controller module.

8.5.1 CONNECTING THE CONSOLE TERMINAL TO THE CONTROLLER

Large cabinet: A console can be connected to the MFB-controller via the plug board (Print 5234) located in the backplane in the rear side of the controller.

ND-5000, REAR VIEW

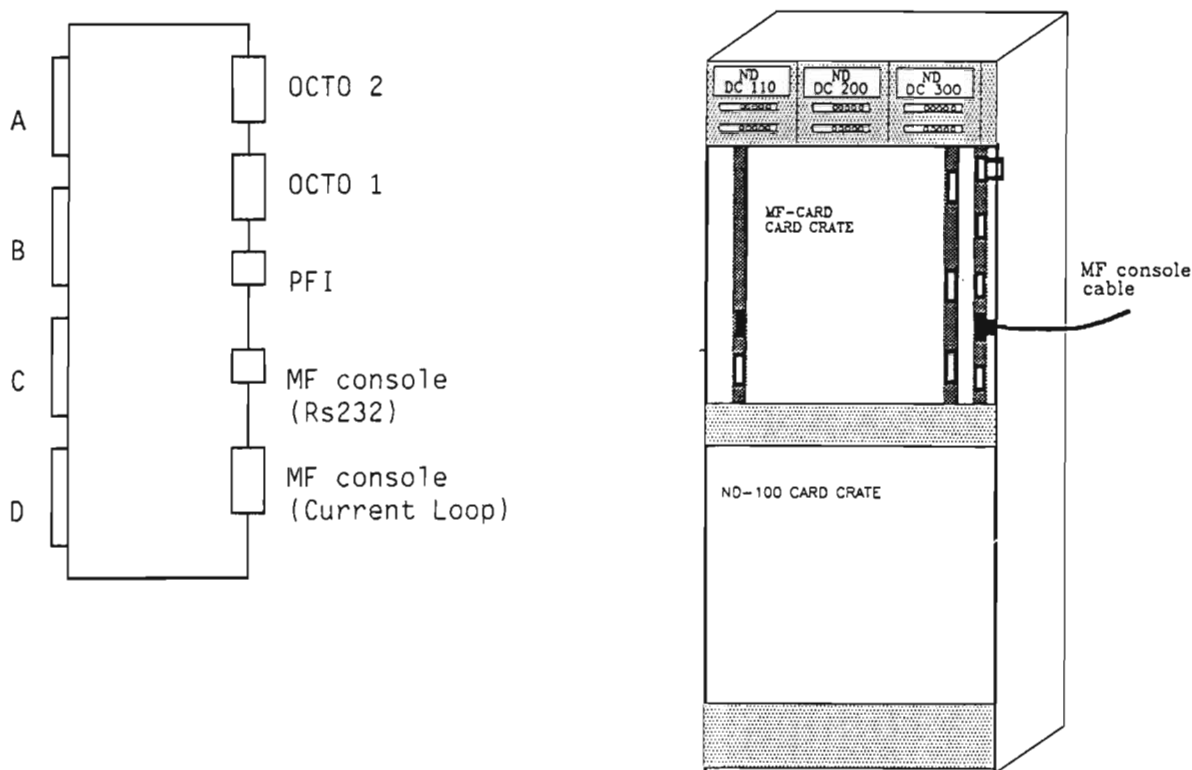


Figure 18. Connecting the MF console on the large cabinet

"Old" cabinet: A console can be connected to the MFB-controller via the plug panel marked CONS.

8.5.2 DESCRIPTION OF THE MOST USEFUL COMMANDS

This section will give a short description of some of the most important commands. For further information, see the manual:

"MPM 5 Technical description"

These commands will be described:

- INITIATE-EEPROM
- CONFIGURATE-SLOT
- LIST-CONFIGURATION
- TEST-MEMORY
- SYNDROME-TEST
- AUTOINITIATE-BANK
- LIST-OCTOBUS-STATION
- OCTOBUS-SELFTEST
- OCTOBUS-FACILITIES
 - ACCESS-OCT-REG <Function> <Value>
 - LIST-SUBPROC-TABLE
 - OCT-CONTROL-FUNCTION <Function> <Retry>
 - OCT-TRANSMIT-STATUS
 - OCTOBUS-DRIVER <Function>
 - READ-OCTOBUS-RECEIVE
 - TRANSMIT-OCTOBUS <Destination> <C(0/1)> <B(0/1)>
<No of bytes (5 max)> <Byte 1> <Byte 5>

NOTE

Two of the commands described in this section must NOT be used unless you are going to update the ND-5000 CPU:

INITIATE-EEPROM
CONFIGURATE-SLOT (for the CPU)

You need a special tool for this updating:

For DBC: Partno. 350156
For MF-contr: Partno. 350157
See description on page 37

INIATE-EEPROM:

Parameters: Slot number

Date (Year, Month, Day (YYMMDD))

Old contents will be lost - continue?(Yes/No)

CONFIGURATE-SLOT:

Parameter: Slot number

With this command, it is possible to configure the MFB system. See the example on page 215

LIST-CONFIGURATION:

Parameter: Slot number

This command lists the contents of a specified slot in the bank. See the example on page 222

TEST-MEMORY:

Parameters: Start Block (128 Kbyte)

Number of blocks

Number of runs

Suppress error report? (Yes/No)

This command performs a statical pattern test on the MFB memory.

EXAMPLE:

```
>TEST-MEMORY<CR>
Start-blk (128 kb):0<CR>
No. of blk:40<CR>
No. of run:1<cr>
Suppress. err. report: yes<CR>
```

SYNDROME-TEST:

This command performs a test on the logic circuitry that is used to detect errors in memory.

```
>Syndrom<CR>
SLOT      03 1-BIT CORRECTION - OK -
           2-BIT DETECTION  - OK -
```

AUTOINITIATE-BANK:

After this command, the bank will automatically be initiated with parameters found in the EEPROM.

The MFB is now ready for normal operation and the entered parameters are stored in the EEPROM in the backwiring.

The following commands are Octobus commands:

- LIST-OCTOBUS-STATION
- OCTOBUS-SELFTEST
- OCTOBUS-FACILITIES
 - ACCESS-OCT-REG <Function> <Value>
 - LIST-SUBPROC-TABLE
 - OCT-CONTROL-FUNCTION <Function> <Retry>
 - OCT-TRANSMIT-STATUS
 - OCTOBUS-DRIVER <Function>
 - READ-OCTOBUS-RECEIVE
 - TRANSMIT-OCTOBUS <Destination> <C(0/1)> <B(0/1)> <No of bytes (5 max)> <Byte 1> <Byte 5>

LIST-OCTOBUS-STATION

Present a list showing the Octobus stations present in this ring.

OCTOBUS-SELFTEST

The Octobus controller sends different bit patterns to itself on the Octobus. The patterns transmitted and received are compared.

The transmit and receive parts are tested. No interrupt check.

ACCESS-OCT-REG

The user has direct access to the Octobus registers. The functions are:

- 0 - Read receive data
- 1 - Not used

- 2 - Read receive status
- 3 - Write receive control
- 4 - Not used
- 5 - Write transmit data
- 6 - Read transmit status
- 7 - Write transmit control

OCT-CONTROL-FUNCTION

The user is able to write the controller's receive- and transmit-control register, without knowing the outlook of these registers. In this way, the command operates on a higher level than 'ACCESS-OCT-REG'. Following functions exists:

- 0 - Read all mode
- 1 - Clear receiver
- 2 - Clear transmitter
- 3 - Clear transmit fifo
- 4 - Remove master
- 5 - Set number of retries on transmit

OCT-TRANSMIT-STATUS

The user is presented the content of the transmit status register.

OCTOBUS-DRIVER

This command gives the user access to the Octobus 680XX driver. Following functions are defined:

- 1 - Send multibytes
- 2 - Broadcast multibytes
- 3 - Send kick
- 4 - Send ident
- 5 - Send emergency (Not implemented)
- 6 - Read transmit status
- 7 - Connect kick
- 8 - Connect ident
- 9 - Connect OMD
- 10 - Access octobus registers (Not implemented)

Verifies that the transmit and receive parts, the Octobus driver and the interrupts work correctly. For further details about the different functions, see the 'Octobus driver programming guide' (written by DVT - 15. oct 1986).

LIST-SUBPROC-TABLE


The message/s received at a specified OMD number, is presented.

READ-OCTOBUS-RECEIVE

The command present one byte from the receive fifo (Info), together with the transmitter (Source) and the status (Status). Following status may be given:

000 - The byte read is valid data.
002 - The byte read is not valid data.

Verify that the receive part of the controller works correctly. Bypass the Octobus driver.

Before using this command, one must disable interrupt on channel 6 by the command 'DISABLE-INTERRUPT <Channel no>'. The interrupt is enabled by the command 'ENABLE-INTERRUPT <Channel no>'.


TRANSMIT-OCTOBUS

The user is able to transmit bytes on the Octobus. Verify that the transmit part of the controller works correctly. Bypass the Octobus driver.

Transmit and receive messages on Octobus

Examples:

1. Communication with use of Octobus driver:
Verify that the transmit and receive parts of the Octobus controller, and the Octobus driver (software and interrupts), work correctly.

To send a multibyte message on Octobus:

(8BUS)@OCTOBUS-DRIVER <CR>

```

Function:1 <CR>
Destination :<Dest station no> <CR>
Omd:<Dest OMD no (0-15)> <CR>
Own omd :<Own OMD no (0-15)> <CR>
Msg content (text string) :Hallo-world <CR>
- Ok - Msg_id: 00000400001B
(8BUS)@

```

To receive a multibyte message on Octobus:

```

(8BUS)@OCTOBUS-DRIVER <CR>
Function:9 <CR>
Omd:X <CR>
Action :<CR>
Flag : <CR>
Buffer size :70 <CR>
- OK -
(8BUS)@LIST-SUBPROC-TABLE <CR>

Omd:X <CR>

OmdOX
Source AA Message size 011
HALLO-WORLD
Omd:-1 <CR>
(8BUS)@

```

Before any message can be received, an OMD must be connected. If one wants the Octobus controller to send a multibyte message to itself, the commands must be done in following order:

1. OCTOBUS-DRIVER 9 (Connect OMD no. X)
2. OCTOBUS-DRIVER 1 (Send message to OMD no. X)
3. LIST-SUBPROC-TABLE (List content of OMD no. X)

2. Communication with bypass of the Octobus driver:

Verify that the transmit and receive parts of the Octobus controller work correctly.

To send bytes on Octobus:

```

(8BUS)@TRANSMIT-OCTOBUS <CR>
Destination:<Dest station no> <CR>
C(0/1):0 <CR>
B(0/1):0 <CR>
No of bytes (5 max):3 <CR>
Byte 1:<Value of byte 1>
Byte 2:<Value of byte 2>
Byte 3:<Value of byte 3>
(8BUS)@

```

To read Octobus receive fifo:

```

>DISABLE-INTERRUPT 6 <CR>
>OCTOBUS-FACILITIES <CR>
(8BUS)@READ-OCTOBUS-RECEIVE <CR>
Source <Source station number>

```

```
Info    <Value of first byte in receive fifo>  
Status  <000/002 - valid data/not valid data>  
(8BUS)@EXIT  
>ENABLE-INTERRUPT 6 <CR>
```

Before any message can be received, the interrupt on channel 6 must be disabled. If one want the Octobus controller to send bytes to itself, the commands must be done in following order:

1. DISABLE-INTERRUPT (Disable interrupt on channel 6)
2. TRANSMIT-OCTOBUS (Send maximum 5 bytes)
3. READ-OCTOBUS-RECEIVE (Read one byte from fifo)
4. ENABLE-INTERRUPT (Enable interrupt on channel 6)

8.5.3 CONFIGURATING THE MF-SYSTEM

This section describes how to configurate a MF-system:

1. Configuring a system by:
 Setting the switches on the memory cards in the ND-100 card crate.
 Configuring the MF-system by using the command CONFIGURATE-SLOT.
2. Upgrading the system by installing a ND-120/CX-4Mb.
3. Configuring the upgraded system.
4. Check the configuration by using the command LIST-CONFIGURATION.

The basis configuration in the example is:

ND-5000
ND-110/CX
2 x 2 Mb local ND-100 memory
Ethernet controller (512 Kb)
16 Mb shared memory

1. Set the switches in the ND-100 card crate as shown below:

Module	Lower limit switchsetting	Upper limit display	Ethernet bank no.
1st 2Mb board	000	100	020
2nd 2Mb board	100	200	
Ethernet			

Configure the MF-system by using the command CONFIGURATE-SLOT.

>CONFIGURATE-SLOT<CR>

MODULES IN THIS BANK

```
=====
SLOT      01 : MF-BUS CONTROLLER STANDARD
SLOT      02 : PORT TWIN 16 BIT (PRINT 5155)
SLOT      03 : DYNAMIC RAM - 4 MB
SLOT      04 : DYNAMIC RAM - 4 MB
SLOT      05 : DYNAMIC RAM - 4 MB
SLOT      06 : DYNAMIC RAM - 4 MB
SLOT      20 : ND-5000 MODEL: 00B
```

Slotno:01<CR>

SLOT 01 : MF-BUS CONTROLLER STANDARD

TIME OUT(2-40 MIC SEC):6<CR>

MAINT CONTROL REG. (RETURN=DEFAULT):<CR>

ERROR INVESTIGATOR ON? :N<CR>

REPORT 1-BIT ERRORS ? :N<CR>

NEW BAUDRATE ? :N<CR>

- WRITING TO NONVOLATILE MEMORY, PLEASE WAIT -

Slotno:02<CR>

SLOT 02 : TWIN 16 BIT PORT (PRINT 5155)

EXPLAIN PORT PARAMETERS :YES<CR>

Memory areas are opened for access by giving LOWER and UPPER LIMITS.

LOWER LIMIT <= area < UPPER LIMIT

Several non-overlapping areas are allowed.

START ADDRESS is the first physical address in the MFB-memory.

LIMITS and START ADDRESS are in modules of 128 KB (0=0B, 1=400000B, 2=1000000B,..., n=n*400000B).

DATALENGTH is 16 or 32 bits.

INTERLEAVE TYPE is 0, 2, 4 or 8.

LOWER LIMIT:44<CR>

UPPER LIMIT:244<CR>

COMMENT:

$$\text{LOWER LIMIT} = \frac{\text{PRIVATE 100-MEMORY (KB)}}{128}$$

$$\text{UPPER LIMIT} = \frac{\text{AMOUNT OF MFB-MEMORY (KB)}}{128} + \text{LL}$$

ACCESS (LOCAL=1, GLOBAL=2, BOTH=3):1<CR>

COMMENT: Local means only within the bank.
Global means outside the bank.

MORE LIMITS (YES/NO):N<CR>

COMMENT: If holes inside the memory are wanted, add more limits.

START ADDRESS (PORT BASE):<CR>
 DATALENGTH (16, 32):16<CR>

COMMENT: The datalength tells if it is a 32-bit source
 or a 16-bit source connected to the port.

INTERLEAVE TYPE (0, 2, 4, 8):2<CR>
 INTERLEAVE PORT NUMBER (0-3):<CR>

COMMENT: The interleave port number refers to bit2 and
 bit 3 in the PORT CONTROL REGISTER.

REQUEST DELAY (10, 30, 40, 60):40<CR>

COMMENT: The request delay refers to bit 4 and bit 5 in
 PORT CONTROL REGISTER. Default=40nS

BUFFERED WRITE (Y/N):YES<CR>

COMMENT: Buffered write means that, when doing a write
 cycle, data ready is returned as soon as the
 data is latched into the port, but before the
 MFB cycle is finished.

MASTER CONTROL REGISTER(CR=DEF.):<CR>

COMMENT: Refer to the MASTER CONTROL REGISTER.

SAVE (YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

Slotno:03<CR>
 SLOT 03 : DYNAMIC RAM - 4 MB
 LOWER LIMIT (256 KBYTE INCREMENT (OCTAL)):<CR>
 RAM CONTROL REG.:<CR>

- LOADING PARAMETERS TO BOARD, PLEASE WAIT -
 SAVE(YES/NO):YES<CR>

COMMENT: NO = Configuration parameters are stored only
 in the registers on this module.
 YES= Configuration parameters are also saved in
 the non-volatile memory in the backplane.

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

Slotno:04<CR>
 SLOT 04 : DYNAMIC RAM - 4 MB
 LOWER LIMIT (256 KBYTE INCREMENT (OCTAL)):20<CR>
 RAM CONTROL REG.:<CR>

- LOADING PARAMETERS TO BOARD, PLEASE WAIT -
 SAVE(YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

Slotno:05<CR>
 SLOT 05 : DYNAMIC RAM - 4 MB
 LOWER LIMIT (256 KBYTE INCREMENT (OCTAL)):40<CR>
 RAM CONTROL REG.:<CR>

- LOADING PARAMETERS TO BOARD, PLEASE WAIT -
 SAVE(YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

Slotno:06<CR>
 SLOT 06 : DYNAMIC RAM - 4 MB
 LOWER LIMIT (256 KBYTE INCREMENT (OCTAL)):60<CR>
 RAM CONTROL REG.:<CR>

- LOADING PARAMETERS TO BOARD, PLEASE WAIT -
 SAVE(YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

WARNING

You need a special tool (part no. 350157) when configuring the CPU.

Slotno:20<CR>
 SLOT 20 : ND 5000 MODEL: 00B
 OCTOBUS STATION NO :70<CR>

COMMENT:	STATION NO:	TYPE OF STATION:
	1	ND-100
	2- 7	MFbus controller
	10-13	SCSI Controllers (disk)
	14-15	Matra VME
	16-17	Multifunction communication
	20	Hyperchannel
	21-23	FDDI (Fibernet)
	24-27	FPS-5000
	30-33	Graphic controller
	34-67	Free for expansion
	70-76	ND5000

POWER FAIL DESTINATION (CR gives default=1):<CR>
 REC. BROADCAST TYPE (CR gives default=0):<CR>

- LOADING PARAMETERS TO BOARD, PLEASE WAIT -
 SAVE(YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -
 SAVE(Y/N):Y<CR>
 WRITING TO NONVOLATILE MEMORY - PLEASE WAIT- READY

>AUTOINITIATE-BANK <CR>

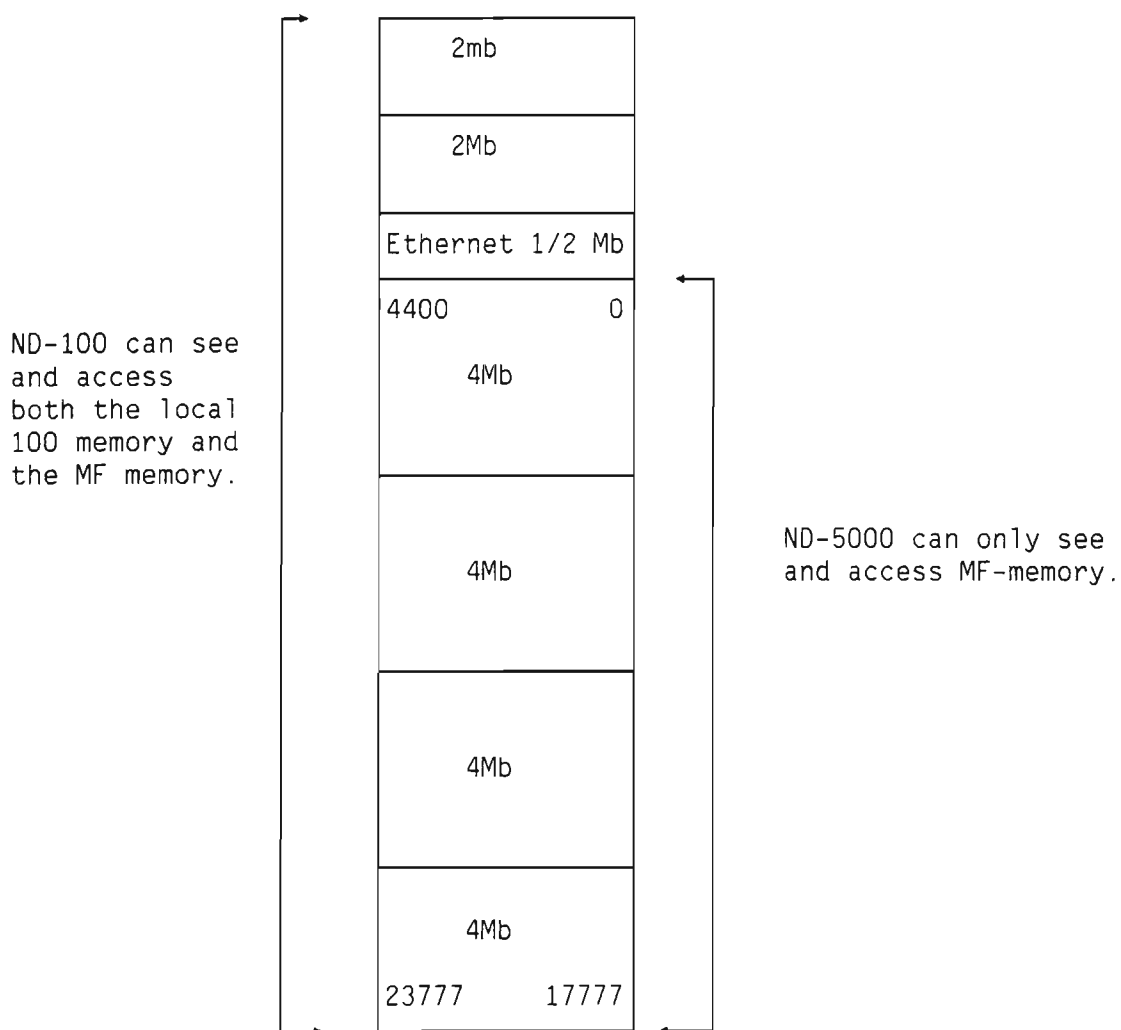
Use the ND-5000 monitor command MEM-CONFI to check the complete memory configuration:

ND-5000: mem-confi

PART	WIDTH	N100	N500P	N500D
OB-	17777B	Y	Y	Y

	PAGE		WORD	BYTE
	ND-100	ND-500	ND-100	ND-500
ND-500 address zero:	004400	000000	000110000000	000000000000
ND-500 register block:	004524	000124	000112500000	000005200000
Physical segment table:	004564	000164	000113500000	000007200000
WIP/PGU table:	004523	000123	000112460000	000005140000

ND-5000: ex



2. Replace the ND-110/CX with a ND-120/CX - 4 Mb. The 4 Mb memory on the ND-120/CX board makes it necessary to change the switchsettings in the ND-100 card crate and to reconfigure the MF-system.

3. Change the switchsettings on the memory cards in the ND-100 card crate to the values shown in the table below:

Module	Lower limit switchsetting	Upper limit display	Ethernet bank no.
ND-120/CX		200	
1st 2Mb board	200	300	
2nd 2Mb board	300	400	
Ethernet			040

Configure the MF-system, using the command CONFIGURATE-SLOT. It is only necessary to configure the port module after the upgrading.

>CONFIGURATE-SLOT<CR>

MODULES IN THIS BANK

```
=====
SLOT      01 : MF-BUS CONTROLLER STANDARD
SLOT      02 : PORT TWIN 16 BIT (PRINT 5155)
SLOT      03 : DYNAMIC RAM - 4 MB
SLOT      04 : DYNAMIC RAM - 4 MB
SLOT      05 : DYNAMIC RAM - 4 MB
SLOT      06 : DYNAMIC RAM - 4 MB
SLOT      20 : ND-5000 MODEL: 00B
```

Slotno:02<CR>

SLOT 02 : TWIN 16 BIT PORT (PRINT 5155)

EXPLAIN PORT PARAMETERS :YES<CR>

Memory areas are opened for access by giving LOWER and UPPER LIMITS.
LOWER LIMIT <= area < UPPER LIMIT

Several non-overlapping areas are allowed.

START ADDRESS is the first physical address in the MFB-memory.

LIMITS and START ADDRESS are in modules of 128 KB (0=0B, 1=400000B, 2=1000000B, ..., n=n*400000B).

DATALENGTH is 16 or 32 bits.

INTERLEAVE TYPE is 0, 2, 4 or 8.

LOWER LIMIT:104<CR>

UPPER LIMIT:304<CR>

ACCESS (LOCAL=1, GLOBAL=2, BOTH=3):1<CR>

MORE LIMITS (YES/NO):N<CR>

START ADDRESS (PORT BASE):<CR>

DATALENGTH (16, 32):16<CR>

INTERLEAVE TYPE (0, 2, 4, 8):2<CR>

INTERLEAVE PORT NUMBER (0-3):<CR>

REQUEST DELAY (10, 30, 40, 60):40<CR>

BUFFERED WRITE (Y/N):YES<CR>

MASTER CONTROL REGISTER(CR=DEF.):<CR>

SAVE (YES/NO):YES<CR>

- WRITING TO NON-VOLATILE MEMORY, PLEASE WAIT -

4. Verify the "new" configuration by using the command LIST-CONFIGURATION:

>LIST-CONFIGURATION<CR>

Slotno: 1<CR>

SLOT 01 : MF-BUS CONTROLLER STANDARD
MAINTENANCE CONTR. REG: 000415B
TIME OUT ON MFB-BUS : 000006
BAUDRATE ON console : 009600

Slotno: 2<CR>

SLOT 02 : TWIN 16 BIT PORT (PRINT 5155)
PORT START ADDRESS : 000000B
PORT CONTROL REGISTER : 000041B
MASTER CONTROL REGISTER : 000125B

LIMITS THAT DEFINE ACCESS AREAS FOR THE PORT.

LOW LIMIT : 000104B HIGH LIMIT : 000304B LOCAL

Slotno: 3<CR>

SLOT 03 : DYNAMIC RAM - 4 MB
LOW LIMIT OF RAM : 000000B
RAM CONTROL REGISTER : 000000B

Slotno: 4<CR>

SLOT 04 : DYNAMIC RAM - 4 MB
LOW LIMIT OF RAM : 000020B
RAM CONTROL REGISTER : 000000B

Slotno: 5<CR>

SLOT 05 : DYNAMIC RAM - 4 MB
LOW LIMIT OF RAM : 000040B
RAM CONTROL REGISTER : 000000B

Slotno: 6<CR>

SLOT 06 : DYNAMIC RAM - 4 MB
LOW LIMIT OF RAM : 000060B
RAM CONTROL REGISTER : 000000B

Slotno: 20<CR>

SLOT 20 : ND 5000 MODEL: 00B
STATION NO : 000070B
POWER FAIL DESTINATION : 000001B
BROADCAST TYPE : 000000B
SPEED : 000000B
CPU MODEL : 000004B
MASTER CONTROL REGISTER: 000201B

LIMITS THAT DEFINE ACCESS-AREAS FOR THIS SLOT.

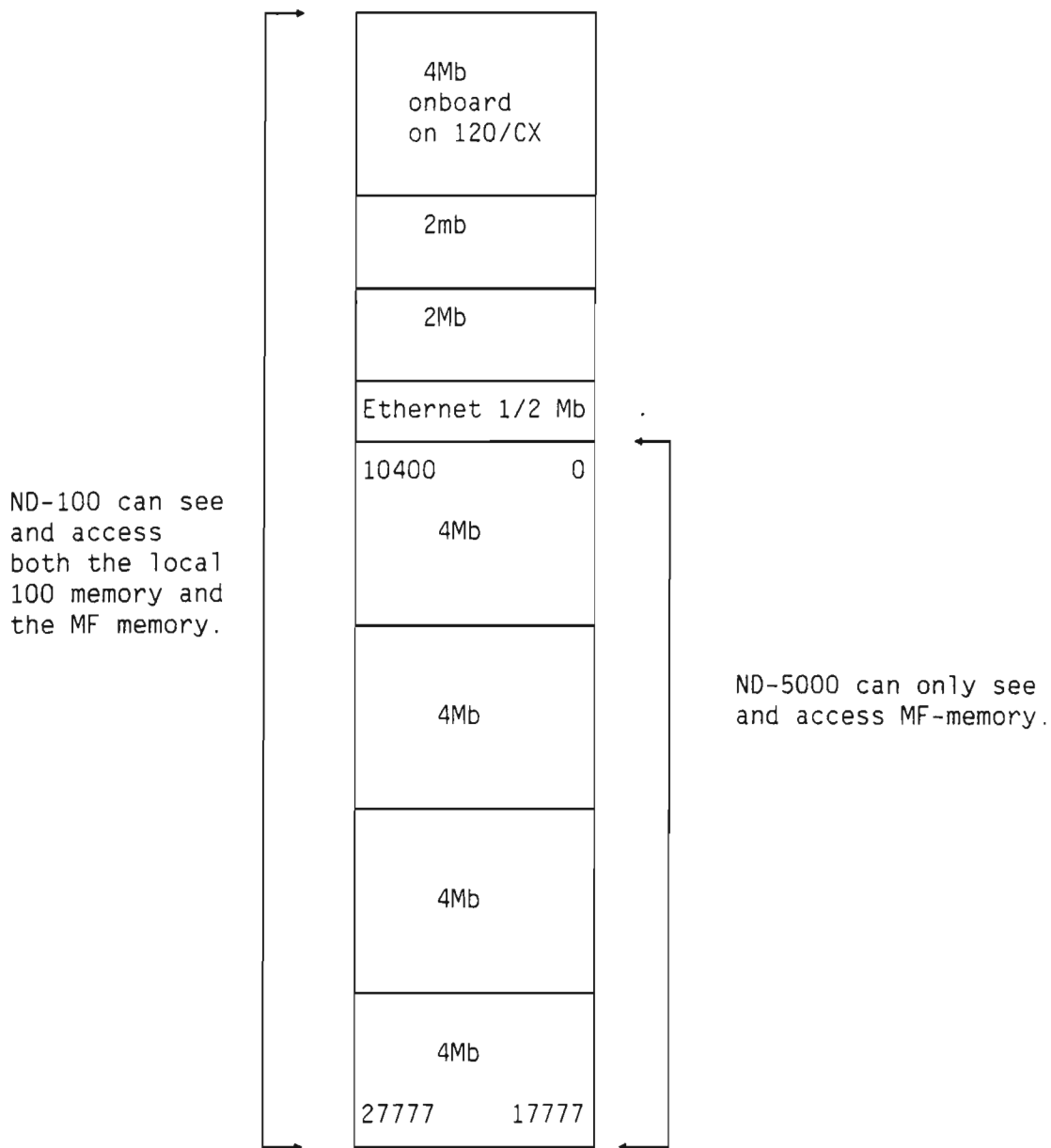
Use the ND-5000 monitor command MEM-CONFI to check the complete memory configuration:

ND-5000: mem-confi<CR>

PART	WIDTH	N100	N500P	N500D
OB-	17777B	Y	Y	Y

	PAGE		WORD	BYTE
	ND-100	ND-500	ND-100	ND-500
ND-500 address zero:	010400	000000	00021000000	000000000000
ND-500 register block:	010524	000124	00021250000	00000520000
Physical segment table:	010564	000164	00021350000	00000720000
WIP/PGU table:	010523	000123	00021246000	00000514000

ND-5000: ex



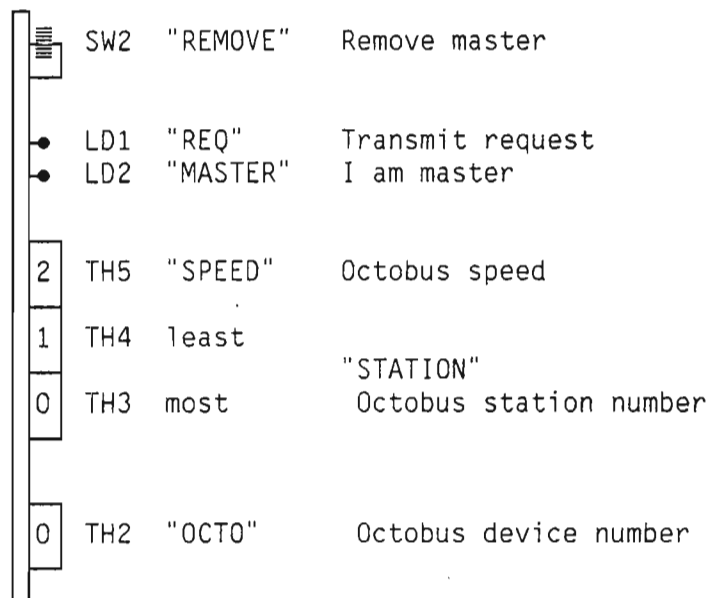
CHAPTER 9 SWITCHES AND INDICATORS

This chapter describes the LEDs and switches on the ND-5000 cards. For ND-100 cards, see the ND-100 Hardware Maintenance Manual (ND-30.008) or the Service Handbook.

THE MOTHER BOARD (5502)

- Yellow LED: "OCTO" = Octobus activity.
Flashes each time the ACCP receives information via the octobus. It will flash heavily when control store is being loaded
- Green LED : "MACRO".
Lights when the CPU executes macro program.
- Yellow LED: This LED lights when the tracer is triggered.
- Red LED : "MERR" = Memory error, normally OFF.
- Green LED : "ECMIR".
Lights when the microprogram is running.
- Yellow LED: "AMODE" = ACCP-mode.
Lights when the ACCP has control of the CPU, i.e. during bootstrapping etc.
- Red LED : "MR" = Master Reset.
Lights at power up reset or when Master Clear is received. Turned OFF by AACCP during initialisation. If self-test failed, this LED flashes until RESET-CPU is performed. The LED may also be turned OFF by typing CTRL-X (ACCP software reset) on ACCP-consol.

THE MPM LINE DRIVER



Setting of octobus device number:

Thumbwheel TH2 "OCTO" uses only 4 of its 16 positions to set the device number.

Th.W. no.	OBC No	IOX No	IDENT CODE LEV 13	
			Receive	Transmit
0	OBC 0	100400	40	41
1	OBC 1	100410	42	43
2	OBC 2	100420	44	45
3	OBC 3	100430	46	47

Setting of octobus station number:

Octobus station number for ND100 is defined to be number 1.
Devices connected to the global octobus shall be given station numbers from 1-17B.

The station number is set by two thumbwheels.

The thumbwheels TH3 and TH4 "STATION" uses 8 of its position each for setting of the station number.

The setting of the station number should then be in octal and the least digit on TH4 and most on TH3.

Setting of octobus speed:

The thumbwheel TH5 "SPEED" uses only 4 of its position for setting the speed of the OCTOBUS.

Th.W. no.	SPEED (MHz)
0	4.0
1	1.0
2	1.0
3	0.5

Normal speed setting is 4 Mhz.

NOTE:

In "old" cabinets the Octobus speed should be set to 1 Mhz.
Remember to set the same Octobus speed on the MF Controller.

Setting of destination station number in case of power fail:

The DIP-switch in pos. 17H is divided in two groups which specifies the values for:

- a) Power Fail Destination
- b) Broadcast Type

17H									
1	2	3	4	5	6	7	8	9	10
PFDES						BT			
0	1	2	3	4	5	6	2	1	0

The station number must be unique for each node in an OCTOBUS system. Power Fail Destination and Broadcast Type however, may be similar for different nodes.

The broadcast type (BT 0-2) determine what type of receiver a station is. BT0-2 can represent 8 different types of receivers, of which some codes are defined:

0	Not assigned
1	Not assigned
2	ND-100
3	Not assigned
4	Not assigned
5	Bank Processor MPM5
6	No broadcasts to be received
7	All broadcasts to be received

THE CACHE MODULE (5610)

- Green LED: "WICO". Lights when the Write In Cache mode is ON.
- Green LED: Lights when the instruction cache is turned ON.
- Green LED: Lights when the data cache is turned ON.

ON OFF



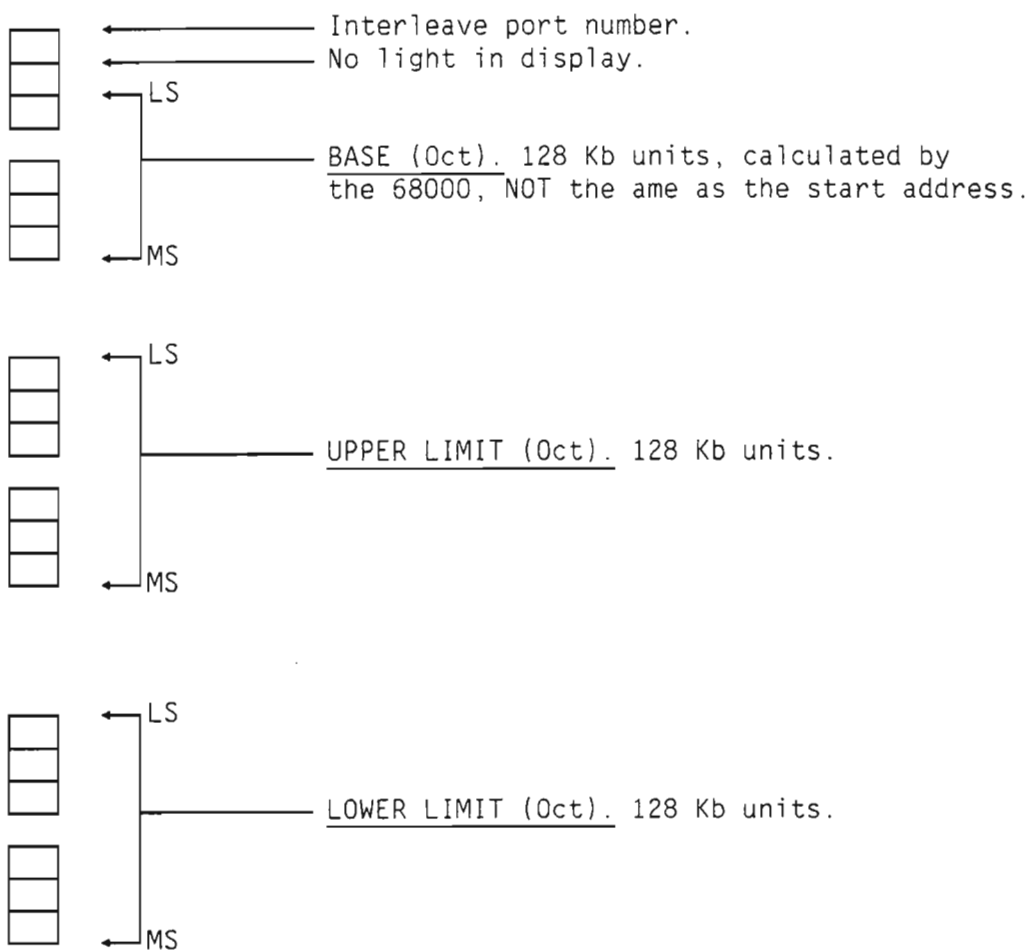
DCA : Data cache enable switch



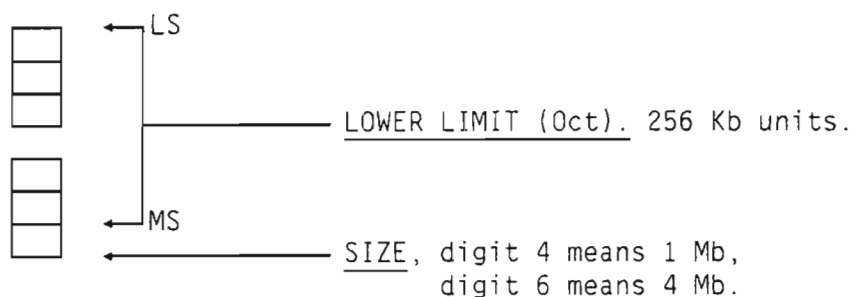
ICA : Instruction cache enable switch

THE MFB BUS PORT

- LD 1 Red : PORT IN TEST or NAVIB (not available).
NAVIB can be programmed or hardwired XMINH,
i.e., port is not connected to any driver.
- LD 2 Yellow : Request to port.
- LD 3 Yellow : Request within port address to port.



THE DYNAMIC RAM (5411)

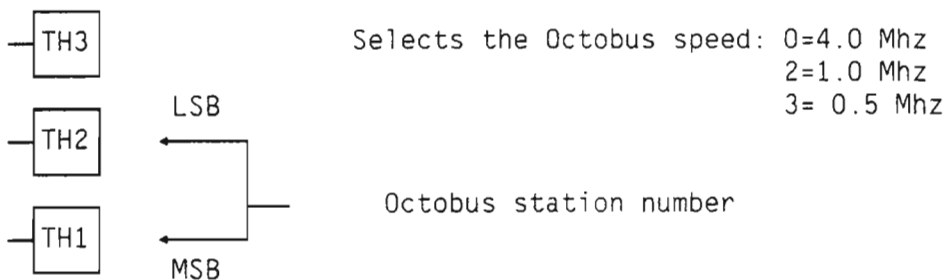


- LD 4 Red : HARD ERROR, means that the error investigator has found an error that must be corrected for every cycle.
- LD 5 Red : BAD MEMORY, non-correctable error has been detected within this module.
- LD 1 Yellow : CORRECTED, means that at least one error correction is done. Cleared by hard or soft reset, power down or disable/enable SW 1.
- LD 2 Green : ENABLED, lit when error correction is enabled.
- ☐ SW 1 ERROR CORRECTION enable/disable.
Normal operation: Enabled (switch up).
- LD 2 Yellow : ACCESS, means module is accessed with memory cycles.

THE MFB CONTROLLER (5464)

- LD 3 Green POWER O.K., means that 5 volts standby is present on the board. Normally lit.
- LD 2 Red DIS, disable, means that switch 3 (powerfail disable) on this card is disabled. Normally not lit.
- LD 1 Yellow TRFO, means that REFRESH is running. Normally lit.

- ☐ SW 3 POWERFAIL DISABLE, normal operation enabled (switch in middle position).
- ☐ SW 2 SOFT RESET, produces a simulated powerfail.
- ☐ SW 1 REMOVE, removes refresh.
- ☐ SW 4 TOTAL RESET, same as both 5 volts and 5 volts standby OFF. Initiates memory and loads the RAM modules with the configuration parameters from the EPROM in the backwiring.



THE DOUBLE BUS CONTROLLER

—TH1	ND-100 Device number	TH1 Device No. Ident
		0 100400 40/41
		1 100410 42/43
		2 100420 44/45
		3 100430 46/47
—TH4	ND-100 Device number	TH4 Station No.
		0 Not allowed
		1 1
		2 2
		etc.
• LD4	Yellow	Indicates that ND-100 accesses within ND-100 limits
		Strap field, 0 0 0 0 0
		ECO-version: 0 0 0 0 0
		0 0 0 0 0
		0 STR 8: Normal strap setting
• LD3	Yellow	MASTER. Means that controller is master. Always lit.
• LD2	Green	Power OK. Means that 5 volt and 5 volt stand by is present on the board. Always lit.
		0
		0 STR 7: Normal strap setting
• LD1	Red	DIS. Means that switch 3 (power fail detection) is disabled. Normally not lit.
—14L	SW3	POWER FAIL DISABLE Normal operation enabled (middle position)
—12L	SW2	SOFT RESET Produces a "power fail" signal to the MF-bus. (same as 5V OFF).
—13L	SW1	TOTAL RESET Normal operation enabled (middle position) Same as both 5V and 5V stand by OFF. Initiates memory, takes the configuration from the EEPROM in the backplane and writes it to the boards.
—TH3	Speed on octobus	TH3 Speed
		0 4.0 Mhz
		** 2 1.0 Mhz
		3 0.5 Mhz
—TH2	Processor station number	TH2 Station
		0 Not allowed
		1 1
		2 2
		** 3 3
		etc.
		ooo STR 6 Normal strap setting
		oooooooooooooooooooo SW5, SW6 Dip switch octobus functions
		oooooooooooooooooooo o STR9 Normal case in.

APPENDIX A

A.1 OCTOBUS MESSAGES FROM/TO THE ND5000 MICROPROGRAM

Fatal:

The CPU is set unavaialble by the microprogram. No one is allowed to run anything on the CPU. Only inspections by LOOK-AT commands by the ND-500/5000 monitor is allowed.

The following is a description of multibyte messages routed via Octo-bus defined ErrorStation or defined HostStation.

```

<StartOfMessage> 100060B Ored <HostStation> or <ErrorStation>.
<SourceOmd>      Set = 4
<NumberOfBytes>   = <MessageBody.<NumberOfBytes>> + 2
<FaultType>
    200B = Hardware Fault           : Fatal      38 Bytes
    201B = General Trap Message     : NotFatal  14 Bytes
    202B = Wrong microprogram       : NotFatal   6 Bytes
    203B = Wrong microprogram       : Fatal      6 Bytes
    204B = Unrecognized kick        : NotFatal   4 Bytes
    205B = Unrecognized message     : NotFatal   4 Bytes
    206B = Unrecognized emergency   : NotFatal   4 Bytes
    207B = Unrecognized Accp Command : NotFatal   4 Bytes
    210B = Unexpected external trap : NotFatal   2 Bytes
    211B = Size of Wip&Pgu table = 0 : Fatal      2 Bytes
    212B = Pst pointer = 0          : Fatal      2 Bytes
    277B = MicroProgram error in Trap_Ocbm 2 Bytes
<ErrorReporter>   1 = MicroProgram
<MessageBody>
<EndOfMessage> 100040B Ored <HostStation> or <ErrorStation>.
  
```

A.2 CONTENTS OF THE DIFFERENT MESSAGES SENT AS AN OCTOBUS MESSAGE

Fault Type = 200B	Hardware Fault	: Fatal
MessageBody	Process No.	: 2 Bytes
	Trapping P	: 4 Bytes
	Restart \bar{P}	: 4 Bytes
	Trap_no	: 2 Bytes
	Logical_address	: 4 Bytes
	Pv_code	: 4 Bytes
	Physical_address	: 4 Bytes
	Physical_segment	: 4 Bytes
	Working register	: 4 Bytes
	Srf.ASTS	: 2 Bytes
	Srf.BADAP	: 2 Bytes
	total	: 38 Bytes

```

Fault Type = 201B  General trap message      : NotFatal.
MessageBody       Process No.                : 2 Bytes
                  Trapping_P                 : 4 Bytes
                  Restart_P                  : 4 Bytes
                  Trap_no                     : 2 Bytes
                                      total : 14 Bytes
*****
Fault Type = 202B  Wrong microprogram        : NotFatal
Fault Type = 203B  Wrong microprogram        : Fatal
MessageBody       Current CPUModel           : 1 Byte A
                  My CPUModel                : 1 Byte A
                  MicroprogramVersion        : 2 Bytes
                                      total : 6 Bytes

```

```

Fault Type = 204B  Unexpected Octo-bus Kick      : NotFatal
Fault Type = 205B  Unexpected Octo-bus MultiByte message : NotFatal
Fault Type = 206B  Unexpected Octo-bus Emergency message : NotFatal
Fault Type = 207B  Unexpected ACCP command       : NotFatal
MessageBody       Message Header                : 2 Bytes
                                      total : 4 Bytes
*****
Fault Type = 210B  Unexpected external trap.      : NotFatal
Fault Type = 211B  Size of Wip&Pgu table = 0     : Fatal
Fault Type = 212B  Pst pointer = 0              : Fatal
Fault Type = 277B  Microprogram error in Ocb-message : NotFatal
MessageBody       -----                      : 0 Bytes
                                      total : 2 Bytes

```

A.3 COMMANDS RECEIVED BY THE ACCP

RSYSPAR =	15B	% 0Dh Read system parameters
SYSPAR =	16B	% 0Eh Load system parameters
ECHO =	17B	% 0Fh echo test
RECO =	20B	% 10h read ECO levels
LPARPNT =	21B	% 11h load parameter pointer
VERPARP =	22B	% 12h verify parameter pointer
LOCS =	23B	% 13h load cs (via MPM)
LCSD =	24B	% 14h load CS direct
DUCS =	25B	% 15h dump cs (via MPM)
DCSD =	26B	% 16h dump CS direct
DEBCNTMIC =	27B	% 17h Debug Contmic (no STOP sync)
AMICTRAP =	30B	% 18h ACCP Microprogram trap
RUNSELFTEST =	33B	% 18h run long selftest
STOPMIC =	34B	% 1Ch stop mic.prog
CONTMIC =	35B	% 1Dh cont. micro program
RESTMIC =	36B	% 1Eh restart micro program
ALIVE =	37B	% 1Fh alive check
LMAR =	40B	% 20h load MAR
LMIR =	41B	% 21h load MIR
RMIR =	42B	% 22h read MIR
TESTBUS =	43B	% 23h bustest
RAIB16D =	44B	% 24h Read AIB16 directly
RAIB32D =	45B	% 25h Read AIB32 directly
LAOB16D =	46B	% 26h Load AOB16 directly
LAOB32D =	47B	% 27h Load AOB32 directly
RASTS =	50B	% 28h read asts
LMODE =	51B	% 29h write mode
LCON =	52B	% 2Ah write con
WMPM =	53b	% 2Bh write multiport
RMPM =	54B	% 2Ch read multiport
SETTRAC =	55B	% 2Dh set trace sel.
SCLOCK =	56B	% 2Eh set clock
RCLOCK =	57B	% 2Fh read clock
READSELFTEST =	60B	% 30h read selftest status
ENKICK =	61B	% 31h enable kikcs
DISKICK =	62B	% 32h disable kikcs
TESTBUF =	63B	% 33h buffertest
LAOB32 =	64B	% 34h Load AOB32 via MPM
RAIB32 =	65B	% 35h Read AIB32 via MPM
STAMICD =	66B	% 36h Start mic-prg. directly
LOOP =	67B	% 37h Set scop-loop mode
SPEED =	70B	% 38h Set clock speed
CPURES =	71B	% 39h Reset Samson CPU
TESTMPM =	72B	% 3Ah Test multiport
DCCD =	73B	% 3Bh dump CC direct
DUCC =	74B	% 3Ch dump CC (via MPM)
PROMVERS =	75B	% 3Dh Read ACCP PROM version
CPU MODEL =	76B	% 3Eh Read CPU MODEL

ACCP-initiated messages to ND-100:

HWfault = 200B % ND-5000 hardware fault

Emergency messages (C-bit (bit 15) set), detected by hardware:

ARES = 241B % Master clear on ND-5000 CPU

ACONT = 242B % Continue ACCP

ASTOP = 243B % Stop ACCP

TERM = 244B % Terminate ACCP

A.4 THE CONTEXT BLOCK(REGISTER BLOCK)

The context block is saved and loaded from physical ND5000 memory. Current executing process number * 400B is used as index in the context block save area in order to access correct context block. Some registers are connected to a domain and will be updated in the domain information table by all instructions affecting of these registers. Thus it is not necessary to save these registers when changing process number. The registers will be loaded before execution is started. Register enclosed by parenthesis are not saved in or loaded from context block when changing to a new process. They will be loaded from the domain information table before execution is started.

A pointer to the start of the context block is patched in location OFFSET (address 20) in the microprogram when loading the control store. The physical address of the context (register) block can also be found by the command LIST-MEMORY-CONFIG in ND5000 Monitor. The command LOOK-AT-RESIDENT from user SYSTEM can be used to look at physical multiport memory. The context block (register block) is 400B byte. The first block is always dummy, further the process number can be used to find the correct block.

Address of current block may be calculated as follows:

Start_of_register_block+400B+process_No*400B

Register number	Context disp.	Trap disp.	DIT Disp.	Register name	Register symbols
		000B	024B	Trapping P register	(P)
0	0B	000B	030B	Restart P register	(P)
1	1B	004B	034B	Link register	(L)
2	2B	010B	040B	Base register	(B)
3	3B	014B	044B	Record register	(R)
4	4B	020B	050B	Index register 1	(X1)
5	5B	024B	054B	Index register 2	(X2)
6	6B	030B	060B	Index register 3	(X3)
7	7B	034B	064B	Index register 4	(X4)
8	10B	040B	070B	Floating most register 1	(A1)
9	11B	044B	074B	Floating most register 2	(A2)
10	12B	050B	100B	Floating most register 3	(A3)
11	13B	054B	104B	Floating most register 4	(A4)
12	14B	060B	110B	Floating least register 1	(E1)
13	15B	064B	114B	Floating least register 2	(E2)
14	16B	070B	120B	Floating least register 3	(E3)
15	17B	074B	124B	Floating least register 4	(E4)
16	20B	100B	130B	Status register 1	(ST1)
17	21B	104B	134B	Status register 2	(ST2)
18	22B	110B	140B	Process segment register	(PS)
19	23B	(114B)	144B	274B Top of stack register	(TOS)
20	24B	(120B)	150B	300B Lower limit register	(LL)
21	25B	(124B)	154B	304B Higher limit register	(HL)
22	26B	(130B)	160B	266B Trap handler register	(THA)
23	27B	134B	164B	Current executing domain register	(CED)
24	30B	140B	170B	Current alternative domain register	(CAD)
25	31B	144B	174B	Current executing segment register	(CES)
26	32B	150B	200B	Current alternative segment register	(CAS)
27	33B	154B	204B	Micro program scratch register 1	(SC1)
28	34B	160B	210B	Micro program scratch register 2	(SC2)
29	35B	(164B)	214B	226B Own trap enable register 1	(OTE1)
30	36B	(170B)	220B	232B Own trap enable register 2	(OTE2)
31	37B	(174B)	224B	236B Child trap enable register 1	(CTE1)
32	40B	(200B)	230B	242B Child trap enable register 2	(CTE2)
33	41B	(204B)	234B	246B Mother trap enable register 1	(MTE1)
34	42B	(210B)	240B	252B Mother trap enable register 2	(MTE2)
35	43B	(214B)	244B	256B Trap enable modification mask 1	(TEMM1)
36	44B	(220B)	250B	262B Trap enable modification mask 2	(TEMM2)
		324B		Reason for programmed trap (ERRCODE)	

Information saved at trap.						
Register number	Context disp.	Trap disp.	DIT Disp.	Register name	Register symbols	
36	44B	220B		Trap number saved causing trap handler missing.		
37	45B	224B	254B	Trapping P		
38	46B	230B	260B	Status, trapped between trap and entt finished		
				0: Normal execution.		
				1: During or before trap.		
				2: After trap.		
39	47B	234B		Trap number.		
40	50B	240B	24B	Restart P.		
41	51B	244B	14B	Protect violation information (MMS.STS)		
42	52B	250B	20B	Protect violation address. (MMS.LA)		
		254B		MMS Physical address. (MMS.PHYS)		
		260B		Physical Segment Number. (MMS.PHS/CAP)		
		264B		WR register. (MMS.WR)		
		272B		Slot no./BADAP status.		

Register number	Context disp.	Trap disp.	DIT Disp.	Register name	Register symbols	
				200B CED of calling domain.		
				201B CAD of calling domain.		
				203B Return address in calling domain.		
				207B Base register in calling domain.		
				213B CED of trapped domain.		
				214B CAD of trapped domain.		
				272B CED of mother domain.		
				273B Flag for 'inside' trap handler.		
				310B Flag for 'PIA' (bit 0 = 1 : 'PIA' = 1		

A.5 ALLOCATION OF REGISTERS IN THE SCRATCH REGISTER FILE

The size of the hardware scratch register file (SRF) is 4K x 32 bit. The SRF may be displayed by giving the command:

LOOK-AT-SRF <address> address within SRF 0-4000B

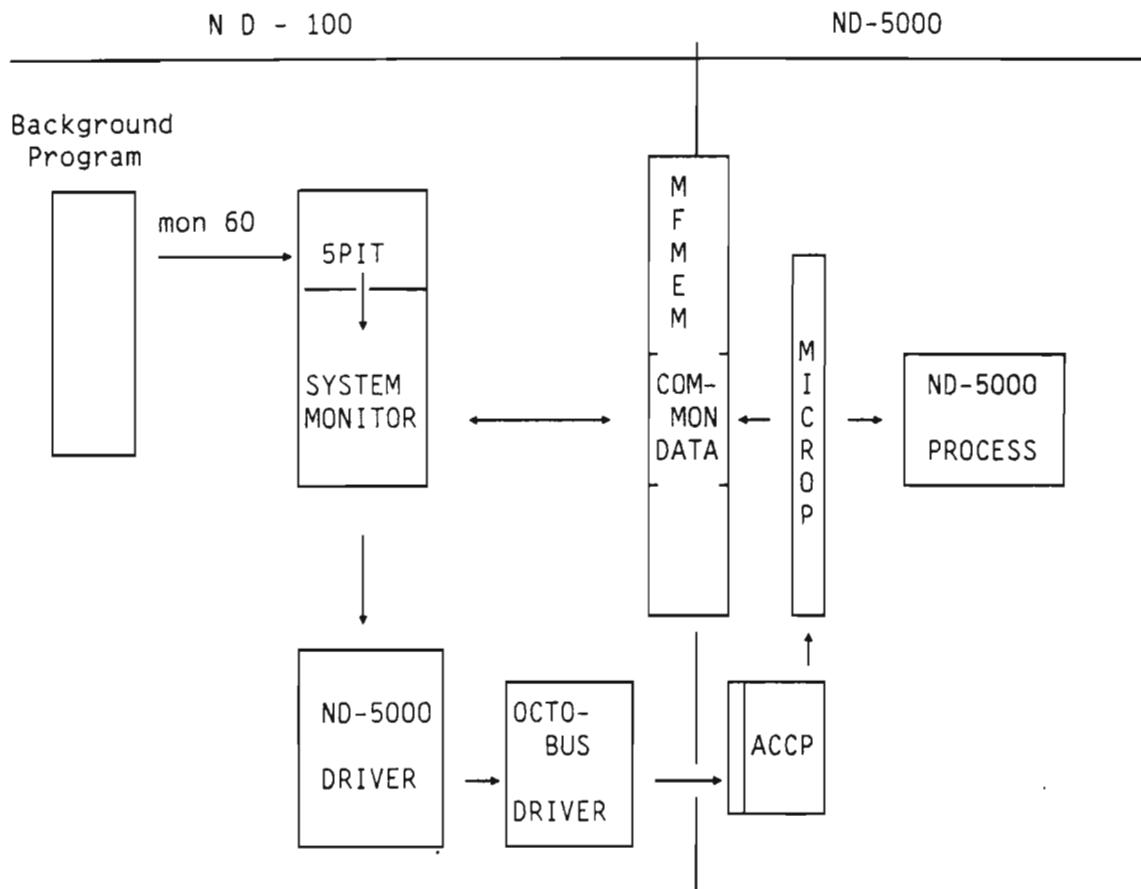
Initializing of different registers in the SRF are done during start up of the microprogram.

status	reg	use
reserved	SRF00	Constant = 0
reserved	SRF01	Constant = 66666666H (used by BCD)
reserved	SRF02	Constant = 60000000H (used by BCD)
reserved	SRF03	Constant = 00017777777B
reserved	SRF04	Constant = 17760000000B
reserved	SRF05	Constant = 30000000000B
reserved	SRF06	Constant = 17777777777B
reserved	SRF07	Constant = 11111110H (used by BCD)
reserved	SRF10	Status 2 surugat
reserved	SRF11	Current process number+1
reserved	SRF12	TOS.
reserved	SRF13	PS register.
reserved	SRF14	CED register.
reserved	SRF15	CAD register.
reserved	SRF17	THA register.
Used when trap occure.	SRF32	Slot pos/BADAP status.
	SRF33	ACCP status.
	SRF34	MMS.WR register.
	SRF35	MMS.CAP/PHS register
	SRF36	MMS.PHYS register.
	SRF37	MMS.LA register.
	SRF40	MMS.STS register.
	SRF41	Restart P
	SRF42	Trapping P
	SRF1777	Last octobus msg from microprogram: backward
	SRF2000	Address of Current Message.
	SRF2001	Max. index in FIFO (WM500)
	SRF2002	Address of the FIFO buffer (WM500)
	SRF2003	Communication flag (0=idle, 1=running >1=trap in process.)
Read from the ACCP at start up	SRF2004	Optional parameter 2 (two bytes)
	SRF2005	Host station number and OMD number.
	SRF2006	Error station number and OMD number.
	SRF2007	Initial setting of the Modus register.
	SRF2010	Slot number and BADAP status at startup.
	SRF2011	ACCP status at startup.
	SRF2012	Modus register saved and used at trap.
	SRF2013	Process 0 started flag (false=0, true=1).
	SRF2014	Legal bits to modify in the Modus register.
	SRF2015	CPU type and model read from the ACCP.
	SRF2016	Flag for CPU available (false=1, true=0).
	SRF2017	Address of this #CPU datafield (WM500).
	SRF2020	Message flag (0: Status=1 1: Status><1)
	SRF4040-4520	Mathematical constants
	SRF4040-4520	Mathematical constants

Allocation of registers/labels used in the microprogram listing:

Labels	Address
ADR_MESS	2000B
ADR_FIFOB-1	2001B
ADR_FIFOB	2002B
ADR_FLAG	2003B
ADR_SYSPAR-2	2004B
ADR_SYSPAR-1	2005B
ADR_SYSPAR	2006B
ADR_MODINIT	2007B
ADR_ASTBAD-1	2010B
ADR_ASTBAD	2011B
ADR_MOD	2012B
ADR_PROCO	2013B
ADR_MODMASK	2014B
ADR_CPUPAR	2015B
ADR_CPUAVA	2016B
ADR_#CPUDF	2017B
ADR_MSGME	2020B

A.6 ND-100/ND-5000 COMMUNICATION



- Octobus and memory are used for communication and synchronisation.
- Common data (like message buffers) are placed in shared (MF) memory.

The micro program is at initialization of the ND5000 loaded into the control store and started. After going through some initialization, the micro program enters the IDLE-loop.

The initialization of the ND5000 implies clear the data and instruction cache and TSB, setting of the floating, BCD and integer constant registers, resetting of trap enable and the status registers.

The CPU type and model settings are checked, and the internal trace module is initiated and armed. The call/enter flag is also initiated. Then the micro program sets the current process number (Df.x5proc) to -1 before the idle loop is entered.

The communication between the ND100 and ND5000 is built on a message block, residing in shared memory. Before a message is activated in the ND-5000 CPU, the message block is initiated, depending on the operation to be executed. When the ND-5000 microprogram is activated, the execution queue will be scanned to find the first message block with status equal to 1. When such a message is found a routine in the microprogram is entered according to the function specified in the message block.

The microprogram will then use the block to return messages back to ND-100. The microprogram begins to scan the messagebuffer from the idle loop if Df.x5act = 0 in the extended datafield or if an octobus kick is received.

A.7 THE EXTENDED CPU DATAFIELD.

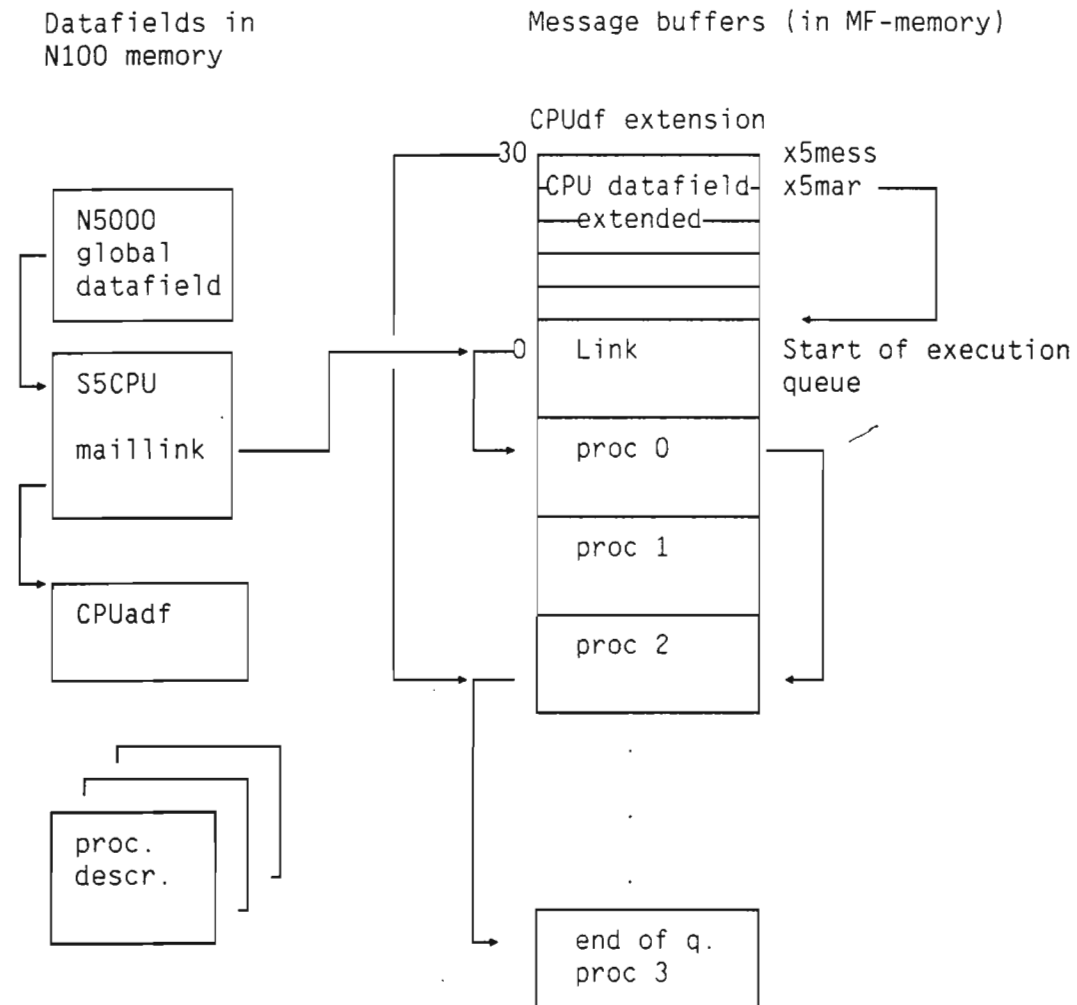
The message buffers are physical located in the shared memory.

The start of the datafield may be located as decribed below.

The CPU datafield can be found in S5CPU in SYMBOL-2-LIST.

```
@LOOK-AT SEGM S3DPIT
<S5CPU> + 21B / <bank no>  % N100 bank number
<address>                  % Address within bank
```

```
@LOOK-AT PHYS
<bank no> + <address> - 30B / <X5Mess>
```

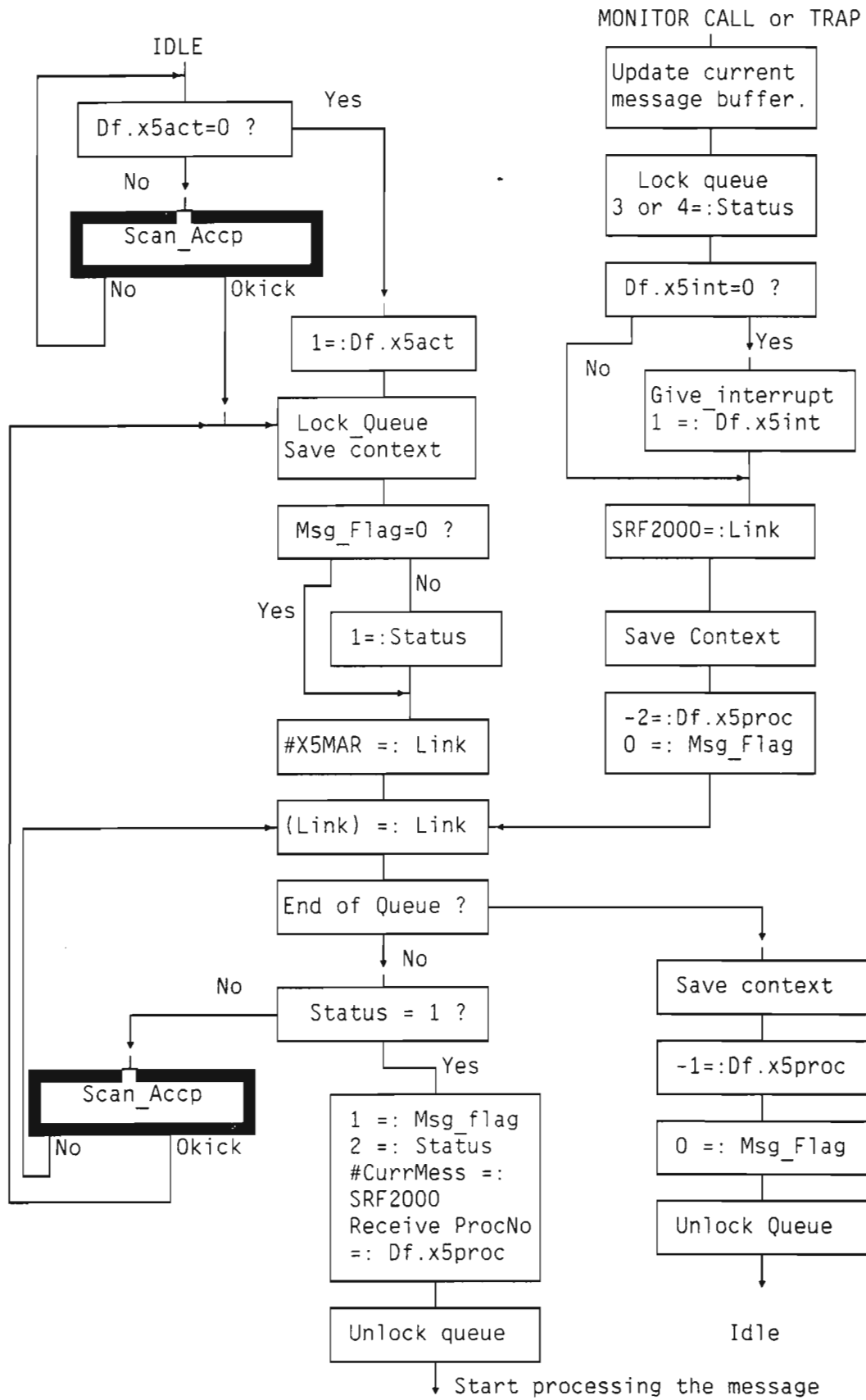


Byte Addr		
0 2	x5mess	Address of last message procesed by ND5000 CPU which an interrupt to ND100 was given.
4 6	x5mar	A pointer to the IDLE message.
10	x5sema	Reserve/Release semaphore of the execution queue. -1= Reserved 0 = Released (TSET is used).
12	x5int	Set to 1 by ND5000 when a message is finished together with interrupt and updating X5Mess. If X5INT >< 0: Inhibit interrupts and updating of X5Mess. X5Int is cleared by ND100.
14	x5act	Activate ND5000 from IDLE (0: Activate 1:Has been activated by this flag).
16	x5proc	Current process number (updated by the microprogram: -1= idle, -2= Scanning the execution queue.)
20 22	x5zero X5ACCP buffer	N100 page number of address zero in shared memory. N100 address of ACCP buffer.
24 26	x50ctob buffer	N100 address of octobus buffer.
30 32	X5HW buffer	N100 address of buffer used by LOOK-AT-HARDWARE.
34 40	x5CPU	ND-5000 CPU number which has reserved the execution queue (Processor number 1-4).
60 62	Next_Link	Link to first message in execution queue.
64	Status	Message status (set by TER500) 0: Free 1: Terminate ND5000 CPU
66 70 72 74	-1 0 0 47B	Micfunc = 47B: Terminate

The ND5000 CPU will when activated from the IDLE loop start scanning the execution queue pointed to by Df.x5mar. A location (START_MESS address 26) in the ND5000 microprogram points to Df.x5mar location in the extended CPU datafield. this location is patched during loading of the control store. When the ND5000 CPU is in IDLE state Df.x5proc will be -1. When the ND5000 CPU is activated the Df.x5proc will be set to -2 until a 1 or 2 in status of a message in the execution queue is found. Then Df.x5proc will be set equal to the RECEIVER process number. If the end of queue is found the ND5000 CPU will go idle and -1 is written into Df.x5proc.

Each time the ND5000 CPU is finished handling a message the X5INT location decides whether the ND5000 CPU are going to send an interrupt via the octobus to ND100 or not. If Df.x5int is zero it is set to one by the ND5000 CPU and an interrupt is send to ND100, and the location Df.x5mess is updated with the current message address. If Df.x5int >< 0,

an interrupt is not issued from the ND5000 CPU, and the Df.x5mess location will not be updated.



Since ND-100 is using -1 as a semaphore, it is not possible to use the semaphore from ND-5000 to decide which CPU reserved the semaphore. Additional information in the datafield is necessary to flag which CPU reserved the semaphore. This is valid both for workmode-400 and workmode-500. ND-100 should be unaffected by this, since release is setting 0 in the Df.x5CPU location.

A.8 THE MESSAGE BUFFER.

Each block contains a header and a data part. The header consists of six 16 bit words describing the message. The data part consists of a function value and a number of parameters depending on the operation to be carried out. The size of each message buffer is 256 bytes.

Locating the message buffer and process description:

- The ND100 memory bank may be found in 5MBBA address 4644 on segment S3DPIT:
@LOOK-AT SEGM S3DPIT
4644/ <bank no>
- Get reserved process description address from RT description for the shadow process in ND100 (BAKxx):
@LIST-RT-DESCRIPTION BAKxx

@LOOK-AT SEGM S3DPIT <RT descr. address+7>/ <address of
current message buffer inside
the bank> @LOOK-AT PHYS <bank no
+ address> / 177777 % Link address
177777
3 % Status

A.9 THE MESSAGE BLOCK

	-60		
	-32	5tslcounta	
	-30	5tslntime	ND-5000 time used when changing priority
	-26	5tslstatus	Timeslice status
	-24	pdclfg	SINTRAN addr. to enter after cleaning-up
	-22	sv5func	MON60 func. code in cleaning-up sequence
	-20	ND5000	ND-5000 CPU time used
	-16	time used	
	-14	CPUdf	Address of ND-5000 CPU df. used by this process
	-12	5priority	ND-5000 priority of process
	-10	htslowpri	Highest "low-timeslice" priority
	-6	Magic no	"Magic" part of process (sequence number)
		outdf	Addr of term output Df using proc.
H	-2	5msflag	Repeat flag (bit 15)
E	0	link.00	Link to next message in the
A	2	link.01	execution queue.
D	4	link.02	Status
E	6	link.03	Sender process
R	10	link.04	Receiver process
	12	link.05	Link to previous message (N100 address).
	14	link.06	Function value
D			Parameter list according to
A			function being processed .
T			
A			
P			
A			
R			
T			

Next link :

The two first words of the block is holding the start address of the next block. If start address of the next block is equal to -1, this means end of link. The link address is a byte address.

Status of the block :

Status gives information about the message currently being processed.

0	:	Block free.
1	:	Message to ND5000.
2	:	Message in process. Set by micro program at start of handling the message.
3	:	Answer to ND-100. Set when the micro program is finished handling the message.
4	:	Error return from ND5000.
13B	:	Stopped by MON 501/502.

Sender :

Sender process number. Owner of the message block.

Receiver :

Receiver is the ND5000 process number to receive the block.

Prev link :

Link to previous message buffer in the execution queue. This is a N100 physical address within the bank.

The data part :

Each message between the ND100 and the ND5000 contains a data part. The first word of the data part defines the function to be performed. The different functions require different numbers of parameters to be involved in the data part of the link.

Function value and their related functions used are :

Value(oct) Function

1	:	Read micro program version.
10	:	Logical data memory read.
11	:	Logical data memory write.
12	:	Clear cache.
22	:	Start process 0
23	:	Start
24	:	Restart after monitor call.
25	:	Restart after trap.
26	:	Restart process with write back of a buffer.
30	:	Physical segment read.
31	:	Physical segment write.
34	:	Logical instruction memory read.
35	:	Logical instruction memory write.
42	:	Programmed trap.
44	:	Histogram read.
45	:	Clear cache & TSB
46	:	Dump dirty
47	:	Go IDLE.
50	:	Restart UNIX.
51	:	Restart UNIX after monitor call.
52	:	Restart UNIX process.
70	:	Initialize Trace module.
71	:	Clear Trace module.
72	:	Arm Trace module.
73	:	Disarm Trace module.
74	:	Dump Trace module.
75	:	Clear Address counter of the Trace module.
76	:	Set cache modus.
77	:	Dump SRF.

