

The ND Emulator project

Version: Draft 1

Carl-Victor Sundling
Halden, Norway

sundling@halden.net
www.haldens.net

ND Emulator User Manual

1	INTRODUCTION	6
1.1	Getting Started/Installation.....	6
1.2	Mandatory to know	8
1.2.1	BG-programs and RT-programs.....	8
1.2.2	The file system	8
1.2.3	File indexing “NO SUCH PAGE”	9
1.2.4	Terminal break strategy.....	9
1.2.5	Parity	9
1.2.6	Users.....	10
1.2.7	2-bank programs.....	10
1.2.8	Reentrant subsystems	10
1.2.9	Floating point	10
1.2.10	Misc.....	10
1.2.11	Instructions not emulated	10
2	THE EMULATOR COMMAND INTERFACE	11
2.1	upper-case <on/off>.....	11
2.2	log-in-user <user name>	11
2.3	list-users.....	11
2.4	trace-execution <on/off>	11
2.5	trace-monitor-calls <on/off>.....	2
2.6	trace-stack <address>	2
2.7	begin-trace-execution-when-at <address>	2
2.8	define-trace-area <trace no.> <lower> <upper>	2
2.9	reset-trace-area <trace number>	2
2.10	open-trace-file <output file>.....	2
2.11	close-trace-file	3
2.12	set-break-point <address>	3
2.13	clear-breakpoint	3
2.14	continue-after-breakpoint	3
2.15	define-histogram <start-address> <increment>	3

2.16	list-histogram [optional output file].....	3
2.17	guard <location>.....	3
2.18	report-memory-reference <location>	3
2.19	reset-report-memory-reference.....	3
2.20	look-at-memory <page table> <address>.....	3
2.21	find-value-in-memory <value> <page table>.....	4
2.22	find-nonzero-in-page-table <page table no>	4
2.23	fill-memory <value> <from> <to>	4
2.24	copy-memory <from> <bytecount> <ntimes>	4
2.25	set-register-value <register> <value>	5
2.26	list-register-values	5
2.27	disassemble-memory <output file> <from addr> <to addr>.....	5
2.28	status.....	5
2.29	dummy.....	5
2.30	! <image file>	5
2.31	load-image-file <image file>	6
2.32	load-byte-swapped-image-file <image file>	6
2.33	set-memory-limits <lower> <upper>.....	6
2.34	set-2bank-mode <on/off>.....	7
2.35	activate-alternative-page-table <APT no.>	7
2.36	save-image-to-file <image file> <SA> <RA>.....	7
2.37	dump-memory <prog.file> <start address> <restart address>.....	7
2.38	dummy2.....	7
2.39	run <start-address>.....	8
2.40	go <address>	8
2.41	continue-at-restart-address	8
2.42	help <command> <output file>.....	8

2.43	open-scratch-file	8
2.44	check-files.....	8
2.45	use-7bit-on-symb-files-write <on/off>	9
2.46	float-32.....	9
2.47	float-48 **** Not currently working ***	9
2.48	compare-images <image file> <image file> [report file].....	9
2.49	add-parity <input file> <output file>	9
2.50	remove-parity <input file> <output file>	9
2.51	byte-swap <inp.file> <outp.file>	10
2.52	od <file>.....	10
2.53	octal-to-decimal <number>	10
2.54	decimal-to-octal <number>	10
2.55	float-test <val1> <val2>	10
2.56	mode <input file>.....	10
2.57	exit.....	12
2.58	quit.....	12
2.59	debug <level(0=off)>	12
2.60	zz	12
2.61	assemble-to-memory <start address>.....	12
2.62	READ-BINARY <input file>	12
2.63	WRITE-BINARY <output file>.....	12
2.64	SINTRAN-III commands	12
2.64.1	@set-terminal-type <n>	12
2.64.2	@list-open-files	13
2.64.3	@close-file <file number (-1=all)>	13
2.64.4	@list-file <file name>	13
2.64.5	@new <user name>.....	13
2.64.6	@who	13
3	STOPPING A RUNNING ND APPLICATION	14

4	PLANNED DEVELOPMENT	14
4.1	Load BPUN-files.....	14
4.2	Terminal emulator	14
4.3	Reentrant subsystems	14
4.4	Monitor calls emulated	14
4.4.1	Nonconformance monitor calls	14
5	LIST OF EMULATOR COMMANDS	14
5.1.1	The PREP2B program.....	16

1 INTRODUCTION

The ND Emulator can be used freely for non-commercial purposes and may be redistributed provided the receiver is made aware of this. The user manual can be copied without any restrictions.

The ND Emulator has been developed in the 'C' programming language partly using Linux and partly using Windows. The Windows version is running as a 32 bit application without a graphical user interface and is controlled via an interactive user interface in a DOS window. Users of SINTRAN will recognize the command interface with abbreviation of the commands and prompts for missing parameters.

The aim of this emulator project was to emulate the instruction set to see how fast an emulator could execute compared to a ND computer. The next step was to be able to run ND programs like the MAC assembler, an early version of the Fortran compiler (FTN) and QED. To be able to do this it was necessary to also emulate many SINTRAN III monitor calls and make an interface to the Windows and Linux file systems.

The emulator has been tested using the following ND programs:

- FTN
- FORTTRAN-100-G02
- MAC
- FMAC
- QED
- NRL
- BRF-LINKER
- LOOK-FILE
- NPL
- LIST-ALL-FILES
- PED

Though some testing has been carried out with these programs the author does not guarantee that the emulator is bug-less. If you get any problems or find errors please do not hesitate to report it to this mail-address:

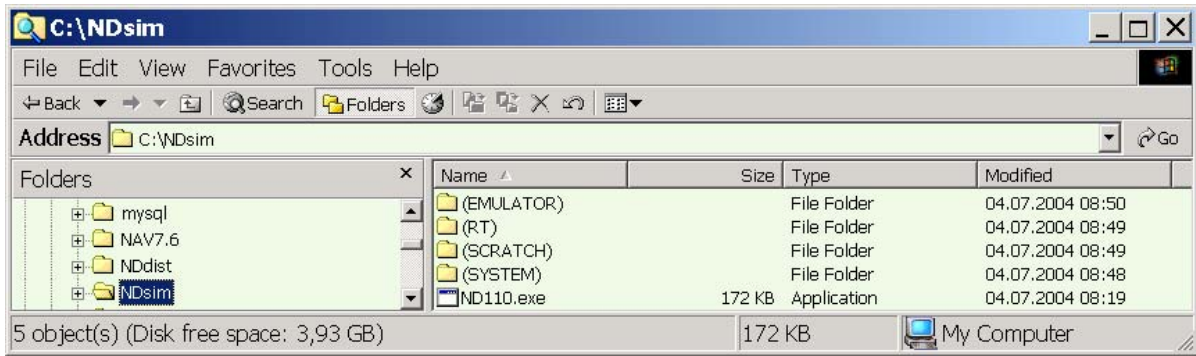
nd-emulator@haldens.net

Speed? Try yourself and be surprised!

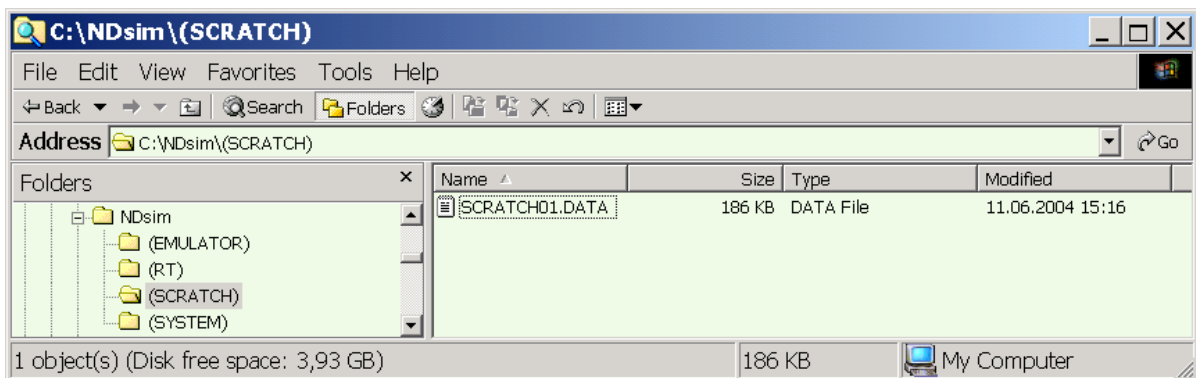
1.1 Getting Started/Installation

NOTE: All folder names and file names (except Ndsim and ND110.exe) should have upper case letters only!

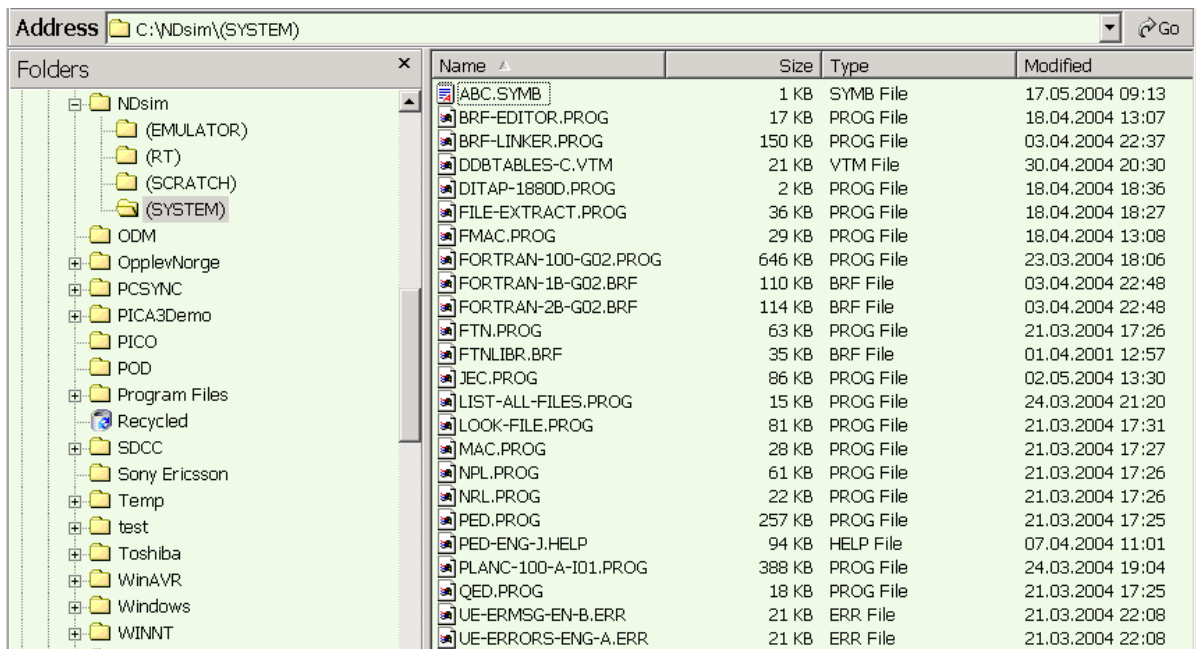
Create a folder called Ndsim. In this folder create the folders as seen in the picture below:



Put the ND110.exe file in the NDsim folder.
 Create the file SCRATCH01.DATA in the folder (SCRATCH) as seen in the picture below:



Install the ND executables and their system files (like PED-ENG-J.HELP) in the (SYSTEM) folder.



Create a shortcut to the ND110.exe file on your desktop and double click it to start the emulator. The emulator is started in a DOS window and you are automatically logged in as user EMULATOR. Typing QED should now start the QED editor

1.2 Mandatory to know

Read this section before you try to find SINTRAN-III functions that are not emulated.

1.2.1 BG-programs and RT-programs

The emulator is only running background programs, RT programs and most of the monitor calls associated with RT is not implemented. The RT-loader does not exist in the emulator and you can not look at segments.

1.2.2 The file system

Most of the monitor calls for doing file-IO is emulated, some monitor calls that reads directory entries etc. are not emulated. You may open files for block access or sequential access and create new files. Direct access is not handled as in SINTRAN-III, but the user should not notice any difference.

The monitor calls for setting block size, reading no. of bytes in file etc. is implemented.

The emulator handles different users, i.e. file names can be (USER)FILE:TYPE and the open monitor calls cope with abbreviated user, file and type as in SINTRAN-III.

Directory names, however, are not emulated. Any attempt to use a (DIR:USER..... notation will cause problems.

Creating new files using "FILE" works.

The SINTRAN-III file version concept is not emulated, do not introduce ; in the file names!

The file extension is using . as separator, not : (which is illegal in Windows file names). The ND applications should still use : as file separator, the emulator will automatically substitute it with . when required.

1.2.3 File indexing “NO SUCH PAGE”

In SINTRAN-III it is possible to have files with “holes”, that is: You may write page 1 and page 99 of a file for example and access these randomly. The emulator does not emulate the SINTRAN-III file system with file indexing, it is using the native Windows/Linux file system which can not handle “holes”. When writing e.g. page 99 of a file in the emulator the file will get 99*1024*2 bytes and afterwards it will be legal to access e.g. page 10, even if it has not really been written by the emulator. As long as your program does not use the “NO SUCH PAGE” error return to make decisions in the program logic, you will not have any problems with this nonconformance with SINTRAN-III. If you try to read beyond the maximum number of bytes, you will get an error return “NO SUCH PAGE” as in SINTRAN. The “hole” syndrome also makes problems when transferring files from a ND-computer to Windows or Linux using ftp. The transfer will terminate because of the NO SUCH PAGE error before all of the file contents is transferred. This is the case for many :DATA files, but also for :PROG files for 2-bank programs.

A program that fills the holes is available and must be run on a ND-computer.

Note: When making :PROG files in the emulator, e.g. using BRF-LINKER, the program files will be created without holes and need not be fixed to be usable in the emulator later on.

1.2.4 Terminal break strategy

In SINTRAN-III you can decide whether your program is restarted or not for each byte typed on the terminal using the BREAK monitor call. This monitor call is also emulated, but the emulator restarts the user program for each byte independent of the break mode. The user program should see no difference. (The reason for implementing the BREAK monitor call in SINTRAN-III was to reduce the context switching when a large number of user worked simultaneously on the same computer. This should be no problem on a GHz computer with one user!) As a consequence of this strategy, the emulator does not buffer input characters from the keyboard, the ISIZE monitor call will therefore always return 0 bytes.

1.2.5 Parity

Many of the ND applications are generating and testing parity on text files. The QED editor is a typical example, you can control parity generation using the MPI(x) and MPO(x) modes. If you want to use e.g. Notepad to edit program files and compile then using e.g. FTN you will get a problem with “PARITY ERROR”.

To cope with this problem use the command “add-parity” which reads the contents of a file, generates even parity and writes the result to another file.

The emulator has a feature that will remove parity on output from an emulated ND program. Ref the command 2.45 use-7bit-on-symb-files-write <on/off>

If you activate this function before starting the ND program, output files will be readable by e.g Notepad afterwards.

The emulator also has a command "remove-parity" that will remove on an existing file.

1.2.6 Users

As mentioned you may have different users. SYSTEM, RT and SCRATCH are mandatory. The user areas are simply folders named (SYSTEM), (RT) and (SCRATCH) and there are no restrictions on the number of pages available. As in SINTRAN the emulator will search for files in SYSTEM's user area if not found at the logged in user. Only SYSTEM may write it's own files, others may read only. Except for user SYSTEM, there are no restrictions for reading and writing files between the users, friends are not emulated.

1.2.7 2-bank programs

The emulator supports 2-bank programs by emulating the Alternative Page Table mechanism. The ALTON and ALTOFF monitor is emulated and bit 0 in the status register may be turned on/off to control which bank to read/write from/to when in ALTON mode. There is no difference in execution time running in 1-bank or 2-bank mode.

1.2.8 Reentrant subsystems

This feature is not supported yet. The PLANC compiler seems to run, but for some commands the REENT monitor call is used and it will stop.

Reentrant subsystems is number 1 priority to be implemented if feasible.

1.2.9 Floating point

The emulator handles both 32 and 48 bit floating point format. (48 bit is currently not working). The emulation is carried out by converting forth and back between the IEEE floating point format used by the ND computers and the IEEE format used by the Intel processors. Add, subtract, divide and multiply are therefore done by HW in the Intel processor. This may lead to some differences in the result of arithmetic operations between a ND computer and the emulator.

The instructions DNZ and NLZ is using a scaling factor that gives a one-to-one result, i.e. 3.000 is converted to 3 and nothing else!

1.2.10 Misc

Spooling files are not implemented.

1.2.11 Instructions not emulated

The instructions that are used for BCD arithmetic are not currently emulated.

Two instructions that are used to copy byte-strings are not currently emulated. (Will be implemented in the next version).

2 The emulator command interface

Using the emulator interface which is command driven should not be a problem for a user known to SINTRAN-III. As in SINTRAN you may abbreviate the command until ambiguous and you will be asked for missing parameters. Some commands are intended to be identical to SINTRAN and can be executed from an application program through the MON 70 monitor calls. Such commands are preceded by an @ and can be listed by typing help @

Please note that the command interpreter is case sensitive (except for the SINTRAN commands).

If the command interpreter does not recognize a command it will try to locate an application program (.PROG file) at the logged in user's area. If it does not find a match there, it will look in user SYSTEM's area before giving up.

To start QED you only need to type QE , the fortran compiler can be started by typing FO-100 etc. (This of course is provided that you have not made any files matching the name in your own user area)

If name a file that by some reason matches a command in the emulator, it can be started by typing !name.

Some of the commands have been introduced to be used for developing and debugging the emulator program and may not be very useful for the "normal" user.

2.1 upper-case <on/off>

Using this command you can control whether the characters typed on the keyboard are converted to upper case or not before returned to the calling application using the MON 1 monitor call.

It can be useful if you are working with e.g. mac which does not accept lower case letters. In practice this command works like the Caps Lock function on the keyboard.

2.2 log-in-user <user name>

When entering the emulator you are logged in as user EMULATOR.by default. Use this command to change user.

2.3 list-users

This command will list all known users in the DOS-window.

```
<>li-users
These users are known:
SCRATCH
EMULATOR
RT
SYSTEM
<>
```

2.4 trace-execution <on/off>

This command turns on or the execution trace function in the emulator. In trace mode the emulator will print the values of all the registers for each instruction executed. It

also indicates which type of instruction that is executed. It is possible to start tracing when reaching a point in the application and it is possible to define trace areas as well. Trace output is directed to the DOS window where the emulator is running unless opening a trace file.

The trace output may be written to a file which may be investigated e.g. by Word Pad, Notepad or any other text processor program.

Note 1: Turning trace off does not close an open trace file.

Note 2: Turning on trace will influence the execution time.

2.5 trace-monitor-calls <on/off>

Turn on or off the tracing of monitor calls. The emulator will report all monitor calls in the emulator window or on the trace file together with the values of all registers.

Note: The monitor calls MON 1, MON 2, MON 71 and MON 72 are not traced because they are frequently used and generate a lot of output.

2.6 trace-stack <address>

Not implemented.

2.7 begin-trace-execution-when-at <address>

The trace function will be activated (and stay active) when the P-register reached the instruction at the value indicated by <address> which should be an octal number between 0 and 177777.

2.8 define-trace-area <trace no.> <lower> <upper>

Defines an area where the trace is active. Several trace areas may be active simultaneously. The upper and lower parameters should be octal numbers.

Tracing is not turned on by this command, you will have to activate tracing by the trace-execution command.

The trace areas that are defined can be listed by the "status" command.

2.9 reset-trace-area <trace number>

Deactivates a defined trace area.

2.10 open-trace-file <output file>

Use this command to open the file where the trace output will be written. Writing trace information is enabled via the trace-execution command. The file remains open until explicitly closed or when terminating the emulator program.

To create a new file embrace the file name by "xxxx".

The default file extension type is .txt and the file will be created in the ND110 folder, not in the logged in user area. To put it elsewhere use standard ../ notation, the command interpreter does not use SINTRAN-III notation.

2.11 close-trace-file

Closes the trace output file. The trace output will now appear in the command interpreter's DOS window.

2.12 set-break-point <address>

Set breakpoint at <address> which should be an octal number between 0 and 177777. The emulator will stop execution when reaching a breakpoint. Only one breakpoint can be defined. To continue after a breakpoint use the command continue-after-breakpoint.

2.13 clear-breakpoint

Clear a defined breakpoint.

2.14 continue-after-breakpoint

Continue execution after having stopped at a breakpoint.

2.15 define-histogram <start-address> <increment>

Not implemented!

2.16 list-histogram [optional output file]

Not implemented!

2.17 guard <location>

Using this command the emulator will report whenever the contents of the memory address at <location> changes value. <location> should be an octal number: You may have up to 20 guards active simultaneously.

Guards are working in page table 0 only.

Guards cannot be reset but are cleared when a program is loaded into memory.

2.18 report-memory-reference <location>

Using this command the emulator will report whenever the contents of the memory address at <location> changes value. <location> should be an octal number between 0 and 3177777. Only one report can be active. (works in all page tables)

2.19 reset-report-memory-reference

Deactivates the defined report of a memory reference.

2.20 look-at-memory <page table> <address>

This command is similar to the SINTRAN-III LOOK-AT-MEMORY command. The page table should be 0 or 2 normally which is always used for BG-programs.

The address should be a number between 0 and 177777 octal. (negative numbers are not accepted).

Example:

```
<>lo-im qed
```

```
Program low = 0 high = 20435 start = 0 restart = 1  
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
```

```
JMPi 125002 @ PC=      0 T=      0 A=      0 D=      0 X=      0 L=      0 ST=      0 B=      0  
<>loo 0 0
```

```
Enter value (octal) or  
"text string" or  
memory address (octal value): (or /) to set location  
Enter q or . and return to terminate
```

```
0 : 125002 ASCII * (with parity bit on) CntrlCh >  
1 : 125002 ASCII * (with parity bit on) CntrlCh >  
2 : 3770 CntrlCh ASCII x (with parity bit on) >  
3 : 4254 CntrlCh ASCII , (with parity bit on) >  
4 : 0 CntrlCh CntrlCh >  
5 : 0 CntrlCh CntrlCh >  
6 : 0 CntrlCh CntrlCh >  
7 : 0 CntrlCh CntrlCh >  
10 : 21 CntrlCh CntrlCh >  
11 : 1302 CntrlCh ASCII B (with parity bit on) >
```

Typing return advances to the next address.

Typing a value (octal) deposits the value in the current location and advances to next location.

Typing a value (octal) followed by / sets current location to the value.

Typing a . or q terminates examination mode.

Note that when 177777 is reached, the next location examined will be | the next page table.

Note: Entering strings using "abcdefg" is not implemented.

2.21 find-value-in-memory <value> <page table>

List all the address of values that match <value> in the specified page table. <value> should be an octal number.

2.22 find-nonzero-in-page-table <page table no>

Find and list the memory addresses of all addresses that contains non-zero values. (Used for development purposes)

2.23 fill-memory <value> <from> <to>

Fill the specified memory area with the given value. All numbers should be octal. A simple performance test can be done by fill-mem 0 0 177777 and then typing go 0. After 5 seconds type ESC and the number of instructions and the time pr. instruction is printed.

2.24 copy-memory <from> <bytecount> <ntimes>

Copies <bytecount> bytes starting at memory position <from> the number of times specified by <ntimes>. The memory position should be an octal number.

Note: Works in page table 0 only. (?)

2.25 set-register-value <register> <value>

Set the specified ND CPU register to <value> which should be an octal number. Legal registers are T A D X B L S P (lower case acceptable).

If <register> equals * then all registers are set to <value>

Example:

```
<>set-reg * 0

(bits in status register not influenced when setting other registers)
Status before:
LDX 54011 @ PC= 20145 T= 15 A= 0 D= 1 X= 55 L= 7370 ST= 142 B= 276
Status now:
LDX 54011 @ PC= 20145 T= 0 A= 0 D= 0 X= 0 L= 0 ST= 0 B= 0
<>
```

2.26 list-register-values

List the values of all registers.

Example:

```
<>li-reg

LDX 54011 @ PC= 20145 T= 15 A= 0 D= 1 X= 161 L= 7370 ST= 140 B= 276
<>
```

2.27 disassemble-memory <output file> <from addr> <to addr>

Not implemented.

2.28 status

Lists information about register values, active trace areas etc.

Example:

```
<>stat

CPU status is: 32 bit ND float format, defined program memory limits are 0 - 20435
LDX 54011 @ PC= 20145 T= 15 A= 0 D= 1 X= 161 L= 7370 ST= 140 B= 276
Status bits: K= 0 Z= 0 Q= 0 O= 1 C= 1 M= 0 SSTG= 0 SSPTM= 0
Tracing memory references of address 123
Trace area number 1: 100 - 1777
<>
```

2.29 dummy

Has no function.

2.30 ! <image file>

Use this command to start a ND program file in the emulator if the name of the program file you want to start is identical to a command name. If you for example make a program file with the name HELP.PROG, the command interpreter would recognize "help" as the command that lists matching commands of the emulator.

Typing HELP would start the program since the command interpreter is case sensitive. (The ! feature was introduced at a stage in the development of the emulator when prog-files also was case sensitive)

Example:

```
<>! qed
Program low = 0 high = 20435 start = 0 restart = 1
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
JMPi 125002 @ PC= 0 T= 15 A= 0 D= 1 X= 161 L= 7370 ST= 140 B= 276
Starting execution at start address 0
```

QED 4.3

*

If the file to execute is not found in the logged in user's area, user SYSTEM will be investigated to see if the file exist there.

Explicitly the user name may be specified:

```
<>! (SYSTEM)NRL
Program low = 0 high = 24631 start
Loaded OK, memory limits (bank 0) a
JMPi 125002 @ PC= 0 T= 15 A
Starting execution at start address
```

RELOCATING LOADER LDR-1935J

*

Note that the program file name is not case sensitive.

2.31 load-image-file <image file>

This command loads a program file into memory, but does not start execution. The parameter <image file> specifies a ND executable :PROG file.

If the file to execute is not found in the logged in user's area, user SYSTEM will be investigated to see if the file exist there.

Explicitly the user name may be specified:

Example:

```
<>lo-im QED
Program low = 0 high = 20435 start = 0 restart = 1
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
JMPi 125002 @ PC= 0 T= 15 A= 0 D= 1 X= 55 L= 7370 ST= 140 B= 276
<>
```

2.32 load-byte-swapped-image-file <image file>

This command is identical to the "load-image" command, except for that it expects the image file to be "byte swapped", that is the byte ordering of the ND file has not been fixed to match the Intel architecture. This command can be used if you by some reason have problems with transferring binary files from the a ND computer.

2.33 set-memory-limits <lower> <upper>

Use this command to set the memory limits of the program area. The values of the lower and upper parameters should be octal numbers between 0 and 177777. The memory limits are used when dumping memory to file, but only when in 1-bank mode.

2.34 set-2bank-mode <on/off>.

Set the emulator in 2-bank or 1-bank mode. The bank mode only influences how the “dump-memory” command will create the ND :PROG file.

Setting 2-bank mode is relevant only when you e.g. make a 2-bank program using MON ALTON in NPL or MAC and dump it. Making program files using the BRF-LINKER is working as it does on a ND computer.

Some times a program file that is transferred from a ND computer is recognized by the emulator as a 2-bank program even if it is a 1-bank program. In these cases the program will be executable without any problems, but an error messages is printed each time it is invoked.

To get rid of the error message do the following:

```
load-image xxx
set-2b off
dump xxx sa ra
```

2.35 activate-alternative-page-table <APT no.>

This command turns on alternative page table (like MON ALTON). It can be used to e.g. test small programs to see how the alternative page table mechanism works. This command was implemented to test the APT mechanism in the emulator.

2.36 save-image-to-file <image file> <SA> <RA>

Identical to “dump-memory”

2.37 dump-memory <prog.file> <start address> <restart address>

Dumps the emulator’s memory to an executable ND file with start and restart addresses that will be used when invoking the program using the ! command. Start and restart address are octal numbers. To create a new ND :prog file, embed the file name in “”.

If the emulator is in 1-bank mode the area in page table 0 defined by the “set-memory-limits” determines the area to be dumped to file.

If the emulator is in 2-bank mode the entire contents of page table 0 and 2 will be dumped to the :PROG file. This is done to cope with the problem of files with “holes”, ref . 1.2.3.

Switching the emulator between 1-bank and 2-bank mode is done using the command “set-2bank-mode <on/off>”.

Using 2-bank mode is only sesible when you make a program that turns on the alternative page table by means of MON ALTON.

2-bank files always use 256 kbytes file space.

2.38 dummy2

Has no function.

2.39 run <start-address>

Start the emulation at the given address which must be an octal number between 0 and 177777.

Identical to "go".

2.40 go <address>

Start the emulation at the given address which must be an octal number between 0 and 177777.

Identical to "run".

2.41 continue-at-restart-address

Continue execution at the restart address (RA) of the currently loaded ND application program.

2.42 help <command> <output file>

List all commands matching <command> to the specified output file. If no outfile is defined, the emulator DOS window will be used for output. To create a new file, embed the file name with "". You may specify a directory path ../, the emulator is not using SINTRAN-III user notation for this command.

Example:

```
<>help load,,,,  
load-image-file <image file>  
load-byte-swapped-image-file <image file>  
<>
```

2.43 open-scratch-file

The scratch file (SCRATCH01:DATA) is opened when starting the emulator. If it is closed by some reason it may be reopened using this command.

2.44 check-files

Checks the files present in the logged in user's area to find files that are accessible as SINTRAN-III files. The example below shows the list of files that are not accepted and which are accepted. Files that are not accepted may be present, but they may not be opened by ND programs.

```
<>log-in sys  
Logged out user EMULATOR and logged in user SYSTEM (user 0)  
  
<>check  
File: FFF-SYMB > File type is missing  
File: A2345678901234567.SYMB > File name is too long  
File: LLL.SYMBX > File type is too long  
  
These files are known for user SYSTEM:  
  
BRF-EDITOR.PROG  
FTNLIBR.BRF  
ABC.SYMB  
BRF-LINKER.PROG  
DDBTABLES-C.VTM
```

```
FILE-EXTRACT.PROG
FMAC.PROG
FORTRAN-100-G02.PROG
FORTRAN-1B-G02.BRF
FORTRAN-2B-G02.BRF
FTN.PROG
LIST-ALL-FILES.PROG
LOOK-FILE.PROG
MAC.PROG
NPL.PROG
NRL.PROG
OOO.SYMB
PED.PROG
PED-ENG-J.HELP
QED.PROG
UE-ERMSG-EN-B.ERR
UE-ERRORS-ENG-A.ERR
LLL.SYMB
KKK.SYMB
<>
```

2.45 use-7bit-on-symb-files-write <on/off>

If 7 bit mode is turned on the emulator will remove the parity bit for each byte when writing to a :SYMB file (works for seq. and random access). This can be convenient if you want to read a file generated by a ND program by e.g. Notepad or Wordpad.

2.46 float-32

Sets the emulator to emulate 32 bit floating point. 32 bit floating point is default.

2.47 float-48 **** Not currently working ****

Sets the emulator to emulate 48 bit floating point.

2.48 compare-images <image file> <image file> [report file]

Compares two ND program files (:PROG) and writes a report of differences in SA, RA, memory limits and the dumped memory area. [report file] is optional. It is possible to compare both 1-bank and 2-bank program files.

The files names are not using SINTRAN-III user names etc., directory paths ../.. may be specified. File name abbreviation is not possible.

2.49 add-parity <input file> <output file>

Reads the input file byte by byte, generates even parity and writes the result to the output file.

The files names are not using SINTRAN-III user names etc., directory paths ../.. may be specified. File name abbreviation is not possible.

2.50 remove-parity <input file> <output file>

Reads the input file byte by byte, clears bit 7 and writes the result to the output file.

The files names are not using SINTRAN-III user names etc., directory paths ../.. may be specified. File name abbreviation is not possible.

2.51 byte-swap <inp.file> <outp.file>

Reads the input file two bytes at the time from the input file and writes them in the opposite order to the output file.

2.52 od <file>

Generates an octal dump of the specified file.

Example:

```
<>od ff
      0 (      0):  11 240 120 322 317 107 322 101 115 240  ..P..G.AM.
     10 (     12): 132 215  12  11 240 322 305 101 314 252  Z.....A..
     20 (     24): 264 240 322 215  12  11 240 322 275 123  .....S
     30 (     36): 311 116  50  63  56 261 251 215  12  11  .N(3.....
     40 (     50): 240 305 116 104 215
<>
```

2.53 octal-to-decimal <number>

Prints the decimal value of the octal <number>.

2.54 decimal-to-octal <number>

Prints the octal value of the decimal <number>.

2.55 float-test <val1> <val2>

Prints the floating point value represented by the two octal numbers as if value1 resided in the A-register and value 2 in the D-register.

Example:

```
<>flo-test 40000 0
Converted to 0.500000
Convert back to ND format: 40000 , 0
<>
```

Note: 32 bit floating point format only.

2.56 mode <input file>

Using this command is similar to the @MODE command in SINTRAN-III, typically used to compile and link programs etc. The mode file should use lower case letters for the emulator commands which should not be preceded by an @ as in SINTRAN-III mode files.

The emulated SINTRAN-III commands must be preceded by an @, e.g. @li-fi
This command has SINTRAN-III abbreviation on the file name and a user name may be specified, the default file extension is :MODE
You may use e.g. Notepad to generate the mode file.

There is no mode output file.

Example:

Contents of mode file:

```
@li-fi,,,
npl
@FLO48
@DEV SIMULATOR-1,,100
MAC
)9ASSM 100
)9END
)9TSS
set-mem 0 177777
dump "SIM" 0 1
```

Makes this output:

```
<>mode comp
```

```
<>@li-fi,,,
```

```
FILE 0 : (EMULATOR)FFF:SYMB
FILE 1 : (EMULATOR)ZXC:SYMB
FILE 2 : (EMULATOR)NMN:SYMB
FILE 3 : (EMULATOR)EEE:SYMB
FILE 4 : (EMULATOR)YYY:SYMB
FILE 5 : (EMULATOR)WWW:SYMB
FILE 6 : (EMULATOR)SIMULATOR-1:SYMB
FILE 7 : (EMULATOR)SIMULATOR-2:SYMB
FILE 8 : (EMULATOR)JJJ:SYMB
FILE 9 : (EMULATOR)JJJ:BRF
FILE 10 : (EMULATOR)FTEST:BRF
FILE 11 : (EMULATOR)JJJ:PROG
FILE 12 : (EMULATOR)BM:SYMB
FILE 13 : (EMULATOR)BMDAC:BRF
FILE 14 : (EMULATOR)BMDAC:PROG
FILE 15 : (EMULATOR)BM:BRF
FILE 16 : (EMULATOR)BM:PROG
FILE 17 : (EMULATOR)WW:BRF
FILE 18 : (EMULATOR)WW:PROG
FILE 19 : (EMULATOR)ABC:SYMB
FILE 20 : (EMULATOR)ABC-2:SYMB
FILE 21 : (EMULATOR)ABCD:SYMB
FILE 22 : (EMULATOR)SIM:PROG
FILE 23 : (EMULATOR)COMP:MODE
```

```
<>npl
```

```
No such command: npl, try to load and run image file
```

```
Program low = 0 high = 74215 start = 0 restart = 1
```

```
Loaded OK, memory limits (bank 0) are now set to 0 - 74215
```

```
JMPi 125002 @ PC= 0 T= 1 A= 12 D=164634 X=172512 L=166267 ST= 240 B=164173
```

```
Starting execution at start address 0
```

```
NORD PL NOVEMBER 1979
```

```
@FLO48
```

```
@DEV SIMULATOR-1,,100
```

```
WARNING: SUBROUTINE ENTRYPOINT CHANGED TO ,SA
```

```
- END OF COMPILATION
```

```
0 ERRORS DETECTED
```

```
***** ND program terminates with MON 0 at 56374 *****
```

```
__ ND-110 emulation halted, close files (except scratch file):
```

```
100: ./((SCRATCH)/SCRATCH01.DATA Opened for access 2, block size 512 bytes
```

```
Time used 9 secs, 9844.000000 milsecs
```

```
77026505 instructions executed, 0.127800 usec/instruction
```

```
<>MAC
```

```
No such command: MAC, try to load and run image file
```

```
Program low = 145000 high = 177777 start = 177777 restart = 177775
```

```
Loaded OK, memory limits (bank 0) are now set to 145000 - 177777
```

```
JMPi 125377 @ PC=177777 T= 0 A= 100 D=161142 X= 55214 L= 56374 ST= 40 B= 54126
```

```
Starting execution at start address 177777
```

```
- MAC -
)9ASSM 100
**** 000000 DIAGNOSTICS ****

)9END
)9TSS

***** ND program terminates with MON 0 at 173411 *****

__ ND-110 emulation halted, close files (except scratch file):
100: ./(SCRATCH)/SCRATCH01.DATA Opened for access 2, block size 512 bytes

Time used 6 secs, 6049.000000 milsecs
45623488 instructions executed, 0.132585 usec/instruction
<>set-mem 0 177777

Memory limits are now      0 - 177777
<>dump "SIM" 0 1
```

2.57 exit

Terminates the emulator program.

2.58 quit

Terminates the emulator program.

2.59 debug <level(0=off)>

If set to a nonzero (positive) value the emulator will print some debug information, mostly from the emulation of the monitor calls.

Used for emulator debugging purposes.

2.60 zz

Do not use this command!

2.61 assemble-to-memory <start address>

Not implemented.

2.62 READ-BINARY <input file>

Not implemented.

2.63 WRITE-BINARY <output file>

Not implemented.

2.64 SINTRAN-III commands

2.64.1 @set-terminal-type <n>

Use this command to set the terminal type.

PED is dependant of recognizing a terminal type.

At start the emulator sets terminal type 0.

2.64.2 @list-open-files

Lists the files that are currently open. When issued as a command from the emulator command interpreter only the scratch file should be open.

```
<>@li-op
    100: ./(SCRATCH)/SCRATCH01.DATA Opened for access  2, block size 512 bytes
<>
```

When issued from a ND program all files currently open will be listed.

```
PED:@li-op
    100: ./(SCRATCH)/SCRATCH01.DATA Opened for access  2, block size 2048 bytes
    101: ./(SYSTEM)/UE-ERRORS-ENG-A.ERR Opened for access  2, block size 2048 bytes
PED:
```

The format is somewhat different from SINTRAN-III

2.64.3 @close-file <file number (-1=all)>

Closes a specified file where file number typically is 100, 101 ... (octal). If file number is -1 all open files except the scratch-file will be closed.

2.64.4 @list-file <file name>

List all files matching <file name> in the DOS window.

```
<>@li-fi (sys)fo,,,
FILE 7 : (SYSTEM)FORTRAN-100-G02:PROG
FILE 8 : (SYSTEM)FORTRAN-1B-G02:BRF
FILE 9 : (SYSTEM)FORTRAN-2B-G02:BRF
<>
```

2.64.5 @new <user name>

Works similar to the NEW command in SINTRAN-III, except for that user RT also may switch to any user as SYSTEM can. When starting the emulator user EMULATOR is logged in. To change to user SYSTEM use the command log-in-user.

2.64.6 @who

Prints the current user:

```
<>@who
===>  1  EMULATOR
<>
```

3 Stopping a running ND application

The ESC button will terminate the running application if ESC has not been disabled by the ND monitor call for this purpose, but only if the program comes to a MON 1 instruction. (In SINTRAN-III it will be stopped unconditionally)

To stop a program that goes into an endless loop, type cntl-C on the keyboard. (cntl-C will not terminate the application if it is currently in MON 1).

If unable to stop using any of these methods simply kill the DOS window.

4 Planned development

4.1 Load BPUN-files

A command for reading :BPUN files as in SINTRAN-III.

4.2 Terminal emulator

A Tandberg TDV terminal emulator making it possible to e.g. run PED with terminal type number 51 or 53.

4.3 Reentrant subsystems

Emulate the monitor call that enables reentrant subsystems.

4.4 Monitor calls emulated

```
MON050(), MON043(), MON062(), MON117(), MON035(), MON026();  
MON071(), MON072(), MON312(), MON033(), MON034(), MON076();  
MON007(), MON010(), MON073(), MON334(), MON113(), MON114();  
MON005(), MON006(), MON032(), MON042(), MON064(), MON004();  
MON117(), MON120(), MON020(), MON015(), MON025(), MON027();  
MON322(), MON263(), MON321(), MON016(), MON066(), MON114();  
MON012(), MON022(), MON074(), MON114(), MON144(), MON070();  
MON077(), MON162(), MON041(), MON214(), MON262(), MON323();  
MON067(), MON104(), MON017(), MON310(), MON065(), MON044();  
MON241(), MON242(), MON256(), MON003(), MON024(), MON063();  
MON021(), MON023();
```

4.4.1 Nonconformance monitor calls

5 List of emulator commands

```
upper-case <on/off>  
close-trace-file  
open-trace-file <output file>  
trace-execution <on/off>  
look-at-memory <page table> <address>  
dummy
```

```
run <start-address>
set-memory-limits <lower> <upper>
save-image-to-file <image file> <SA> <RA>
copy-memory <from> <bytecount> <ntimes>
define-trace-area <trace no.> <lower> <upper>
reset-trace-area <trace number>
status
report-memory-change <address>
set-break-point <address>
clear-breakpoint
continue-after-breakpoint
dummy2
help <command> <output file>
fill-memory <value> <from> <to>
assemble-to-memory <start address>
quit
load-image-file <image file>
find-value-in-memory <value>
step-execution [optional start address]
define-histogram <start-address> <increment>
list-histogram [optional output file]
float-32
reset-report-memory-reference
set-register-value <register> <value>
exit
list-register-values
disassemble-memory <output file> <from addr> <to addr>
float-48
trace-monitor-calls <on/off>
READ-BINARY <input file>
WRITE-BINARY <output file>
trace-stack <address>
begin-trace-execution-when-at <address>
load-byte-swapped-image-file <image file>
guard <address>
use-7bit-on-symb-files-write <on/off>
mode-input-file <file name>
continue-at-restart-address
! <image file>
octal-to-decimal <number>
decimal-to-octal <number>
compare-images <image file> <image file> [report file]
add-parity <input file> <output file>
remove-parity <input file> <output file>
float-test <val1> <val2>
activate-alternative-page-table <APT no.>
find-nonzero-in-page-table <page table no>
@set-terminal-type <n>
set-2bank-mode <on/off> <page table>
go <address>
dump-memory <prog.file> <start address> <restart address>
```

```
od <file>
byte-swap <inp.file> <outp.file>
@list-open-files
@close-file <file number (-1=all)>
open-scratch-file
debug <level(0=off)>
log-in-user <user name>
zz
list-users
@list-file <file name>
check-files
@new <user name>
@who
```

5.1.1 The PREP2B program

When transferring files from the ND computer e.g. using ftp an error will occur if the files has “holes”, i.e. missing pages. If a file for example has page 0,1,2 and 10 written, pages 3,4,5,6,7,8 and 9 is not accessible. This is the way the SINTRAN-III file system works, while Windows and Linux does not allow files with “holes” in them. The emulator does not emulate the SINTRAN-III file system, which means that it is actaull possible to access a file written by an emulator program even if the page is not written. If you for example open a file and write page 10, you will be able to read pages 0-9 as well number 10.

To be able to transfer files using ftp from a ND computer it will need some preparation if it has missing pages. Typically 2-bank program files need to be prepared, other files like :DATA files can be candidates for preparation as well.

The program PREP2B reads a file and writes the contents to another file checking for empty pages in the input file and writing dummy data if found.

The program looks like this:

```
PROGRAM RFIL
INTEGER*2 IDATA(1024)
INTEGER*2 NPAGES
INTEGER*4 NBYTES
C INTEGER ERRCODE
INTEGER*2 IFILE, OFILE
CHARACTER*80 INNFILE, OUTFILE
OUTPUT(1)'This program reads a :PROG file dumped in 2-bank mode'
OUTPUT(1)'and writes it to the specified output file while'
OUTPUT(1)'filling in the missing file pages that cause problems'
OUTPUT(1)'when trying to ftp these type of files (NO SUCH PAGE)'
OUTPUT(1)'Output file size will be 129 pages ~ 250 kbytes'

WRITE(1)6412B,'Enter input file name: '
INPUT(1)INNFILE
WRITE(1)6412B,'Enter output file name: '
INPUT(1)OUTFILE

IFILE = 7
OFILE = 8
```

```
OPEN(IFILE,STATUS='OLD',FILE=INNFILE, ACCESS = 'RX'  
C , ERR=666)  
CALL RMAX(IFILE,NBYTES)  
output(1)'Bytes in input file: ', NBYTES  
OPEN(OFILE ,FILE=OUTFILE, ACCESS = 'WX', ERR= 777)  
CALL SETBS(OFILE,1024)  
CALL SETBS(IFILE,1024)  
  
IF ( ERRCODE .NE. 0 ) THEN  
    OUTPUT(1)'SETBS error'  
ELSE  
    NPAGES = (NBYTES+1)/2048  
    NPAGES = NPAGES + 1  
    OUTPUT(1)'Total pages to write: ',NPAGES  
    DO FOR I = 1,NPAGES  
        CALL RFILE(IFILE,0,IDATA,I-1,1024)  
        IF ( ERRCODE .NE. 0 ) THEN  
            OUTPUT(1)'No data found bl.no : ',I-1,  
C            ' ..write data to fill missing page'  
            DO FOR J = 1,1024  
                IDATA(J) = 333  
            ENDDO  
            CALL WFILE(OFILE,0,IDATA,I-1,1024)  
            IF ( ERRCODE .NE.0 ) THEN  
                OUTPUT(1)'Write failed on bl.no: ',I-1  
                OUTPUT(1)'Terminate! (check user space?)'  
                CLOSE(IFILE)  
                CLOSE(OFILE)  
                GO TO 9999  
            ENDIF  
        ELSE  
            OUTPUT(1)'Success on block: ',I-1  
            CALL WFILE(OFILE,0,IDATA,I-1,1024)  
            IF ( ERRCODE .NE.0 ) THEN  
                OUTPUT(1)'Write failed on bl.no: ',I-1  
                OUTPUT(1)'Terminate! (check user space?)'  
                CLOSE(IFILE)  
                CLOSE(OFILE)  
                GO TO 9999  
            ENDIF  
        ENDIF  
    ENDDO  
ENDIF  
ENDIF  
CLOSE(IFILE)  
GO TO 9999  
  
666 OUTPUT(1)'OPEN input file', INNFILE,' failed'  
GO TO 9999  
777 OUTPUT(1)'OPEN output file', OUTFILE,' failed'  
CLOSE(IFILE)  
9999 OUTPUT(1)'PROGRAM END'  
END  
EOF
```