PROGRAM DESCRIPTION

Product	Name	Reg. no.	Category
	ND PASCAL for ND-500/5000	211003B	STPR

Reason

. New product
X Change/Addition
X Error Correction
. Different Environment

Documentation

Title	Reg. no.
PASCAL Reference Manual	60.222.1 EN

Purpose

Compiler and Runtime System for programming language PASCAL for ND-500 32-bit and 64-bit real format.

Prerequisites

Computer	Type	Floating format	Op. system Version
ND-500	All	All	SINTRAN III -1

Reg. no.	Product name
210319F	LINKAGE-LOADER for ND-500

Minimum mass storage resources for installation

User	Userspace	Number of files
SYSTEM	400 pages on	7 files
DOMAIN-USER	320 pages on	15 files

Minimum permanent mass storage resources

User	Userspace	Number of files
SYSTEM	350 pages on	5 files
DOMAIN-USER	320 pages on	15 files

Reg. no. for manual
250157B

FILES INCLUDED ON THE SYSTEM DISKETTES:

File Name	Type	Containing
INST-PASC500-B<rev>	PROG	Installation program
PASC-PROLOG-B<rev>	PASC	Header file
PASC-MONITOR-B<rev>	PASC	Monitor include file
PASC-SIBAS-B<rev>	PASC	SIBAS include file
PASC-FOCUS-B<rev>	PASC	FOCUS include file
PASC-ISAM-B<rev>	PASC	ISAM include file
CAT-PASERR-B<rev>	ERR	Error messages
CAT-LIB-B<rev>	NRF	RTS-library 64-bit
F32CAT-LIB-B<rev>	NRF	RTS-library 32-bit
PASCAL-DOMAIN5	MODE	Standard-C main module
PASCAL-B<rev>	PSEG	PASCAL User-Interface
PASCAL-B<rev>	DSEG	
PASCAL-B<rev>	LINK	
CAT-CROSS-B<rev>	PSEG	Source Formatter
CAT-CROSS-B<rev>	DSEG	
CAT-CROSS-B<rev>	LINK	
CAT-PASC-B<rev>	PSEG	PASCAL Front-End
CAT-PASC-B<rev>	DSEG	
CAT-PASC-B<rev>	LINK	

Product	Name	Reg. no.	Category
	ND PASCAL for ND-500/5000	211003B	STPR

CAT-CAT5-B<rev>	FSEG	CAT - Compiler ND-500
CAT-CAT5-B<rev>	DSEG	
CAT-CAT5-B<rev>	LINK	

NOTE: <rev> is to be replaced by the current revision of the DIRECTORY or FILE.

The attached Software INFORMATION Reports should be read before installation.

1 SUMMARY

This new version of PASCAL offers considerable benefits. In particular this version is able to handle strings as defined in the ISO standard.

Another major change is the facility to produce symbolic debug information. This enables the user to test the programs on source program level.

2 SYSTEM OVERVIEW

The CAT - PASCAL System consists of the following parts:

1) The compiler, which is able to produce code for 32-bit and 64-bit floating point representation.
   Default is 64-bit real.

2) Runtime libraries for 32-bit and 64-bit floating point representation.
   If you want to produce code for 32-bit real, you have to load the library F32CAT-LIB:NRF instead of CAT-LIB:NRF.

3) A prologue file with several include files, which contain declarations of constants, types, procedures and functions.

3 ERRORS CORRECTED SINCE VERSION A

All errors reported since the last release have been corrected. This concerns in particulary the following areas:

- It is now possible to call the procedures REWRITE and RESET several times after calling the procedure CONNECT.
- Packed structures and sets will no longer cause problems.
- Errors concerning the interface routines to FOCUS and SIBAS are also corrected.
- The random function produces satisfactory results for all floating point representations.
- Some minor errors concerning IO-handling have been eliminated.

4 ERRORS KNOWN BUT NOT CORRECTED

In this version and in all future versions it will not be possible to define the CAT-library as auto-link-segment in the LINKAGE-MAP. The reason for this is that the library contains unresolved references to the ISAM, SIBAS, and FOCUS libraries which must be resolved by loading the appropiate libraries after loading the CAT library.

If no ISAM, SIBAS or FOCUS library is required, you can use the auto-load-file facility instead. The language, which must be specified in the auto-load-file command, must be FORTRAN.

5 MODIFICATIONS SINCE VERSION A

5.1 CHANGED COMMANDS

The following commands have been changed:

cross <source file: >,<cross reference file: >,
<keywords (upper-case / lower-case / no-change): >,
<lines per page: >,<cross reference list (yes / no / only): >
This command produces a source file listing and/or cross reference listing, which is either sent to a peripheral file or to mass-storage file. It is also possible to produce only a cross reference listing.

help <command: >
With a command name as parameter the help command lists the appropiate command together with its parameter.

options <option: >...
There are four new option values:

e+	with language extensions
d+	with symbolic debug
t+	with tagfield check
a+	with complete listing

value <options>
After each option it is specified in brackets how the option can be set:

options :

```
prologue file is (SYSTEM)PASC-PROLOG-B:PASC
target machine is ND-500          ( m2 )
64 bit real                       ( r4 )
intermixing is not allowed        ( x- )
with language extensions          ( e+ )
with line numbers                 ( l+ )
with symbolic debug               ( d+ )
with procedure names              ( n+ )
with subrange check               ( s+ )
with pointer check                ( p+ )
with index check                  ( i+ )
with overflow check               ( o+ )
without tagfield check            ( t- )
without complete listing          ( a- )
page length is 48 lines
```

5.2 NEW COMMANDS

cc
This command is used to write comments into mode or batch files.

check <source file: >,[<list file: >],[<CAT file: >]
This command only checks the syntax and semantic of the specified source file. If no errors are detected, it is possible to generate code with the generate-code command.

format <source file: >,<new source file: >,
<keywords (upper-case / lower-case / no-change): >
This command is used to format the specified source file into a new file. This was integrated in the cross command in the A00-version.

generate-code <CAT file: >,<object file: >
This command is used to generate code from the specified CAT-file.

initialize-compile-parameters [<init file: >]
This command is used to initialize the options and prologue files from an init file. If no file is specified a file with the name PASCAL-B:INIT is used.

library <library file: >...
This command is used to define one or more libraries, which are to be load before the CAT-library when using the link command.

page-length [<lines per page: >]
This command is used to modify the page length, which is used in the cross command and the a+ option. The default page length is 48.

save-compile-parameters [<init file: >]
This command is used to save the options and prologue files into an init file. If no file is specified a file with the name PASCAL-B:INIT is used. When starting the compiler the options will be read in from the default init file, if it exists.

value <libraries>
The value libraries command lists the specified libraries, which are to be load before the CAT library when using the link command.

```
libraries :
  MY-LIBRARY:NRF
```

with <additional prologue file: >...
This command is used to define additional prologue files, which will be read in. These definitions can be deleted with the clear-command.

5.3 IMPROVED PERFORMANCE

The compile time has been improved by the following changes:

- One domain has been eliminated in order to save the time of calling it.
- The compiler itself is compiled without checks, which gives an improvement of approximately 20%.
- Some files for intermediate code have been eliminated; instead the scratch file 100B is used.
- For intermediate files the compiler uses the file-as-segment facility.

5.4 NEW FACILTIES

5.4.1 STRINGS

A new data type STRING has been implemented, which enables the user to have strings of variable length. Internally a variable string is defined as a structure containing a variable current length and a packed character array. This allows the dynamic length of a string to be handled automatically instead of being handled explicitly by the source code.

A variable string is declared as follows:

```
VAR
  str: STRING[n];
```

where n specifies the maximum-length of the string "str". STRING is a reserved word.

A null string, i.e. a string with a current length of zero, may be specified by two adjacent apostrophes (''). String assignment of variable strings is defined such that the target string gets the current length of the source string. It is an error if the current length of the source string exceeds the maximum length of the target string.

If the parameter declaration:

```
  VAR x: STRING;
```

is used in a procedure or function declaration, the procedure or function will accept any variable of STRING.

A substring of a variable string can be obtained by specifying the first and last character position separated by two dots (..).

```
example:
str[2..6] := 'hello';
```

There is one new operator to concatenate strings: the binary operator '&' takes characters, packed character arrays or variable strings as operands and return a variable string containing the concatenation of the operands.

```
example:
str := str & ' hello';
```

The current length of a variable string can be determined by calling the function LENGTH(str), where str is an string-expression.

The maximum length of a variable string can be determined by calling the function MAXLENGTH(str), where str is an string-expression.

There are five new procedures to simplify the use of strings:

```
PROCEDURE append(VAR s1 : STRING; VAR s2 : STRING); EXTERN;
```
Append(s1,s2) appends the string s2 to the the string s1.

```
PROCEDURE delete(VAR s : STRING;
                 i : string_index;
                 j : string_range); EXTERN;
```
Delete(s,i,j) deletes j characters from the string s starting at position i.

```
PROCEDURE fill(VAR s : STRING;
               i : string_index;
               j : string_range;
               ch : char); EXTERN;
```
Fill(s,i,j,ch) fills the substring s[i..i+j-1] with the character ch. If necessary the current-length of s is increased.

```
PROCEDURE insert(VAR s1 : STRING;
                 VAR s2 : STRING;
                 i : string_index); EXTERN;
```
Insert(s1,s2,i) inserts the string s1 into the string s2 at position i in s2.

```
FUNCTION index(VAR s1, s2 : STRING;
               i : string_index) : string_range; EXTERN;
```
Index(s1,s2,i) returns the starting position of the first occurrence of the string s2 in the substring s1[i..length(s1)], or zero if it is not found.

5.4.2 I/O

On the ND-500 it is now possible to open a file as a segment by adding a 'S' to the access type in the CONNECT call.
On the ND-100 this specification will be ignored, so that no source code modification is necessary.

Example:
```
CONNECT(OUTFILE,'MY-FILE','SYMB','WXS');
```

Furthermore, file access has been optimized by using the monitor calls DVINST and DVOUTS when reading or writing from a peripheral device and the optimal access type will be taken when opening a file.

5.4.3 SYMBOLIC DEBUG

In this version symbolic debug information will be produced by using the option d+. This enables the user to test his programs on source program level.

5.4.4 SIBAS INTERFACE

The following calls of SIBAS version F have been implemented:

```
(* forg + remb + get *)
PROCEDURE pssget(
   VAR  temp_db_key : integer2;
        no_of_items : integer2;
   VAR  item_list   :        ARRAY[lb1..ub1:byte2] OF
                      PACKED ARRAY[lb2..ub2:byte2] OF char;
   UNIV item_values : integer2;
   VAR  status      : integer2); EXTERN;

(* synchronized checkpoint *)
PROCEDURE pssync(
        code       : integer2;
   VAR checkpoint  : integer2;
   VAR status      : integer2); EXTERN;

(* get physical record number *)
PROCEDURE pswhat(
        temp_db_key : integer2;
   VAR realm_name   : PACKED ARRAY[lb1..ub1:byte2] OF char;
   VAR record_no    : longint;
   VAR status       : integer2); EXTERN;
```

Product	Name	Reg. no.	Category
	ND PASCAL for ND-500/5000	211003B	STPR

```
(* direct find - 'find using record number *)
PROCEDURE psfrno(
   VAR  realm_name : PACKED ARRAY[lb1..ub1:byte2] OF char;
        record_no  : longint;
   VAR  status     : integer2); EXTERN;

(* sfrno + sget *)
PROCEDURE psfrgt(
   VAR  realm_name    : PACKED ARRAY[lb1..ub1:byte2] OF char;
        record_no     : longint;
        no_of_items   : integer2;
   VAR  item_list     :        ARRAY[lb3..ub3:byte2] OF
                       PACKED ARRAY[lb4..ub4:byte2] OF char;
   UNIV item_values   : integer2;
   VAR  status        : integer2); EXTERN;

(* reading SSI-SEC code and user and log info *)
PROCEDURE psemsg(
        code   : integer2;
   UNIV values : integer2;
   UNIV buffer : integer2;
   VAR  status : integer2); EXTERN;

(* relative get with various options *)
PROCEDURE psrget(
   VAR  temp_db_key : integer2;
   VAR  temp_sr_key : integer2;
        direction   : integer2;
        rem_code    : integer2;
        no_of_items : integer2;
   VAR  item_list   :        ARRAY[lb3..ub3:byte2] OF
                      PACKED ARRAY[lb4..ub4:byte2] OF char;
   VAR  record_no   : longint;
   UNIV item_values : integer2;
   VAR  status      : integer2); EXTERN;
```

5.4.5 **MISCELANEOUS**

A new procedure STOP has been implemented, which gives the user the possibility to abort his program after writing the string str to the terminal.

```
Declaration:
PROCEDURE stop(VAR str   : PACKED ARRAY[lb..ub:byte2] OF char;
               error_no : byte2); EXTERN;
```

The runtime system writes the source line number and the procedure name where the procedure STOP was called and aborts the program. In the case of a RT-program the following monitor-calls will be executed:

```
ERMON (50,error_no);
ERMON (51,<source line number>);
RTEXT;
```

which write the error_no and the source line number to the error device and aborts the RT-program.

The following error codes are used by the runtime system:

```
err_stack_overflow                = 1;
err_heap_overflow                 = 2;
err_subr_or_index_out_of_range    = 3;
err_pointer_with_nil_value        = 4;
err_undefined_pointer             = 5;
err_undefined_case_label          = 6;
err_arithmetic_overflow           = 7;
err_set_element_out_of_range      = 8;
err_illegal_tagfield_value        = 9;
err_division_by_zero              = 10;
err_invalid_operation             = 11;
err_eof_on_input                  = 12;
err_illegal_number_sytax          = 13;
err_too_big_number                = 14;
err_illegal_operand_value         = 15;
err_address_trap                  = 16;
err_connect                       = 17;
err_descriptor_range              = 18;
err_exponent_arg                  = 19;
err_ln_arg_not_greater_0          = 20;
err_sqrt_arg_negative             = 21;
err_index_scaling                 = 22;
err_sin_arg_too_big               = 23;
err_illegal_arg_of_power          = 24;
err_sinh_arg_too_big              = 25;
err_illegal_format_specification  = 26;
err_illegal_instruction_code      = 27;
err_illegal_operand_specifier     = 28;
err_read_without_reset            = 29;
err_write_without_rewrite         = 30;
err_protection_violation          = 31;
err_instruction_sequence          = 32;
err_not_random_file               = 33;
err_dispose                       = 34;
err_release                       = 35;
err_io_error                      = 37;
err_string_index                  = 38;
err_string_length                 = 39;
```

The mathematical functions like SIN, COS, LN, EXP, etc. are implemented in software, because the hardware instructions being used in the A00-version are not very accurate. Nevertheless is it possible to use the hardware instructions by setting the flag for conditional compilation with SET HARDWARE before compiling the program.

Example:
Pascal: SET HARDWARE
Pascal: COMPILE <source file>,<list file>,<destination file>

In this new version the result type of functions may be any type except files or structures containing files.

6 INSTALLATION PROCEDURE

NOTE: As the installation procedure will delete all existing B-version files before installing revision B06, you may wish to take a backup of your old files before proceeding.

- Log in under user SYSTEM.
- Insert the floppy disk with the directory name 211003B06-XX-01 (double density).
- Enter the following commands (operator input is underlined):

```
@ENTER-DIRECTORY ←
 DIRECTORY NAME: 211003B-XX-01 ←
 DEVICE NAME: FLOPPY-DISC-<drive-no.> ←
 DEVICE UNIT: <floppy-unit> ←
@

@(211003B-XX-01:FLOPPY-USER)INST-PASC500-B:PROG,, ←
@
```

The installation procedure will then copy the new files to user SYSTEM and DOMAIN-USER and define the standard domains.

Finally, copy the mode file PASCAL-DOMAINS:MODE into the HEMT-DIR: