N.V.PHILIPS Natuurkundig Laboratorium

NATLAB
NR : TPM
DD :
PAR : EO

TWIN PASCAL

N.V.PHILIPS Natuurkundig Laboratorium

NATLAB
NR : TPM
DD :
PAR : E0

TWIN PASCAL

1. Introduction

This is the description of Twin Pascal. The document contains the language definition and directives for the compilation and execution of programs written in Twin Pascal.

The implemented language is Standard Pascal (see: K. Jensen and N. ''irth, "Pascal User Manual and Report", Springer Verlag, second ˛dition 1978) with a few limitations and certain extensions. It is assumed that the reader is familiar with Standard Pascal.

The main differences between Standard Pascal and Twin Pascal are:

1. Twin Pascal cannot accept "variants" of records, and "procedures" or "functions" may not be used as parameters;

2. Twin Pascal is equiped with the functions "TAN" (tangent) and "RAND" (random number generator);

3. The CASE statement is extended with an ELSE clause;

4. Twin Pascal programs can be composed from separately compiled modules, the Twin Link Editor is able to unite separately compiled modules into a single executable program.

The Twin Pascal compiler generates programs for an abstract stack ›mputer. This abstract computer has instructions to handle integer numbers of 16 bits, real numbers (8 bit exponent and 40 bit mantissa), characters and sets (with up to 64 elements).

The Twin Link Editor will automatically include the program that implements the stack computer on the Signetics 2650.

In the following sections the differences between Twin Pascal and Standard Pascal are described in detail. Also the Twin System commands to use Twin Pascal are discussed.

2. Description of Twin Pascal

Only the differences between Twin Pascal and Standard Pascal are described. Herewith the division of the Pascal Report is followed.

2.1. Notation, terminology and vocabulary

The basic vocabulary of Pascal consists of basic symbols classified into letters, digits and special symbols.

The collection of letters of Twin Pascal is restricted to the capitals.

so the special symbols "left brace" and "right brace" are not available. The symbol pairs '(*' and '*)' are used as synonyms for the braces.

The symbols FORWARD and EXTERN are reserved words in Twin Pascal.

2.2. Identifiers, numbers and strings

Identifiers serve to denote constants, types, variables, procedures and functions. Identifiers denoting distinct objects must differ over their first 8 characters, but only the first 6 characters of external references are significant.

All identifiers, except type identifiers, defined at level 0 are external references in Twin Pascal.

2.3. Data type definitions

A data type determines the set of values which variables of that type may assume and associates an identifier with the type.

Types wich are designated at two or more different places in the program text are identical if the same type identifier is used at these places, or if different identifiers are used which have been defined to be equivalent to each other by type definitions of the form T1 = T2.

Two types are compatible if one of the following statements is true:

- the types are identical.
- one type is a subrange of the other type
- both types are subranges of the same type.
- one type is INTEGER (or a subrange thereof) and the other type is REAL.
- one type is a pointer type and the other type represents the constant NIL.
- the types are string types with the same number of components.
- the types are set types of compatible base-types.

An expression E of type T2 is assignment-compatible with a type T1 if T1 and T2 are compatible and both the following statements are false:

- T1 is a file.
- T1 is INTEGER while T2 is REAL.

2.3.1. Simple types

2.3.1.1. Standard types

The following types are standard in Twin Pascal:

.NTEGER	The values are the subset of the whole numbers in the range 0 - 32767.
REAL	Its values are a subset of the real numbers in the range 0 - 2**127 (0 - 10**38). Their accuracy is about 2**39 (10**11).
BOOLEAN	Its values are the truth values denoted by the identifiers TRUE and FALSE. However, outputs of these values are denoted by the digits 0 and 1 respectively.
CHAR	Its values are a set of characters determined by the ASCII alphabeth, including the space and the characters:

```
   !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
@  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
P  Q  R  S  T  U  V  W  X  Y  Z  [  æ  ]  ´  _
```

2.3.2. Structured types

A structured type is characterised by the type(s) of its components and by its structuring method.

The directive "packed" is ignored by Twin Pascal.

2.3.2.1. Array types

An array type is a structure consisting of a fixed number of components which are all of the same type, called the "component type". The elements of an array are designated by indices, values belonging to the so called "index type".

The index type must determine a limited number of discrete values.

2.3.2.2. Record types

A record is a structure consisting of a fixed number of components, possibly of different types. The record type definition specifies for each component, called "field", its type and an identifier which denotes it. The scope of these so called field identifiers is is the record definition itself, and they are also accessible within a field designator to a record variable of this type.

In Twin Pascal a record type cannot have "variants".

2.3.2.3. Set types

A set type defines the range of values which is the powerset of its so called "base type". Base types must not be structured types.

The base type must determine a limited number (limit is 64) of discrete values. If the base type is a subrange, then the lower bound of the subrange must be the lowest possible value of the concerned scalar type, e.g. 0 for subranges of integers and "space" for subranges of characters.

2.3.2.4. File types

A file type specifies a structure consisting of a sequence of ɔmponents which are all of the same type. In Twin Pascal variables of the type "file" are related to disk files or peripheral devices. A file type cannot be an element of a structured type.

Twin Pascal distinguishes two types of files: "local" files and "external" files. The difference between the two file types is noticed upon block exit. Then local files, corresponding to file variables defined in the block, are deleted while external files are closed.

A file is an external file when the file identifier is also a program parameter. File identifiers which are no program parameters are local files.

2.4. Procedure declarations

Procedure declarations serve to define parts of programs and to associate identifiers with them so that they can be activated by procedure statements.

The procedure heading specifies the identifier naming the procedure and the formal parameter identifiers. Twin Pascal will only accept value parameters and variable parameters; procedure parameters and function parameters are rejected.

2.4.1. Standard procedures

The standard procedures of Twin Pascal are predeclared in a scope surrounding the main program.

2.4.1.1. File handling procedures

It is allowed to put (write) data to a file and to get (read) data from a file without prior execution of the procedures REWRITE, respectively RESET.

2.4.1.2. Dynamic allocation procedures

The standard procedure DISP is not implemented. The procedure statement DISP in a Twin Pascal program is ignored.

2.4.1.3. Data transfer procedures

The standard procedures PACK and UNPACK are not implemented. The procedure statements PACK and UNPACK in a Twin Pascal program are ignored.

2.5. Function declarations

.unction declarations serve to define parts of the program which computes a scalar value or a pointer value. Functions are activated by the evaluation of a function designator which is a constituent of an expression.

The function heading specifies the identifier naming the function, the formal parameters of the function and the type of the function result.

Twin Pascal will only accept value parameters and variable parameters, procedure parameters and function parameters are rejected.

2.5.1. Standard functions

The standard functions of Twin Pascal are predeclared in a scope surrounding the main program.

5.1.1. Additional standard functions

Twin Pascal features the following additional predeclared standard functions:

- TAN (X)
  This function computes the tangent of X. The type of X must be either REAL or INTEGER, and the type of the result is REAL.

- RAND
  This function has no formal parameters. It generates a random number of the type REAL in the range 0 - 1.

2.6. Statements

Statements denote algorithmic actions and are said to be executable. A statement is either a simple statement or a structured statement.

2.6.1. Structured statements

Structured statements are constructs composed of other statements which have to be executed either in sequence, conditionally or repeatedly.

2.6.1.1. Conditional statements

A conditional statement selects for execution a single one of its component statements.

2.6.1.1.1. Case statement

^ case statement consists of an expression (the selector) and a list ./ case list elements. A case list element is a statement preceded by a (string of) case labels. A case label is a constant. The constant type must be compatible with the selector type. A case list element specifies execution of the statement in the case the selector is equal to one of the case labels.

The last case list element may be of the form "ELSE <statement>". This case list element describes execution of the statement when the selector is not equal to any of the preceding case labels in the case statement.

```
<case_statement>     ::= CASE expression OF
                         <case_list_element>
                         [ ; <case_list_element> ]
                         END
                       ! CASE expression OF
                         <case_list_element>
                         [ ; <case_list_element> ]
                         ELSE statement
                         END

<case_list_element>  ::= <case_label_list> : statement
                       ! empty

<case_label_list>    ::= constant [ , constant ]
```

2.7. Programs

A Twin Pascal program has the form of a procedure declaration except for its heading.

The program parameters denote the external files. The two standard files INPUT and OUTPUT need not to be listed as parameters in the program heading, if they are used.

The default standard input file is CONI (console keyboard) and the d `ault standard output file is CONO (console output).

2.8. External references

The compilation of a complete program may require a considerable amount of time and space. The needed time and space can be reduced substantially (especially with corrections) when the original program is partitioned into several modules. The source modules are compiled separately and the resulting object modules will be united into a single executable program by the Link Editor.

A Pascal program consists of a main program which contains a number of procedures and functions. This structure suggests to consider a procedure (function) as a module. This partition is followed in Twin Pascal. The Twin Pascal compiler will accept both program modules and procedure (function) modules. The Twin Link Editor is able to unite a single program object module and a number of procedure (function) object modules into a single executable program.

In general, links between the main program and the procedures (functions) are wanted. These links are provided automatically when the main program and the procedures are compiled together. With separate compilation these links are to be defined in another way. Twin Pascal realizes such links as follows:

1. The first module must contain the main program.

2. Following modules, called "procedure modules", will contain procedure declarations and function declarations.

3. If necessary, label declarations, constant definitions, type definitions and variable declarations are copied from the main program into procedure modules.

4. Links between the main program and procedure modules are realised as follows:

    1. All identifiers, except type identifiers, defined at level 0 in the main program are made available for separately compiled procedure modules;

    2. In procedure modules all label identifiers, constant identifiers and variable identifiers defined at level 0 are considered as external references;

    3. Procedures and functions declared at level 0 may be classified as "external" (i.e. they are treated as external references);

    4. Procedure identifiers and function identifiers declared at level 0 in procedure modules, not being classified as "external", are made available for other modules.

To describe these features, the following extensions in the syntax of Standard Pascal have been made:

```
    <module>                    ::= <main_program>
```

```
                                   ! <procedure_module>

    <main_program>              ::= program_heading block .

    <procedure_module>          ::= label_declaration_part
                                    constant_definition_part
                                    type_definition_part
                                    variable_declaration_part

procedure_and_function_declaration_part

    <procedure_declaration>     ::= procedure_heading block ;
                                  ! procedure_heading <directive> ;

    <function_declaration>      ::= function_heading block ;
                                  ! function_heading <directive> ;

    <directive>                 ::= FORWARD
                                  ! EXTERN
```

3. Use of Twin Pascal

The use of Twin Pascal can be partitioned into three parts:

- compilation;
- link editing;
- execution.

3.1. Compilation of Twin Pascal modules

The compilation of Twin Pascal modules is initiated by the system command:

```
PASCAL <sfile> <lfile> <ofile> <debug option>
```

in which:

```
1.     <sfile>         ::= source file (must be a disk file)
       <lfile>         ::= listing file
       <ofile>         ::= object module file
       <debug option>  ::= 'y'     or empty
```

Only the first parameter is required. If <lfile> is omitted, no listing is produced; omission of <ofile> supresses the generation of the object module.

After successful initialisation of the compiler the message:

```
** PASCAL **
```

is displayed on the console.

Detected errors are displayed on the console. When a source line contains several errors, only the first detected error is shown. The compilation is performed in two passes. Multiple defined labels are indicated in pass I; all other errors are reported in pass II. Errors detected in pass II are also shown in the listing.

An error message on the console consists of an error indicator followed by the current source line. The error indicators with their meaing are:

A:	addressing error
L:	label error
M:	multiple defined identifier
S:	syntax error
T:	type error
U:	undefined identifier

Note: Errors detected with the compilation of the last item in a source line may be reported in the next source line.

The listing contains four columns, named LOC, E, LINE and SOURCE.

The column LOC gives the state of the Static Storage Area location counter at the beginning of each statement.

The column E indicates the first (or only) detected error, the column LINE gives the sequence number of the source line and the column SOURCE contains the source line as read from the source file.

The listing is terminated by the total number of detected errors.

It is senseless to proceed with a module when the number of compiling errors is not 0.

3.2. Link editing

After successful compilation of the source text, the generated object module(s) is/are to be link edited in order to obtain an executable program.

Link editing is executed by the Twin Link Editor. The user has to specify the object modules that are to be linked together. The following rules apply:

1. The first (or only) module must be the main program, i.e. the object module compiled from the source file containing the program heading.
2. The main program may be followed by separately compiled procedure modules.

The main program contains the command to include the object module that realizes the abstract stack computer.

The output of the Link Editor is a load module that can be executed by Twin.

3.3. Execution of Twin Pascal programs

The execution of the instructions in the load module, as produced by the Link Editor, is initiated by the system command XEQ. The XEQ command may have parameters. These parameters will correspond to the program parameters of the Twin Pascal program.

When the program parameters are file identifiers, these file identifiers are replaced by the actual parameters from the XEQ command.

Example. Consider the program heading:

```
    PROGRAM PASCAL (INPUT,OUTPUT,FILE1,FILE2)
```

When a program with this heading is executed by the system command:

```
    XEQ <mod>,A,B,C
```

the following replacements are made:

```
  INPUT  becomes A
  OUTPUT         B
  FILE1          C
  FILE2  remains FILE2.
```

Execution of the program by the system command:

```
    XEQ <mod>,,,C
```

gives the next substitutions:

```
  INPUT  becomes CONI (default standard input file)
  OUTPUT         CONO (default standard output file)
  FILE1          C
  FILE2  remains FILE2.
```

When textfiles are assigned to CONI, the character ESC has a special function. Upon entering of this character the current input line is erased and a new input line is requested.

The abstract stack machine is able to detect certain errors. Upon the detection of an error the program is aborted. Then one of the following messages is displayed on the console:

Input/output errors:

```
    STATUS xx AT LOCATION zzzz
```

other errors:

```
    ERROR  yy AT LOCATION zzzz
```

in which:

xx:	SRB status (see Twin Reference Manual)
yy:	error code (see below)
zzzz:	instruction pointer of the stack machine

The possible error codes are:

01:	arithmetic overflow
02:	memory overflow
03:	addressing error
04:	illegal operation code
05:	invalid data

The value "zzzz" minus the offset of the object module (as given by the Link Editor) is the location of the Twin Pascal statement as indicated in the program listing.