SECTION 7

INSTANT PASCAL LANGUAGE DEFINITION

The language definition in this chapter presupposes a knowledge of Pascal concepts, particularly as they are presented in the "Report" section of "Pascal User Manual and Report", Second Edition, by Kathleen Jensen and Niklaus Wirth (see Appendix F). Following is a list of deviations from Jensen and Wirth Pascal; it may be presumed that any feature of Pascal which is not explicitly discussed here corresponds to Standard Pascal.

7.1 SUMMARY OF EXTENSIONS AND RESTRICTIONS

7.1.1 Extensions

Following is a list of the features of the AIM 65 Instant Pascal language which are not found in Standard Pascal.

a. Variables, both simple and structured, may be given absolute memory addresses. The syntax "=$hhhh" (where h is a hexadecimal digit) after any variable identifier in a variable declaration fixes the variable at address $hhhh and makes it global. (That is, its value survives the block in which it is declared.)

b. The underline character "_" may appear in user-defined identifiers wherever a digit may appear. This character is not on the AIM 65 keyboard, but it can be entered from a terminal.

c. The OTHERWISE: default clause may precede the last statement in a CASE statement in place of a case constant list.

d. Identifiers and labels may have any length; the entire string (except for insignificant zeros in labels) is significant.

e. The STRING data type may have values whose length varies dynamically up to the declared maximum.

f. The predeclared procedure BREAK causes interruption of program execution, if it is enabled.

g. The predeclared procedure SUBR(ENTRANCE) calls the machine-language subroutine whose entrance address is the declared address of the absolute CHAR variable ENTRANCE. SUBR(ENTRANCE,DATA) does the same, but before entering the subroutine it places the (lowest) address of DATA in $FE,$FF and places the first (lowest-addressed) byte of DATA in A.

h. The predeclared function FUNC(ENTRANCE) has a value of type CHAR. It calls a subroutine the same way that SUBR(ENTRANCE) does; upon return, it uses the contents of A for its value. FUNC(ENTRANCE,DATA) does the same as SUBR(ENTRANCE,DATA); upon returning, it uses the content of A for its CHAR value.

7.1.2 Restrictions

The following features of Standard Pascal are not found in AIM 65 Instant Pascal.

a. FILE types are not implemented, nor are the predeclared procedures GET, PUT, RESET, REWRITE, PAGE or the predeclared functions EOLN, EOF. The predeclared procedures READ, READLN, WRITE, WRITELN, are implemented to work with all character-serial devices available to the AIM 65

Monitor. These devices act like TEXT files and can read into and write from the following types: CHAR, INTEGER, REAL, BOOLEAN (Y or N on input), variable-length STRING.

b. Set expressions ([expression..expression] and operators +, -, *) are not implemented. One-byte sets (up to eight elements in the base type) and the relational operator IN are implemented; together with absolute-addressed variables these permit Pascal-level testing of bits in memory, for example, I/O status.

c. Records are implemented, but record variants are not.

d. Dynamic storage allocation (procedures NEW, DISPOSE) and the pointer type are not implemented.

e. The directive FORWARD is not implemented.

f. The constant definition identifier=identifier; is not implemented.

g. Ambiguity between field and variable names is not supported. Other Pascal visibility rules are fully supported.

h. The procedures PACK and UNPACK are not implemented. Packing of data is not done below the byte level. The word-symbol PACKED is accepted but nonfunctional.

i. Procedural and functional parameters to procedures and functions are not implemented.

j. GOTO may not jump outside its own block.

7.2 WORD-SYMBOLS AND PREDEFINED IDENTIFIERS

This section lists all word-symbols and predefined identifiers in Instant Pascal. If no comment accompanies a particular list item, it may be presumed to follow Pascal in its definition.

7.2.1 Word-Symbols

The list of word-symbols in Instant Pascal is the same as in Standard Pascal, with one addition: OTHERWISE. Two Pascal word- symbols, FILE and NIL, are not implemented and should be avoided as identifiers. (In fact, they cannot be used as identifiers.)

The Instant Pascal word-symbols are:

AND, ARRAY, BEGIN, CASE, CONST, DIV, DOWNTO, DO, ELSE, END, FILE, FOR, FUNCTION, GOTO, IF, IN, LABEL, MOD, NIL, NOT, OF, OR, OTHERWISE, PACKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH.

7.2.2 Predefined Type Identifiers

The predefined type identifiers are the same as in Pascal:

BOOLEAN, CHAR, INTEGER, REAL, STRING, TEXT.

TEXT is recognized but not implemented and should not be used as an identifier.

STRING is implemented differently from Pascal, as described below.

As a group, the predefined type identifiers deviate from Pascal in the fact that they cannot be redefined. In theory, Pascal

allows the identifier REAL to be used as a defined type name; Instant Pascal does not allow that.

The type STRING[length] where length is a positive integer constant (either a literal or identifier) whose value is less than 256, is defined as ARRAY[0..length] OF CHAR. The dynamic value of a variable S:STRING[N] is the sequence of characters S[0],S[1],...,S[ORD(S[0])], where 0<=ORD(S[0])<=N. That is, the first byte S[0] stores the dynamic length of the string.

Character and string literals are internally realized as values of type STRING. That is, the constant 'A' is a two-byte object whose bytes have the hexadecimal values $01 $41. For this reason, type compatibility constraints have been relaxed to the following extent: any value of type STRING, if its dynamic length is 1 (i.e., if the value of the first byte is $01), may stand in place of a value of type CHAR in assignment and comparison. For example,

```
PROGRAM STRINGDEMO;
CONST
 CHARCONST='A';
VAR
 S:STRING[10];
 C:CHAR;
BEGIN
 READ(S);
  IF S<>CHARCONST THEN
   C:=S
END
```

will execute without a diagnostic if, and only if, one character followed by RETURN, is entered in response to the READ.

Pascal allows a statement like X:='ABC' where X is of type PACKED ARRAY[3] OF CHAR; Instant Pascal does not. The compensation for this is that, in Pascal, X:='ABC' would not work if X were of type PACKED ARRAY[4] OF CHAR, but in Instant Pascal, any STRING type of declared length 3 or greater can be

assigned the value 'ABC'. In general a STRING[N] variable can be assigned a value of type STRING whose dynamic length is N or less, even zero, and even if the declared maximum length of the source is greater than N. Reference to the individual array components of a STRING type value is allowed. For example, given S:STRING[N], the CHAR-type variables S[0],...,S[N] are available to the program.

7.2.3 Predefined Boolean Constants

The BOOLEAN type is implemented as a one-byte enumerated type with the two values (FALSE,TRUE). The predefined identifiers FALSE and TRUE have the same meaning as in Pascal, as the constants of type BOOLEAN. For example, REPEAT ... UNTIL FALSE is a way of bracketing a nonterminating loop. The identifiers FALSE and TRUE may not be defined; they may be used only as the Boolean constants.

7.2.4 Predeclared Procedures

As in Pascal, the predeclared procedure identifiers are considered to be declared at a level above the program block; they can be redefined in the program.

The following procedure identifiers are predeclared in Instant Pascal; their definitions are given below.

```
BREAK
READ
READLN
SUBR
WRITE
WRITELN
```


The following Pascal procedure identifiers are not predeclared in Instant Pascal and are available as user-defined identifers.

```
DISPOSE
GET
NEW
PACK
PAGE
PUT
RESET
REWRITE
UNPACK
```

The definitions of the predeclared procedures follow.

a. BREAK. See 4.2.

b. READ, READLN. READ and READLN are the same except that after reading into the last parameter, READLN outputs a new line (CR or CR/LF) to the printer or attached terminal. READ(P1,...,Pn) sequentially executes READ(P1),...,READ(Pn); it is therefore necessary to describe only the single-parameter procedure READ(P).

The behavior of READ(P) is consistent with the Pascal READ(P) from the implied TEXT file INPUT, where the INPUT file is the AIM 65 Monitor's input device. Specifically it depends on the type of P, as follows:

BOOLEAN: Only two one-character values are accepted: "Y" and "N". "Y" sets P to TRUE; "N" sets P to FALSE. Other inputs cause "?" to be output and the READ waits for a correct input.

CHAR: One keystroke is accepted; its ASCII value is assigned to P.

INTEGER: A signed or unsigned integer in the range -32768..32767 is accepted and assigned to P. Inputs are read into an input buffer and only scanned after RETURN is keyed; therefore the DEL key may be used to correct keying errors. Scanning terminates at the first nondigit (after the possible initial sign). If the input buffer is empty, "?" is output and the READ again waits for input.

REAL: A signed or unsigned real number, with or without an exponent part, is accepted and assigned to P. The syntax accepted is exactly as in the source language, which corresponds to Pascal in this respect. Inputs with integer syntax are also acceptable. Remarks under INTEGER regarding the input buffer apply to REAL.

STRING: If the declared length of P is N, up to N characters followed by a RETURN are accepted and assigned to P. More than N characters of input will produce a run-time error. The DEL key is available for correcting keying errors. Empty input strings are acceptable.

d. SUBR. See 7.1.

e. WRITE, WRITELN. WRITE and WRITELN are the same except that after writing the value of the last parameter, WRITELN outputs a new line (CR or CR/LF) to the printer or attached terminal. WRITELN with no parameter is equivalent to SUBR(CRLF) where CRLF=$E9F0:CHAR;. WRITE(P1,...,Pn) sequentially executes WRITE(P1),...,WRITE(Pn); it is

therefore necessary to describe only the single parameter procedure WRITE(P).

The behavior of WRITE(P) is consistent with the Pascal WRITE(P) to the implied TEXT file OUTPUT, where the OUTPUT file is the same as the AIM 65 Monitor's output device. The ":" parameter separators of Pascal, and the formatting function they control, are not supported by Instant Pascal. The behavior of WRITE(P) depends on the type of P; what follows also describes output during tracing.

BOOLEAN: The letter "F" or "T" is output, if the value is true or false, respectively.

CHAR: The character is output untranslated. If the character is an ASCII nongraphic, behavior depends on the output device.

INTEGER: Values are output in the range -32768..32767. Nonnegative values are output with a leading space.

REAL: Values are output in nonexponential or exponential format, depending on their magnitude. Nonnegative values are output with a leading space.

STRING: The dynamic value of the string is output untranslated. The length byte is not output but defines the number of characters written.

ENUMERATED: The ordinal value is output as a two-digit hexadecimal number preceded by "$".

7.2.5 Predeclared Functions

As in Standard Pascal, the predeclared function identifiers are considered to be declared at a level above the program block; they can be redefined in the program.

Instant Pascal implements, in accordance with Pascal, all the predeclared functions of Pascal except the Boolean functions EOF, EOLN. In addition, Instant Pascal includes the predeclared function FUNC.

Following is the list of Instant Pascal predeclared functions:

ABS, ARCTAN, CHR, COS, EXP, FUNC, LN, ODD, ORD, PRED, ROUND, SIN, SQR, SQRT, SUCC, TRUNC.

All are defined as in Pascal except FUNC, which is defined in 7.1.

7.3 COMPARISON TO STANDARD PASCAL

This section is organized to facilitate a direct comparison between Instant Pascal and Standard Pascal, as reported in the "Report" section of "Pascal User Manual and Report", Second Edition. The number in parenthesis parallels the numbering of the Report. The content of each paragraph is a statement of any deviation which exists from Standard Pascal as defined in that paragraph of the Report.

7.3.1 Notation, Terminology, and Vocabulary (3)

a. Lower-case letters are not supported.
b. The character "[" is F1 on the AIM 65 keyboard; "]" is F2.
c. "^" is not supported.

d. "{" is replaced by "(*"; and "}" is replaced by "*)".
e. NIL is not supported.
f. FILE is not supported.
g. OTHERWISE is added to the list of special symbols.
h. Comments may only occur in places where text units are accepted.

7.3.2 Identifiers, Numbers, and Strings (4)

a. The underline "_" may appear in an identifier wherever a digit is allowed.

b. Strings of any length N in quotes are constants of the type ARRAY[0..N] OF CHAR, where the 0th element contains the value N.

7.3.3 Constant Definitions (5)

<constant identifier> is excluded from the <constant> definition.

7.3.4 Data Type Definitions (6)

<pointer type> is not supported.

a. Structured Types (6.2)

<file type> is not supported.

b. Record types (6.2.2)

(1) Variants are not supported.
(2) There must be a semicolon following the <field list> and preceding the END.

c. Set types (6.2.3)

(1) The operators +, -, and * on sets are not supported.
(2) The base type of a set may have a cardinality of at most 8.

d. File type (6.2.4)

(1) FILE type is not supported.
(2) TEXT type is not supported.

e. Pointer types (6.3)

Pointer type is not supported.

7.3.5 Declarations and Denotations of Variables (7)

Redefine <variable declaration> as follows:

```
<variable declaration> ::=
     <variable id> { , <variable id> } : <type>
<variable id> ::= <identifier>|<identifier>=$ <hex digit>
     { <hex digit> }
<hex digit> ::= <digit> | A | B | C | D | E | F
```

(1) Component variables (7.2)

<file buffer> is not supported.

(2) File buffers (7.2.3)

This form of variable is not supported.

(3) Referenced variables (7.3)

This form of variable is not supported.

7.3.6 Expressions (8)

a. NIL is not supported.

b. <set> is not supported.

(1) Multiplying operators (8.1.2)

Operations on sets are not supported.

(2) Adding operators (8.1.3)

Operations on sets are not supported.

7.3.7 Statements

a. Procedure statements (9.1.2)

Procedure and function parameters are not supported.

b. Goto statements (9.1.3)

It is not possible to jump out of a procedure or function.

c. Case statements (9.2.2.2)

(1) Redefine <case statement>:

```
<case statement> ::= CASE <expression> OF
<case list element> { ;<case list element>}
     <otherwise element> END
<otherwise element> ::= OTHERWISE:<statement> |
     <empty>
```

(2) The END may be preceded by a semicolon.

d. With statements (9.2.4)

(1) On page 49 of the Jensen and Wirth Pascal User Manual and Report, there is an example showing ambiguity between a variable and field identifier. Instant Pascal does not support such ambiguity.

7.3.8 Procedure Declarations (10)

The word symbols PROCEDURE and FUNCTION may not be used in a formal parameter section.

a. File handling procedures (10.1.1)

These procedures are not supported.

b. Dynamic allocation procedures (10.1.2)

These procedures are not supported.

c. Data transfer procedures (10.1.3)

These procedures are not supported.

7.3.9 Predicates (11.1.2)

EOF, EOLN are not supported.

7.3.10 Input and Output (12)

a. The predeclared files INPUT and OUTPUT are not named. (However, see 7.2.4 of this document.)

(1) The procedure read (12.1)

(a) The only file supported is INPUT, which is never named.

(2) The procedure readln (12.2)

(a) The only file supported is INPUT, which is never named.

(b) READLN is interpreted as in 7.2.4 of the present document.

(3) The procedure write (12.3)

(a) The only file supported is OUTPUT, which is never named.

(b) The write-parameter separator ":" is not supported, nor are the formatting functions which they control.

(4) The procedure writeln (12.4)

The only file supported is OUTPUT, which is never named.

(5) Additional procedures (12.5)

PAGE is not supported.

7.3.11 Programs (13)

The <program parameters> part is empty.