SECTION 3

INSTANT PASCAL OPERATION

All resources of the Instant Pascal system are made available to you by means of a command interpreter which executes one-letter commands. The Instant Pascal command prompt is "+<". This chapter describes the six major processes of the Instant Pascal system which you may invoke:

a. Source input from keyboard or other devices available to the AIM 65 Monitor.

b. Source output (listing) to the AIM 65 printer or other output devices available to the AIM 65 Monitor.

c. Initialization to an empty program condition.

d. Editing of the Pascal program.

e. Binding (identifier references) and syntax checking of the Pascal program.

f. Execution of the Pascal program.

3.1 READ SOURCE INPUT (R COMMAND)

Turn on AIM 65 power, then press 5 after the Monitor prompt. Press RETURN in response to both of the initialization questions. This will ready Instant Pascal to accept the first command:

```
ROCKWELL AIM 65
<5> MEMORY SIZE? 4096
 WIDTH? 20
+<
```

The "R" (Read) command causes Instant Pascal to accept Pascal source input until it receives an empty line; i.e., two RETURNs in a row. Press R, then RETURN in response to the IN= prompt; this denotes keyboard input. (Other input devices may be designated; see the AIM 65 User's Guide, 3.8.1.)

Now enter the following from the keyboard as shown:

```
PROGRAM SQUARES;
VAR I:INTEGER;
BEGIN
FOR I:=1 TO 10 DO
WRITELN(I,SQR(I))
END
```

Finish with an extra RETURN to terminate input and return to the command prompt.

For input and editing purposes, the language of Instant Pascal is grouped into indivisible units called "text units". Each line of input must contain a sequence of complete, syntactically correct text units. An input line may be up to 60 characters long, including the final RETURN.

If, after keying in a line, the line is immediately printed back out with two "#"s, one at the beginning of the line, then an input syntax error was detected. The second "#" is immediately to the left of the character which led the input translator into an impasse. No part of the line went into the program memory, so key it in again.

3.2 SOURCE OUTPUT COMMANDS

The Pascal program in memory is a sequence of text units. The Instant Pascal editor maintains a pointer, called the text unit pointer, which always points to one text unit in the program.

When the program is initialized to an empty state, it contains one text unit, which is the "." at the end of every Pascal program. This text unit is always present and is never entered.

All editing is done relative to the point in the program denoted by the text unit pointer. In particular, text units entered by the R and I commands are inserted into the program immediately before (above) the text unit designated by the text unit pointer, and the text unit pointer remains pointing at that text unit. Therefore, after the program of 3.1 is entered, the text unit pointer still points to the ".".

3.2.1 List Current Text Unit (SPACE Command)

The SPACE command +< > lists the designated text unit (i.e., the text unit designated by the pointer). Press the SPACE bar after entering the program in 3.1.

```
+< >
 .
+<
```

3.2.2 Move Text Pointer to the Top (T Command)

The pointer may be moved to the top of the program by the T command. Type T:

```
+<T>
PROGRAM SQUARES;
+<
```

3.2.3 List Text Units (L Command)

The L command lists a specified number of text units (not lines), beginning with the designated text unit. The listing is generated by a process called "the lister". The text unit pointer is not moved by the L command. After you key L, the system will prompt for a numeric input with "/". The standard AIM 65 conventions apply:

```
RETURN = 1
. = forever (actually 9999)
sequence of digits terminated by RETURN = that number
```

Then the OUT= prompt will ask for the output device. RETURN specifies the AIM 65 printer. (Other output devices may be specified; see the AIM 65 User's Guide, 3.8.2.)

Now execute the standard sequence for listing a program:

```
    +<T>
PROGRAM SQUARES;
 +<L>/.
 OUT=RETURN
 PROGRAM SQUARES;
 VAR
  I:INTEGER;
 BEGIN
1 FOR I:=1 TO 10 DO
2  WRITELN(I,SQR(I))
 END.
```

Details:

a. The VAR and I:INTEGER; are on separate lines even though they were keyed in on the same line. They are separate text units; the formatting algorithm works entirely from the internal form of the program, which disregards the number of text units appearing on an input line.

b. The numbers in the left-hand margin of the statement part of the program count the indentation level. (The blank preceding BEGIN and END is really a zero, which is blanked for the sake of appearance.)

c. The listing format is such that corresponding structured statement delimiters (BEGIN...END, REPEAT...UNTIL, IF...ELSE, CASE...END) are at the same indentation level, and can thus be matched up over a span of program lines by use of the marginal numbers.

d. If corresponding statement delimiters are not shown by the lister to be at the same indentation level, the program has a sequence (syntax) error which will be caught by the Binder (see 3.5).

If a tape recorder is connected to your AIM 65, you can avoid rekeying this program later by writing it to tape now. First adjust the interblock gap parameter in $A409 to a value appropriate to your recorder. ($20 is a reasonable place to start.)

```
+<ESC                    Return to Monitor
<M>=A409 08 xx xx xx     Display $A409...$A40C
</> A409 20              Change $A409 to $20
<5>                      Return to Instant Pascal (pointer at
                           top)
<5>+<L>/.                List all; position tape
OUT=T F=SQRS RETURN      Start tape motion in record mode
T=1 RETURN               Recording begins
+<                       Recording is complete; turn off
                           recorder
```

You may wish to examine the contents of this tape by reading it into the AIM 65 Text Editor and listing it. The text is essentially as listed by Instant Pascal except for indentation and marginal numbers. You therefore have the option of using the AIM 65 Text Editor for preparing and correcting Pascal source program tapes.

3.3 INITIALIZATION

Before destroying this program you can execute it by keying the Go command:

```
+<G>OK
 1 1
 2 4
 3 9
 4 16
 5 25
 6 36
 7 49
 8 64
 9 81
 10 100
```

3.3.1 Report Available Memory (M Command)

The number of free bytes in memory may be interrogated with the M (memory) command.

```
+<M>=2653
```

Details:

a. Excluded from the definition of free space is space allocated to all fixed tables (most of page 2), pages 0 and 1, and the translator input and output buffers (317 bytes at the very end of declared RAM). Also, 250 is subtracted so that even when free space reads zero there will be 250 bytes available for the system's stacks to permit execution to begin.

b. You will find that memory occupied by programs is independent of the use of spaces as separators in the input text.

c. Typically, the space occupied by a program is 150% of the space occupied by the ASCII text of the Pascal source. This number is obtained by subtracting the program's M value from the M value of an initialized system. The source code identifiers are included in the program to allow the lister to reconstruct the source program for output.

3.3.2 Initialize Memory (N Command)

The program may be cleared by the N (New) command. Because N is irreversible, you get a chance to change your mind; you must answer "Y" to R U SURE? for N to work.

```
+<N> R U SURE? Y
```

From this point on, a new program can be entered as described in 3.1.

3.3.3 Initialize Instant Pascal (Z Command)

Instant Pascal may be completely restarted by typing Z. In addition to initializing memory like the N command, the memory size and page width prompt messages are displayed the same as upon initial entry from the AIM 65 Monitor. The R U SURE? prompt is also displayed to minimize accidental clearing of the memory. (You even get one more chance to save your program if you push ESC in response to the MEMORY SIZE? prompt.)

```
+<Z> R U SURE?  Y
MEMORY SIZE?  4096
WIDTH?  20
```

3.3.4 Redefine Page Width (W Command)

The WIDTH? part of the initialization sequence can be executed on its own with the W command. This is useful if you wish to change output devices.

```
+<W>WIDTH? 80
```

3.4 PROGRAM EDITING

After initializing to an empty program recreate the program SQUARES in memory, either by keying it in (see 3.1) or by reading it from tape, as follows. First position the tape.

```
+<R>IN=T F=SQRS T=1
```

Now start tape reading. (See the AIM 65 User's Guide, 9.1.5 and 9.1.6, for a more detailed discussion of the use of tape.) The command prompt +< indicates a successful completion of the read operation. (If the tape is not read successfully, you may have to RESET the AIM 65 and then re-enter Instant Pascal. Listing the program will then show you how much has been entered.)

In this section, the following commands will be used to modify the program SQUARES.

U/n	Move the pointer up n text units (toward the top)
D/n	Move the pointer down n text units (toward ".")
B	Move the pointer to the bottom, or ".", text unit
F	Find the line containing the indicated string
K/n	Delete n text units beginning with the designated one
I	Insert one input line of source
V	View a neighborhood of the designated text unit

First, list the program (T L . RETURN). Now verify that the pointer is at the top by executing the +< > (SPACE) command.

3.4.1 Move the Text Pointer to the Bottom (B Command)

Move to the bottom by executing the +<B> command. The "." that indicates the end of source code is displayed.

3.4.2 Move the Text Pointer Up (U Command)

Now move up to the I:INTEGER; by executing +<U>/5 RETURN. Note that after U the new designated text unit is listed.

```
+<U>5
I:INTEGER;
```

3.4.3 View Five Text Units (V Command)

Sometimes the designated text unit will be a semicolon or some other indistinguishable feature such as BEGIN or END. You can view a neighborhood of the designated text unit with the V command. Try it.

```
+<V>
PROGRAM SQUARES;
VAR
 I:INTEGER;
BEGIN
1 FOR I:=1 TO 10 DO
2
```

Details:

a. V is equivalent to U/2 L/5 except that the text unit pointer is not moved.
b. Normally V displays five text units with the designated text unit at the center. Exceptions occur near both ends of the program.

3.4.4 Move the Text Pointer Down (D Command)

After satisfying yourself that you are at the I:INTEGER, move down three text units to the output statement by executing D/3:

```
+<D>/3 RETURN
WRITELN(I,SQR(I)
+<
```

3.4.5 Delete Text Units (K Command)

Now delete the text unit WRITELN(I,SQR(I)) by executing +<K>/RETURN. (Remember that RETURN means 1 when a numeric input, prompted by "/", is being requested.)

```
+<K>/RETURN
WRITELN(I,SQR(I))
***
END.
```

Instant Pascal lists exactly what it is deleting, followed by "***", followed by the new designated text unit.

Detail:

> K/n is implemented as L/n followed by an internal block move which overlays the portion which was just listed. If you have second thoughts during the listing part of a K command, quickly hit RESET. Upon returning to Instant Pascal you will see that nothing has been deleted (assuming you were fast enough).

3.4.6 Insert a Text Line (I Command)

Insert one line of text by pressing I, typing in the next text and terminating the entry with RETURN. The text will be entered immediately before the current text unit (use the SPACE command if needed).

Detail:

> The command sequence K/RETURN I is the standard method of altering one text unit. The typical Pascal line A:=1; consists of two text units; if the A:=1 is the designated one, this command sequence will not touch the semicolon.

Before inserting something in place of the WRITELN, list the program.

```
+<T>
PROGRAM SQUARES;
+<L>/.
OUT=RETURN
PROGRAM SQUARES;
 VAR
  I:INTEGER;
 BEGIN
1 FOR I:=1 TO 10 DO
2
 END.
```

Note that this is a syntactically correct program with an empty statement in the scope of the FOR. You can execute it by pressing G. If you don't believe that anything happened, you can re-execute it with the assignment trace on (see 4.1). This traces assigned variables. Remember to toggle the assignment trace off after the program has executed.

```
+<A>ON
+<G>
FOR I:=1 TO 10 DO
> 1
> 2
> 3
> 4
> 5
> 6
> 7
> 8
> 9
> 10+<A>OFF
```

To insert a new statement in the scope of the FOR, you must find the END, because Pascal source is inserted before the designated text unit. This can be done by commanding a +<B> 1<U>/RETURN.

3.4.7 Find a Text String (F Command)

You can also find the END using the F (Find) command. Position to the top (+<T>) then key the +<F> command. At this point the

AIM 65 is waiting with a text input prompt in the display. (If the KB/TTY switch is set to TTY, the text prompt is "*" at the terminal.) Key in some string characteristic of the FOR statement, such as FOR or :=, then return. The editor will find the text unit. Now move down one text unit to the END.

```
+<T>
PROGRAM SQUARES;
+<F>
:= RETURN
FOR I:=1 TO 10 DO
+<D>/RETURN
END.
+<
```

Details:

a. Find is implemented by internally performing a L/. and diverting the lister's output to an internal buffer, in which a string match is performed at the end of each line (not text unit).

b. Some lines contain two text units, notably lines ending in ";" and ".". If a string match is found, the designated text unit is the last one listed, namely the ";" or ".". This can be disconcerting if the search for "X" IN "X:=1;" ends up on ";", so the following special case has been internally programmed: if a string match is found and the designated text unit is ";", +<U>/1 is expected. This special case does not test for ".", so a search for END in this example will end up pointing to ".".

c. Because the spaces that come out follow the rules of the lister and bear no necessary connection to the use of spaces (as separators) during source input, it is inadvisable to use space separators in a search argument until one understands how the lister puts them out, namely as seldom as possible.

3.4.8 Program Change Example

Now that the END has been found, change the program to generate a table of factorials by inserting a new statement before the END, using the I (Insert) command.

```
+<I>
WRITELN(I,FACT(I))
+<
```

This program is incomplete, because FACT is an undefined function; this can be verified by pressing G:

```
+<G>
*UNDEF* FACT
WRITELN(I,FACT(I))
*SEQUENCE*
WRITELN(I,FACT(I)
END.
```

Details:

a. The diagnostics come from the Binder, which is called by the C (Check) command (see 3.5) or whenever execution is called for (+<G>) but the program has been changed or has not been bound. The Binder looks up all identifier and label references and performs a total syntax check on the program. The lookup function of the Binder led to the *UNDEF* message; the syntax check function led to the *SEQUENCE* message.

b. Normally an undefined identifier would not lead to a *SEQUENCE* error, but an undefined identifier in an expression is assumed to be a variable, and the "(" following FACT is erroneous under that assumption.

c. The "OK" following +<G> or +<C> is produced by the Binder when it finds no errors.

Now key in a definition of FACT:

```
+<T>
PROGRAM SQUARES;
+<F>                                  Find
BEGIN                                the string argument
BEGIN                                Found it.
+<R>IN=RETURN                        Begin input.
FUNCTION FACT(N:INTEGER):REAL;
VAR I:INTEGER;
J:REAL;
BEGIN
J:=1;
FOR I:=2 TO N DO
J:=J*I;
FACT:=J
END;
RETURN
```

Go to the top and list the program:

```
+<T>
PROGRAM SQUARES;
+<L>/.
 OUT=
PROGRAM SQUARES;
 VAR;
  I:INTEGER;
 FUNCTION FACT(N:INTEGER):REAL;
 VAR
  I:INTEGER;
  J:REAL;
 BEGIN
1 J:=1;
1 FOR I:=2 TO N DO
2  J:=J*I;
1 FACT:=J
 END;
 BEGIN
1 FOR I:=1 TO 10 DO
2  WRITELN(I,FACT(I))
 END.
```

Now, pressing G will give you the first ten factorials.

```
+<G>OK
1  1
2  2
3  6
4  24
5  120
6  720
7  5040
8  40320
9  362880
10  3628800
```


3.5 PROGRAM BINDING (C COMMAND)

Binding is normally an automatic process, called by +<G> before execution if a program is unbound, that is, if it has not yet been bound since it was entered or if it has been changed since it was last bound.

Since the Binder produces intelligible diagnostic messages, it can be used as a syntax and undefined identifier checker; the C (Check) command exists for this purpose. It calls the Binder only, and produces the message "OK" or one or more Binder diagnostics (see Section 6.2).

Details:

a. The Binder does not check for type mismatches in expressions, assignments, or actual parameters. These are checked at execution time.

b. If the binding is without error, the text unit pointer is left at the top of the program. If the binding terminates with a *SEQUENCE* (syntax) error, the text unit pointer is at the first text unit printed after the word *SEQUENCE*. If the binding goes to completion (i.e., does not terminate on a *SEQUENCE* error) but causes diagnostics such as *UNDEF* (undefined identifier), *DUP* (duplicate identifier or label), *UNDECL* (undeclared label), or *TYPE* (type inconsistency in declaration), the text unit pointer will be left at the bottom of the program.

3.6 PROGRAM EXECUTION

3.6.1 Execute Program (G Command)

The G (Go) command calls the Binder if necessary and then, if the program has been bound without error, gives control to the Pascal Interpreter to execute the program.

Normally execution begins at the beginning of the program. There is one exception to this rule. Break-in-progress is a command-time condition which is true if the program was executing and execution was discontinued either by execution of the BREAK predefined procedure or by depression of the DEL key during execution. If Break-in-progress is true, G will cause execution to resume at the point of the break.

Details:

a. Break-in-progress is preserved even if ESC causes an exit to the AIM 65 Monitor. It is turned off by the New (+<N>) command.

b. Break-in-progress is turned off by invocation of the Binder. This has two consequences:

   (1) If the program is changed after a break, G will cause execution to start at the beginning, because the Binder will have been invoked.

   (2) The C command may be used explicitly to force G to start at the beginning after a break.

3.6.2 Execute and Trace Program (, Command)

The +<,> (Step) command is basically the same as +<G>, with the following additional properties.

a. Execution is traced. That is, text units are printed out before they are executed and the values of variables are printed out after they are changed by assignment.

b. Execution continues only until the start of the next statement, at which time the command prompt occurs with Break-in-progress true.

The +<,> is, in effect, a source-level single-step command. Any time that +<,> is in use, execution may be resumed with the G command, since break-in-progress is set.

Continuous execution may be interrupted and control returned to the command interpreter by holding down the DEL key. This key is sampled at the start of execution of every text unit. If it is down, KEY BREAK is printed and control is returned to the command interpreter with Break-in-progress set. Then the +<,> may be used to single-step for a while.

Details:

a. A text unit is printed just before the command prompt when execution is discontinued, both when the DEL key and the +<,> command are used. This is the text unit about to be executed.

b. The text unit pointer (which is not the same thing as the Interpreter execution pointer) is set to the text unit printed out before the command prompt. If it is not clear where the stop occurred, you can use the V command to enlarge the view.