THE ELLIOTT ALGOL SYSTEM.

(The content of section is taken from Don Hunter's manuals, updated to reflect the operation of the simulator and with added material from Volume 2 of the Elliott 900 Technical Manual).

What kind of ALGOL is Elliott ALGOL? Well it comes with the Elliott I/O system, 18-bit integers and reals with 8 significant decimal places. It was written by CAP and Elliotts in 1966/67 as an IFIP subset ALGOL for an 8K machine and was derived loosely from KDF9 Whetstone ALGOL. Norman Spink at Elliotts subsequently turned it into a Load-and-Go system for a 16K machine and Don Hunter removed most of the IFIP restrictions with patches in the spare space.

On an Elliott 903, translation speed was 100 characters per second and programs ran at 60 statements per second on the Whetstone benchmark, one third as fast as its KDF9 parent.

The main restrictions imposed by Elliott ALGOL compared to the formal definition of ALGOL are listed later in this manual.

The simulator contains five different versions of Elliott ALGOL:

1. In directory 903ALGOL: standard Elliott 903 ALGOL Issue 6 distributed to 903 users as a two pass "translator" and "interpreter" system for 8K machines, together with "load and go" and "large program" systems for machines with 16K or more of memory. In addition an ALGOL library was provided as a further tape. It assumes 903 telecode for input.

2. In the directory MASDALGOL: a version of Elliott 903 ALGOL Issue 6 distributed by the Maritime Aircraft Systems Division of Elliott's in 1974. The main difference from the other ALGOLs is the ability to handle both 903 and 900 telecode and the ability to use { and } as string quotes.

3. In directory HUNTER: an extended version of Elliott 903 ALGOL Issue 6 produced by Don Hunter. It removes several restrictions from the original (e.g., it supports recursion, removes the requirement to declare labels in switch lists) is more generous in the handling of call-

by-name actual parameters. It also removes a number of bugs (e.g., in the handling of overflow for integer comparisons). This ALGOL assumes 903 telecode. It is available as a 2 pass system for 8K machines and a laod-and-go system for 16K or larger machines. There is no large program system for HUNTER ALGOL.

4. In directory AJHALGOL: a new version of ALGOL by the author that combines the best features of 903 ALGOL, MASD ALGOL and HUNTER ALGOL. In addition it contains further bug fixes and improvements. AJH ALGOL renders 903 and HUNTER ALGOL obsolete.

5. In directory 920ALGOL: the 903 ALGOL Issue 6 two pass system adapted to handle 920 telecode in place of 903 telecode.

In the following sections the phrase "Elliott ALGOL" refers to aspects common to all versions of the ALGOL system. Aspects specific to particular variants are indicated as such.

Elliott distributions of ALGOL were labelled with "issue numbers". Generally Issue 6 was the last issue distributed, apart from an Issue 7 version of the ALGOL library. There was no Issue 6 version of the load-and-go ALGOL system making Issue 5 the final one. (This has a tiresome consequence, in as much as the conventions for writing code procedures for ALGOL changes with Issue 6, making the Issue 6 and 7 library tapes incompatible with the load and go system. An Issue 5 library has therefore been included in the relevant directories.)

Character set.

Depending on the version, Elliott ALGOL can use the 920, 903 or 900 Elliott telecodes. Each character set includes upper and lower case letters, but they are regarded as the same. On output all letters are in upper case.

In 903 telecode, the characters ½ $ % & ( ) * + - . / : ; < = > ⑩ [£] ↑ are legal in strings. !, ‘ and ’ are excluded and must be encoded as "inner strings" (see HUNTER\DEMO6). # can be used as an alternative to ½, ? for ⑩, ^ for ↑, ' for ‘, ` for ’ and _ for ←.

In 900 telecode, the characters £ $ % & ( ) * + - . / : ; < = > ? [\ ] ↑ are legal in strings, ! ‘ @ { | } are excluded and must be encoded as "inner strings" (see HUNTER\DEMO6). # can be used as an alternative to £, ^ for ↑, ' for ‘, and ← for _.

Note that in HUNTER and AJH ALGOL the character _ (or ←) is used to cancel an input line of text, by placing it as the last character in the line, thus it must be encoded as an inner string if it is to appear as output. This was arranged to allow corrections of programs typed on a Teletype.

903 ALGOL expects strings to be quoted using ‘...’. In 900 telecode this is equivalent to '...@, and not '...`.

MASD ALGOL and AJH ALGOL both accept both the 903 representation and the 900 telecode representation, and, in addition, the sequence {....} on input.

The representation of ALGOL symbols is as shown below and this allows spaces to occur in names (the stropping convention is double quotes around keywords in 900 and 903 telecode, or ~ in 920 telecode).

begin end	"BEGIN" "END" or "begin" "end"	
<=	"LE"	
>=	"GE"	
/=	"NE"	
and	"AND"	
or	"OR"	
not	"NOT"	
equiv	"EQUIV"	
impl	"IMPL"	
string quotes	' and @, ‘ and ’, or { and }	(900 telecode)
	‘ and ’, ‘ and `	(903 telecode)
	|< and |> (as overpunches)	(920 telecode)
subscript ten	? (900 telecode)	
	□ (920 and 903 telecodes)	
*+-/()[]<=>:;.,	stand for themselves	
exponentiation	↑ or ^	
div	"DIV"	

Restrictions

903 and 920 ALGOL implement an IFIP subset of ALGOL 60. Some of these restrictions are removed in HUNTER and AJH ALGOL.

1. Everything must be declared before it is used (in Hunter ALGOL this restriction is relaxed for labels, i.e., no switch statement is required to declare a label).

2. Recursion is not permitted in 903 or 920 ALGOL.

   This restriction is permitted in HUNTER and AJH ALGOL, however the local variables of a recursive procedure are in static storage, but see HUNTER\DEMO4 for a way of getting around this (use of a local procedure to provide dynamic storage). Note also the declaration before use restriction prevents mutually recursive procedures from being defined.

3. Expressions are not allowed as all-by-name actual parameters.

   This restriction is removed in HUNTER and AJH ALGOL.

4. Switch lists may only contain labels, not designational expressions.

5. Unsigned integers may not be used as labels.

6. The controlled variable in a for clause may not be subscripted.

7. own is not allowed.

8. All formal parameters of a procedure must be specified.

9. call of a type procedure can only occur in an expression.

10. Only the first six characters are significant in an identifier.

11. The identifiers checkr, checki, checkb and checks are reserved (see section on checking functions).

12. At most 14 parameters are allowed in a procedure call.

13. The formal parameters of an actual procedure parameter must all by "call-by-name".

    This is relaxed in Hunter ALGOL, but the translator needs help with calls of the formal procedures: the call must indicate which parameters are called by name and which by value. It assumes a call by value is meant when a closing bracket, round or square, precedes the comma or closing round bracket of the call. This means that an extra bracket must sometimes be used.

14. if A+B then is indistinguishable from if A+B > 0 then

15. Comments can only contain characters that are legal in strings, plus the characters permitted as string quotes.

Miscellaneous

The largest real is about 9^18.

If the Hunter ALGOL extended call by name facilities are not required, the generated code can be made shorter and will run slightly faster if the following patch is applied to the load and go system:

```
   E 8273 =8 86
```

Or to the two pass system translator:

```
   E 81 =8 86.
```

(See e.g., HUNTER\DEMO8 for an example).

In AJH ALGOL there is a pair of translator entry points to enable and disable the extended call-by-name facilities.

Both the HUNTER and AJH load and go systems can be told that more store is available. Place this patch

```
 E 27  +65535
```

near the front of the program (see e.g., HUNTER\DEMO2). This extends the data space, for example to declare an integer array with 60000 elements.

Standard library procedures.

The standard library for all version of Elliott ALGOL contains arctan, cos, instring, lowbound, outstring, range, sin, sqrt.

instring and outstring provide means to read and store text from a data tape. Such text has to be enclosed in string quotes.
instring(A,M) has the effect of searching for an opening string quote and then reading the text that follows up to the closing string quote that brackets the first. This text is stored in the locations A[M], A[M+1], ... three characters to each location. (A must be an integer array and M an integer variable. Before the procedure is used M must be assigned a value (normally) the lower bound of A. Following the call, M will be equal to the index of the next available element of M. Thus instring can be executed repeatedly in a way that the input strings do not overwrite each other.

Strings that have been read and stored by instring may be punched out by means of the procedure outstring(A,M). Initially M must have the value it had before the for the first instring procedure. Each time a string is punched out, M is advanced to be the start of the next string in A. Inner strings are interpreted as in print statements.

lowbound(A,M) gives the lower bound of the M-th bound pair of the array A. range(A,M) gives the range.

The following procedures are built-in to the ALGOL system: abs, aligned, digits, entier, exp, free point, ln, prefix, punch, reader, same line, scaled, sign, stop, wait.

aligned, digits, free point, prefix, punch, reader, same line and scaled are Elliott I/O procedures. If these procedures are called with a print statement their effect is local to that statement; if used outside of a print statement their effect applies to all subsequently executed print statements (unless overridden by other calls within those print statements or when updated by a subsequent call outside of a print statement).

prefix(S) causes every subsequent output item the be preceded with by string S; same line forces subsequent items to be printed on the same line (the default is a new line for each item).

Floating point numbers can be printed in free point, aligned or scaled form. Free point(N) prints numbers in N spaces with the decimal point where it belongs. The scaled(N) format always places the decimal point after the first digit and uses an exponent part to indicate the scale of the number. The aligned(M,N) format prints with M decimals before the point and N after. Leading zeros are replaced by spaces.

digits(N) prints integers in N spaces. Leading zeros are replaced by spaces.

reader(1) selects input from paper tape, reader(3) from teletype. punch(1) selects output to paper tape, punch(3) to teletype.

stop: the program halts.
wait: the programs halts and may be continued by entry at 9.

Some further names are known to the translator but are not in the standard library. These are advance, buffer, cencharacter, decode, draw line, move pen, set origin and way. These can be loaded using the extended library facilities described in a later section.

advance(I): read one character from device I into its buffer.
decode(I): decode the character in the buffer for device I into SIR internal code (a call of this procedure must be preceded by either advance, read or instring. After a "read" or instring, the buffer contains the character after the last digit or ` (grave) read.
buffer(I,S): "true" if the single character string ‘S’ is the character present in the buffer for device I, "false" otherwise.

cencharacter, drawline, movepen, set origin and way are described in the later section on plotting.

Checking mode.

Elliott ALGOL provides checking functions to enable intermediate results of a calculation to be printed out if the

program is translated in "checking mode". These functions are:

```
checkr    (for real argument)
checki    (for integer argument)
checkb    (for Boolean argument)
```

The statement

```
A := B/checkr(C);
```

causes the value of C to be output (preceded by a change to a new line and an asterisk), prior to the instruction A:=B/C being completed.

The argument of a checking function can be a variable or an arithmetic expression. Checking functions can be nested as in:

```
A:=B+checkr(M[checki(j+3)])
```

which would print first the value of (j+3) and then the value of M[j+3] on the next line.

Additionally there is a checking procedure

```
checks('...')
```

which can be used to trace the progress of a calculation.

If translation is not in checking mode, checking functions are all ignored.