THE ENGLISH ELECTRIC COMPANY LIMITED

DEPT.: Atomic Power Division

WHETSTONE.

REPORT No. W/AT 1153

A.C.E.D. Ref. No.

Deptl. Ref. APD/SDD/APS/MAB/MF

Date 8th July 1963

Copies | TO:—

(See first page)

Users' manual for the DEUCE ALGOL COMPILER

(Program No. 70623; A.N.D. No. P207)

Report by

M.A. BATTY

This report describes the use of the DEUCE Algol Compiler, which is the outcome of a collaborative project undertaken by J.M. Watt, D.S. Collens and G.M. Gillow of the Liverpool University Computer Laboratory, and the author.

Apart from the following further restrictions DEUCE Algol is compatible with Whetstone KDF9 Algol (see section 16):-

(1) No own arrays.
(2) No code procedures (but see section 9.4).
(3) No strings (but there is a "copy" facility -see section 9.5)
(4) Maximum of 14 parameters per procedure.
(5) Maximum of 14 dimensions per array.
(6) Maximum of 30 block levels.
(7) Integer range $-2^{22} \leq$ integer $< 2^{22}$
(8) Real range $-2^{127} \leq$ real number $< 2^{127}$

(Also, method of data input, output and layout of results, program testing facilities, and failure action are different, and less comprehensive).

The Compiler comprises a total of approx. 10,000 DEUCE words (Translator 6,000, Control 4,000).

With the initial version of Control, the running of translated programs is performed at $\frac{1}{3}$ - $\frac{1}{4}$ of the speed of Alphacode.

Translation time for the sample program given in Appendix II is 6 mins., total running time for the four cases shown being 12 mins.

Minimum DEUCE configuration required is a MK I machine with A.I.M., 64-column card input/output, and paper tape input/output.

(No correlation is intended with the original experimental DEUCE Algol (see report No. W/AT 844 by B. Randell and I.J. Russell, 12/2/62.))

Distribution.

Mr. J. Cameron
Mr. W. Wadkin
Mr. C. Bottrell
Dr. G.M. Insch
Mr. P.H.W. Wolff
Library
Mr. E. Long
Mr. J.D. McKean
Mr. A.J. Joyce
Dr. D.M. Parkyn
Dr. J.G. Balfour

Mr. B. Randell
Mr. L.J. Russell
Mr. M.A. Batty
Dr. A. Young
Mr. J.M. Watt
Mr. D.S. Collens
Mr. G.M. Gillow
Mr. A. Kirk

Computer Laboratory, The University, LIVERPOOL, 3.

Mr. W.E. Scott
Mr. C. Robinson
Mr. F.G. Duncan
Mr. J. Boothroyd

English Electric-Leo Computers Ltd., KIDSGROVE.

Contents.

(Sections are numbered from 6, to facilitate implicit reference to the Algol 60 report).

6. Introduction

The decision to write this Compiler was made at a meeting at Liverpool University between Dr. A. Young and J.M. Watt of the Liverpool University Computer Laboratory and Dr. D.M. Parkyn and B. Randell of the Mathematics Office of the Atomic Power Division, Whetstone, on November 1st. 1962.

At these two establishments there was felt a need for a means of easing the changeover to the forthcoming KDF9 and KDF9 Algol - by providing an interim mechanism to facilitate the gaining of experience in using Algol and the testing on DEUCE of Algol programs which would eventually be run on the KDF9.

Requirements of the Compiler were that it should be capable of use on a MK 1 DEUCE (The Liverpool University machine is a MK 1) and that it should be as compatible with KDF9 Algol as possible.

A prime consideration also was of time; the Compiler had to be ready for use by the middle of 1963 to be of practical use.

In view of the latter requirement it was decided to program on DEUCE the machine-independent flow diagrams and logic for the Whetstone KDF9 Algol Compiler (of which B. Randell and L.J. Russell are the authors*).

A collaboration was therefore set up in December, 1962 whereby the Translator part of the Compiler (i.e. that part which accepts Algol text as input and produces Object Program as output) would be programmed by J.M. Watt and D.S. Collens at Liverpool University, and the run-time Control (which obeys the Object Program) by the author, at Whetstone.

The completed Compiler was available from June, 1963, having involved a total of 20 man-months. As this time does not include the two man-years involved by B. Randell and L.J. Russell in the development of the logical system programmed, the total effort which the DEUCE Algol Compiler represents is approx. 44 man-months.

(* Of considerable help to all concerned were the manuscripts, which were made available to us, of a forthcoming book ("Algol 60 Implementation") by these same authors.

Basically, this book describes in detail the very method of implementation of Algol 60 which was adopted for the DEUCE Algol Compiler, and incorporates the actual flow diagrams used.)

7. How the Compiler operates.

7.1 The Compiler comprises two distinct parts - Translator, and run-time Control.

7.2 An Algol program tape produced on a Flexowriter (or a 5-hole tape in Telex (modified) or Ferranti code) is accepted as input by the Translator, and processed and punched out on cards as an Object Program, preceded by a set of cards known as the Failure Pack, which is used by Control in the event of a failure at run-time.

7.3 The card output from the Translator is accepted as input by Control, which then interpretively obeys the Object Program, thus performing the actions laid down by the original Algol program.

7.4 Input of Data at run-time is by 5-hole tape (Telex code at Liverpool University and Whetstone).

7.5 Full checking of the legality of the Algol and of calculation is provided throughout translation and at run-time, any errors found being notified to the user by means of messages punched out on paper tape. The next program will then be called for automatically.

7.6 At the end of both translation and run-time, the next program is called for automatically; there is also a load-and-go facility (see section 14.1.).

7.7 Program testing facilities available are Partial Answer and Trace (see section 12).

7.7.1 The Partial Answer facility is optional at run-time (P32 on I.D.) and causes the value assigned at each assignment statement to be printed out. (Boolean true appears as -1, false as 0). Block levels may be selective (see section 12.2.5).

7.7.2 The Trace facility is also optional (P31 on I.D.) and causes each Position Identifier as it is encountered to be printed out (first 8 characters only) during run-time. This facility is only available if there was a P31 on the I.D. during translation of the program.

7.8 Failures.

7.8.1 An error found during translation will be defined by line number (of Algol text), delimiter number on that line, the line on which the identifier was first used, and a number referring to an entry in a descriptive failure table (Appendix I).

7.8.2 Run-time failures are defined by a number referring to an entry in a descriptive failure table (Appendix I), the line number in the Algol text at which the failure occurred, and the preceding Position Identifier in the text (i.e. typographically). The Failure Pack is read in and used by Control to provide this failure information.

8. Restrictions due to Machine limits.

8.1 Size of object Program.

During translation the amount of space jointly available for Object Program and the list of identifiers having valid declarations is not more than approx. 2,048 DEUCE words at any one time. This is checked automatically.

8.1.1 Examples of space required by various items in the Object Program are:-

Item	Syllables*
Operator	1
Identifier	3
Numbers (other than integers 0 and 1)	5
Integers 0 or 1	1
Procedure declaration (n parameters)	5+n
Block	8
Procedure call (n parameters)	7+4n
go to (extra)	5
switch declaration ('n' desig. expressions)	5+10n
for ...step...until...do... (extra)	21

(* 4 syllables per DEUCE word).

8.2 Data Storage

At run-time, the maximum amount of storage available for Object Program, arrays, variables, intermediate calculations, links, recursive calls, etc., is approx. 4,100 DEUCE words, and a failure will occur if this limit is exceeded.

8.3 Block Levels. (Maximum 30)

8.3.1 The block level of any part of the text of a DEUCE Algol program can be determined by applying the following rules:-

8.3.1.1 The outermost block of a program is level 1.

8.3.1.2 Increase this by one on entering (typographically) any of the following structures, and decrease by one on exit:-

(a) Block (section 4.1)

(b) Procedure declaration (section 5.4) (If a procedure body is a block, this block is not regarded as an extra level).

(c) For Statement (section 4.6) (If a controlled statement is a block, this block is not regarded as an extra level).

8.3.2 It is the current block level which is indicated, when using the Partial Answer facility (see section 12) by the number of repetitions of the letter P following each value printed out.

8.4 Number representation (see section 2.5).

8.4.1 Integers have the permissible range:-

$-2^{22} \leq$ integer $< 2^{22}$

8.4.2 Real numbers within the range $-2^{127} \leq$ real number $< 2^{127}$ are represented accurately to about six decimal digits.

Real numbers falling in the range:-

$-2^{-127} <$ real number $\leq 2^{-127}$

are given the value zero.

These ranges are fully checked and will give the appropriate failure if exceeded.

9. Standard functions (see sections 3.2.4, 3.2.5.).

9.1 The standard functions set out in the Algol 60 Report are available as defined, declarations being implicit (i.e. the user does not have to incorporate corresponding declarations in his program).

9.1.1 sin (E) is computed using a formula given on P.140 of "Approximations for Digital Computers" (Hastings).

Maximum error is approx.
2×10^{-6} for $-\frac{\pi}{2} \leqslant E \leqslant \frac{\pi}{2}$
Failure No.166 will occur if
$$-2^{16} > \frac{2E}{\pi} \geqslant 2^{16}$$
(N.B. accuracy deteriorates outside the range
$$-2^4 \leqslant \frac{2E}{\pi} < 2^4)$$

9.1.2 cos (E) is computed using the formula given above in 9.1.1.

Failure No.167 will occur if
$$-2^{16} > \frac{(2E+1)}{\pi} \geqslant 2^{16}$$
(N.B. Accuracy deteriorates outside the range
$$-2^4 \leqslant \frac{(2E+1)}{\pi} < 2^4)$$

9.1.3 arctan (E) is computed using a formula given on p.136 of "Approximations for Digital Computers" (Hastings).

Maximum error is approx. 10^{-6} for any value of E.
Failure No. 165 will occur if
$$-2^{127} > \text{calculation} \geqslant 2^{127}$$

9.1.4 ln (E) is calculated using a routine based on DEUCE subroutine D10F/1.

Maximum error is approx. ± 2 in the sixth decimal figure.

9.1.5 exp (E) is calculated using a routine based on DEUCE subroutine D05FM/2.

Maximum error is approx. ± 2 in the sixth decimal figure.
Failure No. 162 will occur if
$$|E| \geqslant 2^{12}$$
(N.B. ln and exp are also used in the calculation of $a \uparrow r$ where r is of type real and $a > 0$ - see section 3.3.4.3.)

9.2 The following additional functions are available, also with implicit declarations:-

function	description in section:-
read;	10.1
print (X,i,f);	11.1
space;	11.2.2.
newline;	11.2.1.
copy;	9.3
deuce (entry);	9.4

9.3 The "Copy" facility.

9.3.1 This facility has been included in part amelioration for the lack of string handling facilities in DEUCE Algol.

It enables a string of any number of characters to be copied directly from data tape to output tape at run-time, so that titles, headings, etc., can be provided with output.

9.3.2 At the procedure statement "copy ;"the data tape will be passed through the reader until an opening parenthesis is found. Subsequent characters are then copied across to the output tape up to, but excluding, the corresponding closing parenthesis. Pairs of parentheses between the outermost pair of parentheses are treated as part of the "string" of characters copied.

9.3.3 It may be worth noting that a 'copy string' consisting only of parentheses, letters, =, CR, LF, space, blank and/or FS will be ignored if encountered by the 'read' function (see section 10).

9.4 The identifier "deuce"

A further function is available which facilitates the insertion and call of DEUCE subroutines.

The procedure statement:

"deuce (entry);"

where "entry" has the value:

track number (a/b)x32 + minor cycle (m),

will cause Control to fetch track a/b to DL.8 and enter it in mc.m.

The rules and necessary information for making use of this facility will be available shortly, on request.

10. Data.

Five-hole paper-tape has been chosen as the medium for input of data (and output of results) rather than 8-hole Flexowriter tape, which would have lead to the problem of reading 7-hole output, in the case where output is used subsequently as input.

10.1 In effect, the function designator

read

represents (and gives) "the next number on the data tape", e.g.:-

```
a:= read;
x:= sqrt (read);
y:= a + b/read;
z:= if read > 0 then a else read;
```

10.2 Punching of numbers for input.

A number of the form $A \times 10^B$ is punched as:-

$\pm A, \pm B$

where:-

10.2.1 + signs need not be punched.

10.2.2 A decimal point may occur anywhere within A, or be omitted if not required.

10.2.3 If B=0, the comma and B need not be punched.

10.2.4 The comma must immediately follow the last digit of A.

10.2.5 Carriage Return (CR) Line Feed (CF) or Space characters terminate a number (but not on Letter Shift).

10.2.6 It is permissible to put spaces between the comma and the first character of the exponent, or between the sign and first digit of A or B, but not elsewhere within the number.

10.2.7 The characters (,), =, and all characters on Letter Shift (except Figure Shift) are ignored. (i.e. TEMP (IN) = 123.45 etc. is possible).

10.2.8 Non-numerical characters, apart from . and , (and those covered in sections 10.2.5 and 10.2.7 above) are invalid, and will cause a failure if read.

10.2.9 A must have not more than 9 digits, excluding non-significant zeros before the point. (This is not fully checked).

10.3 Examples:-

0	-200.084	-.083,-02	(i.e. $-.083 \times 10^{-2}$)
177	+07.43,8	-,7	(i.e. -1×10^{7})
.5384	9.34,+10	,-4	(i.e. 1×10^{-4})
+0.7300	2,.-4	+,+5	(i.e. 1×10^{5})

+3.14159 could be punched in any of the following ways:-

+3.14159 +314159,-5

0.314159,1 +00.314159,+1 etc.

11. Output of results

11.1 The procedure statement:-

print (X,i,f);

causes X (integer or real) to be printed in fixed or floating form according to the values of the parameters i and f, as follows:-

11.1.1 i ≥ 0. X printed in fixed point decimal, with i places before the point and f places after the point. Non-significant zeros before the decimal point are replaced by spaces. The sign immediately precedes the first digit printed. If X has more figures before the point than allowed for by i, then the extra digits will be printed, but the layout of the page may be spoiled.

11.1.2 f ≤ 0. Decimal point and fractional part omitted.

11.1.3 i = 0. X printed in floating point decimal with one figure before the point and f after.

11.1.4 i < 0. As i=0, but -i spaces are left before the first digit printed.

11.1.5 The values of i and f are taken modulo 16.

11.1.6 Each number printed out is followed automatically by three spaces.

11.1.7 NUMBERS DO NOT AUTOMATICALLY START ON A NEW LINE.

11.2 Layout of output This is left under the user's control. In addition to the use of the print parameters there are two further facilities:-

11.2.1 newline. The procedure statement

newline;

causes the sequence Figure shift, Carriage Return, Line feed to be punched on the output tape.
(N.B. A newline is performed automatically at the beginning of each program).

11.2.2 space. The procedure statement

space;

causes one space to be left in the output.

12. Program testing facilities.

Two facilities have been provided to assist the user in program testing:-

12.1 Trace.

12.1.1 If a P31 is present on the I.D. during translation of a program, then for each Position Identifier in the program an extra seven syllables of Object Program are inserted containing the first eight characters of the Position Identifier.

12.1.2 If the Object Program is subsequently run with a P31 on the I.D., then on passing each label and at each call of a procedure the first eight characters of each Position Identifier will be printed out - at the beginning of a new line.

12.1.3 Using Trace, no distinction is made between upper and lower cases in the original Algol text, i.e. SUM and Sum would both be printed as SUM.

12.2 Partial answers. (No special action is called for at translation)

12.2.1 At run-time, if a P32 is present on the I.D., then the value assigned at each assignment statement will be printed out, at the beginning of a new line.

12.2.2 If the value assigned (X) is of type real, then the equivalent print statement (excluding the action of section 12.2.5) is:-

```
newline; print (X,-6,6); newline;
```

12.2.3 If X is of type integer, then the equivalent print statement (excluding the action of section 12.2.5) is:-

```
newline; print (X,7,0); newline;
```

12.2.4 Boolean assignments (sections3.4., 4.2). If X is Boolean, then true appears as integer -1, and false as integer 0.

12.2.5 Block levels. (see section 8.3). Following each partial answer printed out will be the letter P, repeated to a total of n times, where n is the block level at the assignment statement.

12.2.5.1 It is possible to select at which block levels partial answers are required. If instead of a P32, a Pn is placed on the I.D., then only partial answers at block level n will be printed out. For example, if partial answers from block levels 3 and 4 only are required, set P3 and P4 on the I.D.

Note that a P32 overrides P1-30.

12.2.5.2 Block level selection cannot be applied to the Trace facility but partial answers with block level selection, and Trace may be in operation together.

13. Preparing a tape.

13.1 Every tape should begin with -

13.1.1 At least 8" of blank tape (leader) - this facilitates loading into the paper-tape reader - followed by :-

13.1.2 CRLF, followed by:- } or vice-versa

13.1.3 upper or lower case shift, as required. }

13.2 Double spacing is recommended; this produces a much more easily read text.

13.3 Indentation according to block level is also recommended, for the same reason.

13.4 There is a TAB facility on some Flexowriters; this greatly facilitates indentation, but if a tape is reproduced on a Flexowriter with no TAB facility then the layout of the hard copy will be spoiled.

13.5 The final end of a program must be followed by the end of message character —> (no printing characters are allowed between end and —>).

13.6 —> should be followed by at least 4" of tape. This is necessary for the brake on the paper-tape reader.

13.7 Number types: effect of unnecessary conversions on running speed. Note that, if, in the following expression a and/or b are of type real, it will be evaluated faster (by just over 7% in the following example) at run-time if written:

(a + b)/2.0

than if written:

(a + b)/2

which would involve an unnecessary type conversion from integer to real at each evaluation.

Similarly, if a and b are both of type integer, it is preferably written:

(a + b)/2

(This philosophy also applies when using x, +, -, > , ≥ , =, ≠, <, ≤ , but not ↑)

14. Operating Instructions.

(The user is assumed to be familiar with the use of DEUCE).

14.1 There are two distinct modes in which the translating and running of DEUCE Algol programs may be carried out.

14.1.1 One is the load-and-go mode, where a program is run immediately it has been translated. This method makes use of the fact that when an Object Program is punched out by the Translator, a copy of it is retained on the DEUCE drum. Thus it is expedient to simply read in Control and have it obey that copy of the Object Program.

14.1.2 However, where several Algol programs are to be translated and run, it is more economical to perform all the translations in sequence, then run each Object Program in sequence, so that Translator and Control need only be read in once each.

14.1.3 A combination of the two methods is possible, i.e. n programs may be translated consecutively, the nth program then being run.

14.1.4 Choice of method is signalled by use of the P32 digit of the I.D.

After translation of a program, if the P32 is off another Algol tape will be called for.

If the P32 is on, Control will be automatically read in, and the program just translated obeyed.

14.1.5 Using load-and-go, the Translator and Control packs are stacked together as one pack; there is no need to remove any cards from the front of Control.

14.2 USING THE TRANSLATOR (From the description provided by Liverpool University).

14.2.1 Hardware representations.

Three different representations are catered for. The first two of these are identical with those to be provided on KDF9.

The representations are 8-hole KDF9 Flexowriter code, 5 hole Ferranti code, and the 5-hole code (modified Telex) used at Liverpool University. The upper case letters of the 5-hole representations are treated throughout as though they were lower case.

Reference Language (I.D. Setting)	8-hole KDF9 ($0P_1$)	5-hole Ferranti ($3P_1$)	5-hole Liverpool University ($2P_1$)
0 to 9	0 to 9	0 to 9	0 to 9
A to Z	A to Z	-	-
a to z	a to z	A to Z	A to Z
+	+	+	+
-	-	-	-
x	x	*	x
/	/	/	/
÷	÷	*DIV	'DIV
↑	↑	* *	' '
<	<	*≥	'≥
≤	≤	*>	'>
=	=	=	=
≥	≥	≥	≥
≠	≠	≠	≠
≡	eqv	*EQV	'EQV
⊃	imp	*IMP	'IMP
∨	or	*OR	'OR
∧	and	*AND	'AND
¬	not	*NOT	'NOT
,	,	,	,
.	.	.	.
10	10	v	n
:	:	→	:
;	;	*,	',
:=	:=	*=	:=
(	(	(	(
)	)	)	)
[	[	*(	'(
]	]	*)	')
begin	begin	*BEGIN	'BEGIN
Boolean	Boolean boolean	*BOOLEAN	'BOOLEAN

Reference Language (I.D. Setting)	8-hole KDF9 (OP_1)	5-hole Ferranti ($3P_1$)	5-hole Liverpool University ($2P_1$)
go to	go to	* GO TO	'GO TO
etc.			
end message	→	?	?

' ' and string are not allowed.

14.2.2 DEUCE Operating instructions.

14.2.2.1 I.D. Settings.

Only the P_1 P_2 P_3 and P_{31} P_{32} positions are significant.

P_1 P_2 0 for 8-hole code.

3 for 5-hole Ferranti code.

2 for 5-hole Liverpool University Code.

P_3 1 for 5-hole Ferranti code failure messages.

0 for 5-hole Liverpool University Code Failure messages.

P_{31} On if the trace facility is needed.

P_{32} On if operating in load-and-go mode. In this case Control must follow the Translator program in the reader.

14.2.2.2 Paper Tape Reader Settings.

Wired plug - The standard plug is used for all codes.

8-hole tape Slide at 8-hole position.
Parity - Even
Ignore all ones - position immaterial.

5-hole tape Slide at 5-hole position.
Parity - Off
Read all ones - On (But for Ferranti code, the position is immaterial).

Paper Tape Punch Settings

Channel - 5

Wired Plug - The standard plug is used for both codes.

Parity - OFF

14.2.2.3 Initial Input the Translator Program (6,6 - 24x buzz - Translator not read in and stored correctly.)

Load the tape in the tape reader.

Check that tape punch and card punch are loaded, and ready.

The tape will be read 1" at a time. The program will stop on either 0,1-1X (or by calling the card reader to read Control

if in the load-and-go mode), after a successful compilation, in which case the compiled program (see section 14.2.3) will be in the card punch, or on 4, 17-17x after detecting an error in the ALGOL program, in which case a message will have been punched on the tape punch, and there may be some (useless) cards in the punch.

In either case, give a single-shot to compile a further ALGOL program.

14.2.3 The Compiled Object Program.

Card output from the Translator for each Algol program is in two distinct sets of cards, the Failure Pack, followed by the Object Program.

14.2.3.1 The Failure Pack. From the first non-zero card punched out, up to and including the card with "all ones" on the α - field Y row and 31P1 + 31P2 down the rest of the α - field, comprises the Failure Pack. This contains a line count of the Algol text together with a coded representation of each Position Identifier, and is used, if necessary, by Control in the event of a run-time failure.

Failure pack cards may be punched out intermittently.

The first card of a Failure Pack will always have 123P17 in the second half of the 1st word in the α - field.

14.2.3.2 The Object Program. The rest of the card output after each Failure Pack comprises the Object Program.

Note that the order in which Failure Pack and Object Program are punched out is THE REVERSE OF THE ORDER IN WHICH THEY ARE REQUIRED BY CONTROL (except when operating the load-and-go mode, when only the Failure Pack is required.)

The first card of any Object Program will have the pattern P1,3,5,7,9,11,13,15 in the first half of the first word in the α -field.

The last card of any Object Program will have "all ones" in the β -field as the last non-zero word.

14.2.4 Failure Messages.

These have the form:-

LINE	p	
DELIMITER	q	where p,q,r,s are 3 digit
IDEN FIRST		integers.
USED LINE	r	
FAILURE	s	

The line and delimiter numbers serve to locate the position of the failure. The first line of the program is line 1 and the count is increased by one for each line which contains printed characters. The first delimiter of each line is delimiter 1.

The error is detected at the point indicated. It may be caused by blunders earlier in the ALGOL program.

The integer r normally gives the line number where the identifier currently being processed was first used in the current block. It is sometimes useful in locating errors.

The failure number s is used to locate the description of the possible cause of failure in the Failure Table (appendix I).

14.3 USING RUN-TIME CONTROL.

14.3.1 Initial state of the machine.

14.3.1.1 'A' DL's (IIA DEUCE). These are not required and should be switched off.

14.3.1.2 State of the I.D. Should be clear, unless the program testing facilities (section 12) are to be used.

14.3.1.3 Paper tape reader (standard plug). If data input is involved load the tape with the slide set for 5-hole, and set the panel keys as follows:-

parity up (off)
read all ones down (on)
clear parity level
load level (ready)

14.3.1.4 Paper tape punch(standard plug). Set for 5-hole output, with parity off, and made "ready".

14.3.2 Card input. Following the Control pack in the card reader must be:-

if load-and-go then Failure Pack only for program being run else Object Program followed by its Failure Pack;

14.3.2.1 Further Object Programs to be run may then follow, each immediately followed by its respective Failure Pack.

NOTE THAT THIS IS THE REVERSE OF THE ORDER IN WHICH THEY ARE PUNCHED OUT BY THE TRANSLATOR.

14.3.3 Control sum check. When the last card of Control has been read, Control is summed down from the drum. If this sum is incorrect a failure will occur (o, 5-5 x, buzz).

14.3.4 Entry to Object Program.

An automatic "newline" is performed before Control begins to obey the Object Program.

14.3.5 Finish of program. A DEUCE Algol program should finish in one of the following three ways:-

14.3.5.1 The final end has been reached, in which case Control finishes by printing out "FINISH"; it then passes the Failure Pack through the reader, and calls for the next Object Program (which in turn should have its Failure Pack behind it), stopping 7,0-21X if none.

14.3.5.2 The program calls for non-existent data. The Failure Pack will be left in the reader; this should be removed if further Object Programs are to be run. These should be stacked behind the Restart Card (see section 14.3.7), which is then initial-inputted.

14.3.5.3 A failure occurs, in which case, after the procedure described in Section 14.3.6 has taken place, the next Object Program will be called for.

14.3.6 Run-time failures.

When a failure has occurred Control provides three items of information to the user to assist him in tracing the cause and reason for failure; they are:-

14.3.6.1 A number referring to the Failure Table (Appendix I).

14.3.6.2 The line number of the Algol text at which the failure occurred (First line = line No. 1).

14.3.6.3 The preceding Position Identifier (typographically).

(Control reads in the Failure Pack and extracts the necessary information for items 14.3.6.2 and 14.3.6.3).

```
begin comment example of run-time failure;
real x ;
clanger: x : = true
end →
```

The assignment statement, as might be expected, produces a failure, as follows:-

"FAILURE NO.130 LINE NO.3 CLANGER"

14.3.7 Restart . At any time when Control is not calling for an Object Program, and this is required, the initial-inputting of the Restart Card (No. 2999 i.e. last card of Control) will cause Control to call for an Object Program, after printing out "RESTART".

15. Glossary (of non-Algol non-DEUCE terms used in this report).

Position Identifier. Any label, or Procedure Identifier in a procedure heading.

Object Program. Binary coded form of the Algol text, produced by the Translator and obeyed interpretively by Control.

Translation. Transformation of the Algol text into Object Program form by the Translator.

Failure pack. Set of cards produced by the Translator with, and preceding each Object Program; contains information enabling Control to provide line number and Position Identifier at a run-time failure.

Run, run-time. The obeying of the Object Program by Control, thus carrying out the actions set out in the Algol text.

Copy string. The set of characters between the outermost pair of parentheses on a data tape, when using "copy" (section 9.3).

Compiler. Translator + Control.

16. Definition of KDF9 Algol (As described in section 1 of KDF9 Algol Note 1, 22/4/63).

The language is substantially that described in the Revised ALGOL 60 report.

Historically the original ALGOL 60 report contained a number of ambiguities and ill-defined concepts. The English Electric ALGOL team were forced to agree a set of revisions covering these points well before the Revised ALGOL 60 report was published. In most cases the team anticipated the recommendations of that report and in others have been forced to go further than it in defining some of the undefined concepts, having carefully studied the discussions in the ALGOL Bulletin.

Further, the ALGOL compilers will contain some numerical limits on, for example, the number of parameters allowed in a procedure declaration and of course the corresponding procedure statement. It is hoped that these arbitrary limits are sufficiently large to be unnoticed by ALGOL programmers. A list of these numerical limitations will be published when the compilers are completed. This deliberate delay is due to the fact that where possible the limit is being made flexible and can therefore be revised when the authors have written and tested the compilers.

16.1. Restrictions on ALGOL 60 as Defined by the Revised Report.

All references are to the ALGOL 60 report.

3.5.1 < label>::= <identifier >

Section 3.5.5. is deleted.

2.4.3 Identifiers - only the first 8 characters are significant (for the Kidsgrove compiler the first 155 characters are significant).

4.3.5 Replace by:

"A go to statement leads to a failure indication if the designational expression is a switch designator whose value is undefined".

4.5.3.2 Replace the sentence "If none .... dummy statement" by:

"In the case of the second form of conditional statement if none of the Boolean expressions of the if clauses is true, the whole conditional statement will have no effect other than that which might be induced by the evaluation of the Boolean expressions".

5.4.5 Last sentence: replace "may be omitted" by "must also be supplied".

4.7.5.5 Add to text:

"Actual parameters corresponding to a real, integer, or boolean formal parameter called by name and to which assignments are made must have the same type as that specified for the formal parameter. Where assignments are not made, the actual parameters are regarded as expressions and provision is made for appropriate transfer functions to be invoked as necessary. This latter does not apply where the formal parameter called by name is specified as an array; in such a case the actual parameters must always be of the specified type.

Where a formal parameter is specified as a procedure, all the corresponding actual parameters must be procedures with identical specification parts (naturally the identifiers do not have to be equivalent or the order of specification identical). Further, for a formal parameter specified as a type procedure, the actual parameters must all be type procedures of the same type.

4.6.4.2 Replace the expansion by:

```
V1 := V := A;
V2 := B;
L1 : if sign (V2) x (V1-C) > 0 then go to Element exhausted;
Statement S;
V2 := B;
V1 := V := V + V2;
go to I1;
```

and add the following "V1 and V2 are auxiliary simple variables". This is to be read in conjunction with 4.6.5 (for reasons of optimisation).

4.6.5 Replace the second sentence by:

"If the exit is due to exhaustion of the for list, on the other hand, the controlled variable whose value caused the exit, is undefined".

5. Replace the fourth paragraph by

"Declarations for simple variables and arrays may be marked with the additional declarator own. The difference between own and non-own variables is as follows: non-own variables are attached to the block in which they are local, both with regard to the meaning of their identifier and with regard to the existence of their values. Each entry to a block, whether recursive or not, will bring a new set of the non-own variables of this block into existence, the values of these variables being initially undefined. On exit from a block all the non-own variables created at the corresponding entry are lost,

both with respect to their identifiers and values. Own variables, on the other hand, are local only with respect to the accessibility of their identifiers. Where the existence of their values is concerned they behave as though they were declared in the outermost block of the program. Thus every entry into a block, whether recursive or not, will make the same set of values of the own variables of this block accessible. On exit from a block the values of the own variables of the block are preserved".

"Dynamic own arrays" are not allowed, i.e. all arrays declared with the declarator own will have the lower bound and upper bound parts of the bound pair list elements given as integers. There is no restriction on the number of bound pair list elements.

5.4.4 Add to text

"If the very first symbol of a procedure declaration is a type declarator, there must occur within the procedure body one or more explicit assignment statements with the procedure identifier in the left part".

5.4.5 Last sentence, replace "may be omitted" by "must also be supplied".

16.2 Miscellaneous

Integers shall not have more than 39 significant binary digits (that is, they must lie in the range $-2^{39} \leq i < 2^{39}$).

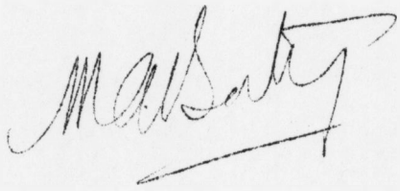
In the Kidsgrove translator, parameters except arrays called by value are evaluated from left to right; the arrays are then copied from left to right. In the Whetstone Translator all parameters are evaluated from left to right. (N.B. The order of taking parameters is not defined by the ALGOL Report).

17. Complaints. A specially prepared pro-forma is supplied by means of which users may notify the Compiler constructors of any inexplicable occurrence during use of the DEUCE Algol Compiler.

18. Acknowledgments.

The author wishes to express his thanks to J.M. Watt, D.S. Collens and G.M. Gillow of the Liverpool University Computer Laboratory for the lively manner and spirit in which their collaboration with him was conducted.

Also, on behalf both of himself and his co-constructors, he wishes to thank B. Randell and L.J. Russell, also of the Automatic Programming Section of the Atomic Power Division Mathematics Office, Whetstone, for making available the manuscripts and flow diagrams of their forthcoming book, without which such a comprehensive Compiler could not have been attempted, and for their encouragement and guidance.



M.A. Batty
Automatic Programming Section,
Mathematics Office,
Atomic Power Division,
Whetstone, Nr. Leicester.

APPENDIX I

FAILURES AT TRANSLATION.

MISCELLANEOUS.

0.
1. Program too big to compile.
2. Too many blocks.
3.
4.
5. Incorrect delimiter (including end message in middle of program).
6. True or false incorrectly delimited.
7. Identifier incorrectly terminated by decimal point or 10
8. Incorrect constant.
9. Constant too big.

TYPE CLASHES ETC.

10. Incorrect type of identifier.
11. Incorrect dimensions or No. of parameters.
12. Constant not allowed here.
13. Neither constant nor identifier allowed here.
14. Label not allowed.
15. Non-type procedure not allowed.
16. Procedure of wrong type.
17.
18. Must be simple variable.
19. Identifier needed here.

DECLARATIONS

20. Delimiter not allowed in this position.
21. Incorrect use of a sequence of delimiters.
22. Incorrect matching of delimiters.
23. Array segment incorrect.
24. Local variables used in array bounds.
25. Own array not allowed in this system.
26. Incorrect switch declaration.
27. Identifier used but not declared.
28. Identifier already declared.
29. Declaration not at beginning of block.

PROCEDURE DECLARATIONS

30. Delimiter not allowed in this position.
31. Incorrect use of sequence of delimiters.
32. Incorrect matching of delimiters.
33. Incorrect parameter delimiters.
34. This type not allowed.
35. Formal para part error.
36. Value or specification part error.
37. Too many or too few identifiers specified.
38. Type procedure contains no assignment to procedure identifier.

PROCEDURE CALLS

40. Delimiter not allowed in this position.
41. Incorrect use of a sequence of delimiters.
42. Incorrect matching of delimiters.
43. Incorrect parameter delimiter.
44. Incorrect use of procedure.
45.
46.
47.
48.
49.

EXPRESSIONS

50. Delimiter not allowed in this position.
51. Incorrect use of sequence of delimiters.
52. Incorrect matching of delimiters.
53. Delimiter must be arithmetic, not relational or logical.
54. Two relational operators cannot be consecutive.
55. Incorrect suffixed variable.
56.
57.
58.
59.

STATEMENTS AND GENERAL

60. Delimiter not allowed in this position.
61. Incorrect use of pair of delimiters.
62. Incorrect matching of delimiters.
63.
64.
65.

STATEMENTS AND GENERAL

66. Incorrect termination of statement.
67. End cannot occur in declarations.
68. No end message at end of program.
69. Incorrect beginning of program.

FAILURES AT RUN-TIME

DEFINITIONS:-

arithmetic - real or integer.
algebraic - boolean or arithmetic.
stack - that part of the DEUCE drum not occupied by Object Program and Control -- used for storage of variables, arrays, links etc.

DATA INPUT (x)

100. $-2^{127} > x > 2^{127}$
101. x incorrectly dealt with by read routine (check number of decimal digits in $x \leqslant 9$).
102. x contains non-valid character.

SIZE OF NUMBERS (x)

104. Real x too big ($-2^{22} > x > 2^{22}$) to convert to integer.
105. $-2^{22} > x > 2^{22}$ (NEG, FLOAT, $\div$, +, -) (integer).
107. $-2^{127} > x > 2^{127}$ (x, /, +, -, $>$, $\geqslant$, $<$, $\leqslant$, NEG, ABS,) (real).
108. Divisor = 0 (/).
109. $-2^{22} > x > 2^{22}$ (x) (integer)
110. Divisor = 0 ($\div$).

PROCEDURE CALLS.

111. Incorrect number of parameters.
112. Actual parameter incompatible with formal parameter called by name.
113. Arithmetic formal parameter called by name and used on the LHS of an assignment statement does not have an actual parameter of the same type.
114. Label or Boolean actual parameter not of same type as formal parameter (called by name or value).
115. Arithmetic actual parameter incompatible with formal parameter called by name or value.
116 Array actual parameter type incompatible with formal array called by value.

117. Base address of array given as actual parameter to formal array called by value is too big ($\geq 2^{31}$).

118. Actual parameter to an algebraic formal parameter called by name or value is not algebraic.

INCORRECT TYPES.

119. x not boolean in not x.

120. Expression F of a while element in a for statement is not boolean.

121. Expression following if of an if clause is not boolean.

122.

123. Expression not arithmetic ($<, \leq, =, \geq, >, \neq, +, -, \times, /$).

124. x not arithmetic in := -x;

125. Expression not boolean (and, or, eqv, imp).

126. Expression not integer ($\div$).

ASSIGNMENTS.

128. LHS not an algebraic address.

129. RHS not an algebraic result.

130. Boolean address with arithmetic result, or arithmetic address with boolean result.

131. LHS's of a multi-assignment statement are not all of the same type.

SWITCHES.

133. Switch index out of range.

SUBSCRIPTED VARIABLES.

135. Subscripted identifier is not algebraic variable (or switch designator).

136. Incorrect number of dimensions in subscript list.

CALCULATION OF ARRAY ADDRESS (A).

137. A too big to store($\geq 2^{31}$).

138. A too small (i.e. below bottom of stack).

139. A not within array specified.

ARRAY DECLARATIONS.

141. Negative number of elements in an array dimension.

142. Array bound is not an arithmetic expression.

CALCULATION OF ARRAY ADDRESSING MECHANISM

143. Starting address of array too big to store ($\geqslant 2^{15}$).

144. Calculation of base address/mapping function too big to store ($\geqslant 2^{31}$).

145. Calculation of starting address, base address or mapping function too big ($\geqslant 2^{15}$).

USE OF THE OPERATOR $\uparrow$ ($x \uparrow y$)

146. x or y not arithmetic expression.

147. $x = y = 0$.

148. $x < 0$ (y real).

149. $x = 0$, $y < 0$ (y real).

150. $-2^{22} > (x \uparrow y) \geqslant 2^{22}$. (x and y integer; $y > 0$).

STANDARD FUNCTIONS (f(x)).

160. ln $x < 0$.

161. ln $x = 0$.

162. exp $|x| \geqslant 2^{12}$.

163. exp $e^{x} \geqslant 2^{127}$.

164. entier $-2^{22} > x \geqslant 2^{22}$.

165. arctan $-2^{127} >$ calculation $\geqslant 2^{127}$.

166. sin $-2^{16} > \frac{(2x)}{\pi} \geqslant 2^{16}$.

167. cos $-2^{16} > \frac{(2x+1)}{\pi} \geqslant 2^{16}$.

168. sqrt $x < 0$.

169. deuce track number invalid.

PROBABLE MACHINE/OBJECT PROGRAM ERRORS.

196. Syllable number of next operation is above top of Object Program.

197.

198. Control is trying to get at a stack address below the bottom of the stack.

STACK CAPACITY

199. No more storage space left in DEUCE.

FAILURES OCCURRING BEFORE THE OBJECT PROGRAM IS ENTERED.

(DEUCE STOPS ON:-)

7,1-1 X At FINISH, Failure Pack not next in reader. (S.S. to read Failure Pack).

5,3-3 X At Failure, Failure Pack not next in reader (S.S. to read Failure Pack).

FAILURES OCCURRING BEFORE THE OBJECT PROGRAM IS ENTERED.

(DEUCE STOPS ON:-)

* 0,5-5	X	Control not stored correctly in DEUCE (S.S. to re-attempt sum check).
0,7-7	X	Object Program not next in reader (S.S. to read Object Program).
* 0,9-9	X	Object Program not read in correctly (S.S. to re-read Object Program).
* 0,11-11	X	Object Program not stored correctly in DEUCE (S.S. to re-attempt sum check).

(* Sum check failures - alarm sounding).

APPENDIX II.

Sample program, data, and output.

```
begin comment Integral of the function axt3 + bxt2 + cx + d from l to u,
              using Simpsons Rule with n intervals of (u-l)/n:
              (from an algorithm by A.P.Relph);
      real a,b,c,d,l,u; integer n,p;
      real procedure f(x);value x ;real x ;
              f:= axxt3 + bxxxx + cxx + d ;
      cases:          copy; p:=read;
      data:           a:=read; b:=read; c:=read; d:=read;
                      l:=read; u:=read; n:=read;
      integrate:      begin real p,q,r; integer m;
                              r:=(u-l)/n ;
                              q:= f(u)-f(l);
                              n:= n-1;
                              for m:= 0 step 1 until n do
                              begin p:= l + mxr ;
                                    q:=q+2xf(p)+4xf(p+r/2.0)
                              end;
                      new line;new line;print(a,2,1);print(b,2,1);
                      print(c,2,1);print(d,2,1);print(l,2,1);
                      print(u,2,1);print((n + 1),2,0);
                      print((q x r/6.0),5,4)
                      end integrate;
      end of int:     p:=p-1;
                      if p > 0 then go to data
end→
```

DATA TAPE FOR APPENDIX 2.

(OUTPUT FROM APPENDIX 2

A B C D FROM TO (N) = INTEGRAL)

NUMBER OF CASES (F)= 4

FIRST CASE

A=4 B=9 C=16 D=23 L=-3 U=4 N=7

SECOND CASE

A=4 B=9 C=16 D=23 L=-3 U=4 N=3

THIRD CASE

A=4 B=9 C=16 D=23 L=-3 U=0 N=5

FOURTH CASE

A=4 B=9 C=16 D=23 L= 0 U=4 N=5

OUTPUT FROM APPENDIX 2

A	B	C	D	FROM	TO	(N)	= INTEGRAL
+4.0	+9.0	+16.0	+23.0	-3.0	+4.0	+7	+665.0000
+4.0	+9.0	+16.0	+23.0	-3.0	+4.0	+3	+664.9998
+4.0	+9.0	+16.0	+23.0	-3.0	+0.0	+5	-3.0000
+4.0	+9.0	+16.0	+23.0	+0.0	+4.0	+5	+668.0007

FINISH

Output of 2nd case run with P31 + P32 on I.D.

```
DATA
      +4.000000, +0   P
      +9.000000, +0   P
      +1.600000, +1   P
      +2.300000, +1   P
      -3.000000, +0   P
      +4.000000, +0   P
      +3              P
INTEGRAT
      +2.333333, +0   PP
F
      +4.870000, +2   PP
F
      -5.200000, +1   PP
      +3.390000, +2   PP
      +2              PP
      +0              PPP
      -3.000000, +0   PPP
F
      -5.200000, +1   PP
F
      -7.314911, -1   PP
      +4.320741, +2   PPP
      +1              PPP
      -6.666670, -1   PPP
F
      +1.514815, +1   PP
F
      +3.374998, +1   PP
      +5.973704, +2   PPP
      +2              PPP
      +1.666666, +0   PPP
F
      +9.318515, +1   PP
F
      +2.315648, +2   PP
      +1.710000, +3   PPP
      +3              PPP
+4.0      +9.0        +16.0   +23.0   -3.0      +4.0   +3   +664.9998
ENDOFLIST
      +2              P
```

APPENDIX III

Summary of DEUCE operating notes.

A. Translation.

A.1 I.D. setting.

Trace required at run-time .....P31 (all the time)

Load-and-go required.....P32 (before Object Program punch-out finished)

A.1.1 Algol tape code.

8-hole.....0 P_1

5-hole Ferranti.....3P_1

5-hole Liverpool Univ. (modified Telex) 2P_1

A.1.2 Failure message code (5-hole).

Ferranti 1P_3

Liverpool Univ. and Telex 0P_3

A.2 Paper tape reader (standard plug).

A.2.1 8-hole tape.

slide - 8-hole

parity - even

"read all ones" - (position immaterial)

A.2.2 5-hole tape.

slide - 5-hole

parity - off

"read all ones" - on (immaterial if Ferranti code).

A.3 Paper tape punch (standard plug).

channel - 5

parity - off

A.4 Stoppers.

0,1-1X - normal end of translation (single-shot to translate another tape - or read in Control if P32 on I.D.)

4,17-17X - end of printout of error message - single shot to translate another tape. (there may be some redundant cards in the punch hopper).

6,6-24X buzz - Translator not read in and stored correctly.

A.5 Card output.

	First card.	Last card.
Failure Pack	123F17(1101111) in 2nd half of 1st word.	"All ones" on top row in α-field; 31F1 + 31P28 down rest of α-field.
Object Program	F1,3,5,7,9,11,13,15 in 1st half of 1st word.	"All ones" in last non-zero word (β-field)

B. Run-time.

B.1. I.D. Setting.

Trace P31

All partial answers P32

Partial answers, block level

n only ..... Pn (no P32)

B.2 Paper-tape reader (standard plug).

slide	—	5-hole
parity	—	up (off)
read all ones	—	down (on)
clear parity	—	level
load	—	level

B.3 Paper-tape punch (standard plug).

channel	—	5
parity	—	off

B.4 Stoppers.

7,1-1X — Failure pack not next in reader (at FINISH)
(single shot to read Failure Pack).

5,3-3X — Failure pack not next in reader (at a failure)
(Single shot to read Failure Pack).

* 0,5-5X — Control not stored correctly in DEUCE
(Single shot to re-attempt sum-check)

0,7-7X — Object Program not next in reader.
(Single shot to read Object Program).

* 0,9-9X — Object Program not read in correctly
(Single shot to re-read Object Program).

* 0,11-11X — Object Program not stored correctly.
(Single shot to re-attempt sum check)

(* Sum check failures - alarm sounding).