ELLIOTT
903
COMPUTER
FACTS

CONTENTS

GENERAL INFORMATION

The Elliott 903 is a desk-sized computer designed to meet the exacting requirements of science, industry and education. Developed from the successful Elliott 920, a second generation military computer, the 903 inherits a rugged reliability and is equally suitable for on or off-line applications. Typical applications include:

Engineering Design
Medical Research
Structural Analysis and Design
Research and Teaching in all Educational Institutions
Process Control
Plant and System Simulation Studies
Computer Typesetting
Real Time Collection of Experimental Data etc.

The basic 903 system comprises a central processor, 8,192 word core store, paper tape input and output equipment and control panel, all housed in a single desk-type cabinet.

Input/output facilities consist of a tape reader, tape punch, teleprinter and an 18-bit parallel data highway, to which up to 2,048 peripheral devices may be attached.

With its large, high-speed immediate access store, effective instruction code and full hardware arithmetic and input/output facilities, the 903 has a versatility and performance previously associated with larger and more costly systems. It is backed by a wide range of peripherals and effective software which are described elsewhere in this booklet.

Weight: (complete with tape reader and punch) 470 lb.

Power Requirements: 250 volts a.c. 50 c/s, 120 W.

Dimensions: (desk version, excluding paper tape equipment):
Height: 3 ft. 1 in. Depth 2 ft. 2 in., Width 3 ft. 7 in

903 TELECODE

The standard 903 telecode is the International Standards Organisation 8-track code which is also used by the NCR-Elliott 4100 system. The paper tape station can however read and punch 5, 6, 7 and 8 track paper tape, a facility of great value in the reduction of 'captive data'.

The 903 is a binary machine with an 18-bit word length and an internal memory of 8,192 words. The store may be extended to a maximum of 65,536 words by the external addition of store blocks of 8,192 words.

Positive integers are represented directly. Negative integers are represented using the "two's complement" notation. The range of integers allowed by single length working is −131072 to +131071. Greater range and accuracy are obtained by multi-length working.

For floating point working, two words are used to hold one number An instruction is stored and interpreted in the following manner:

B Modifier bit 18	F Function bits 17–14	N Address bits 13–1

Bits 13 to 1 ('N')
Address bits which specify any of 8,192 store locations.

Bits 17 to 14 ('F')
Function bits which specify the operation to be carried out.

Bit 18 ('B')
The modifier bit. If it is a 0, the instruction is obeyed as stored; if it is a 1, the address bits are modified before the instruction is obeyed by the addition of the contents of the B Register. The function bits remain unaltered and the version of the instruction held in store remains unchanged.

The Control Unit provides either manual or automatic control over the computer. It can also be used to test operation of the computer independently of its peripheral system.

In the "AUTO" mode, only the power ON/OFF, RESET and TRACE controls are operative. When switched ON in this mode, the computer jumps to the program trigger location 8177 and commences operation.

The number generator keys are operative for setting up starting addresses when the Mode Switch is either in the OPERATE or TEST position. The Control Unit provides the following: Power ON/OFF, COMPUTER STOP, COMPUTER RESTART, COMPUTER RESET, JUMP.

With the Mode Switch in the TEST position additional number generator keys are operative, allowing complete instructions to be set up and the following special test controls are provided: ORDER STOP/CYCLE STOP, CYCLE REPEAT, ENTER, OBEY.

A loudspeaker with volume control is provided as an audible means of identifying individual program operation. Manual priority level control is effected by three buttons while associated lights indicate the level at which the computer is currently operating.

ISO Code Value	Value with Parity	Telecode Character	Binary Pattern	SIR Internal code		ISO Code Value	Value with Parity	Telecode Character	Binary Pattern	SIR Internal Code	
				Octal	Decimal					Octal	Decimal
0	0	blank	00000·000			64	192	`(grave)	11000·000	40	32
1	129		10000·001			65	65	A	01000·001	41	33
2	130		10000·010			66	66	B	01000·010	42	34
3	3		00000·011			67	195	C	11000·011	43	35
4	132		10000·100			68	68	D	01000·100	44	36
5	5		00000·101			69	197	E	11000·101	45	37
6	6		00000·110			70	198	F	11000·110	46	38
7	135	Bell[3,4]	10000·111			71	71	G	01000·111	47	39
8	136		10001·000			72	72	H	01001·000	50	40
9	9	Hor. Tab[1]	00001·001			73	201	I	11001·001	51	41
10	10	Line Feed[2]	00001·010	01	1	74	202	J	11001·010	52	42
11	139		10001·011			75	75	K	01001·011	53	43
12	12		00001·100			76	204	L	11001·100	54	44
13	141	Car. Ret.[3,4]	10001·101			77	77	M	01001·101	55	45
14	142		10001·110			78	78	N	01001·110	56	46
15	15		00001·111			79	207	O	11001·111	57	47
16	144		10010·000			80	80	P	01010·000	60	48
17	17		00010·001			81	209	Q	11010·001	61	49
18	18		00010·010			82	210	R	11010·010	62	50
19	147		10010·011			83	83	S	01010·011	63	51
20	20	Halt[1,4]	00010·100			84	212	T	11010·100	64	52
21	149		10010·101			85	85	U	01010·101	65	53
22	150		10010·110			86	86	V	01010·110	66	54
23	23		00100·111			87	215	W	11010·111	67	55
24	24		00011·000			88	216	X	11011·000	70	56
25	153		10011·001			89	89	Y	01011·001	71	57
26	154		10011·010			90	90	Z	01011·010	72	58
27	27		00011·011			91	219	[	11011·011	73	59
28	156		10011·100			92	92	£	01011·100	74	60
29	29		00011·101			93	221	] |	11011·101	75	61
30	30		00011·110			94	222	↑	11011·110	76	62
31	159		10011·111			95	95	←[3,4]	01011·111	77	63
32	160	Space	10100·000	00	0	96	96	@[1]	01100·000		

33	33	![3]	00100·001			97	225	a	11100·001	41	33
34	34	"[4]	00100·010	02	2	98	226	b	11100·010	42	34
35	163	±	10100·011	03	3	99	99	c	01100·011	43	35
36	36	$	00100·100	04	4	100	228	d	11100·100	44	36
37	165	%	10100·101	05	5	101	101	e	01100·101	45	37
38	166	&	10100·110	06	6	102	102	f	01100·110	46	38
39	39	' (acute)	00100·111	07	7	103	231	g	11100·111	47	39
40	40	(	00101·000	10	8	104	232	h	11101·000	50	40
41	169	)	10101·001	11	9	105	105	i	01101·001	51	41
42	170	*	10101·010	12	10	106	106	j	01101·010	52	42
43	43	+	00101·011	13	11	107	235	k	11101·011	53	43
44	172	, (comma)	10101·100	14	12	108	108	l	01101·100	54	44
45	45	—	00101·101	15	13	109	237	m [4,5]	11101·101	55	45
46	46	.	00101·110	16	14	110	238	n	11101·110	56	46
47	175	/	10101·111	17	15	111	111	o	01101·111	57	47
48	48	0	00110·000	20	16	112	240	p	11110·000	60	48
49	177	1	10110·001	21	17	113	113	q	01110·001	61	49
50	178	2	10110·010	22	18	114	114	r	01110·010	62	50
51	51	3	00110·011	23	19	115	243	s	11110·011	63	51
52	180	4	10110·100	24	20	116	116	t	01110·100	64	52
53	53	5	00110·101	25	21	117	245	u	11110·101	65	53
54	54	6	00110·110	26	22	118	246	v	11110·110	66	54
55	183	7	10110·111	27	23	119	119	w	01110·111	67	55
56	184	8	10111·000	30	24	120	120	x	01111·000	70	56
57	57	9	00111·001	31	25	121	249	y	11111·001	71	57
58	58	:	00111·010	32	26	122	250	z	11111·010	72	58
59	187	;	10111·011	33	27	123	123		01111·011		
60	60	<	00111·100	34	28	124	252		11111·100		
61	189	=	10111·101	35	29	125	125		01111·101		
62	190	>	10111·110	36	30	126	126		01111·110		
63	63	[1] 0	00111·111	37	31	127	255	erase	11111·111		

[1]Ignored by Teleprinter
[3]Ignored by Flexowriter
[4]Space on Lineprinter
[5]Upper case on Teleprinter

BASIC INSTRUCTION CODE TIMES

Function		*Effect*	*Time* μs[5]
/	[1]B-modification		6·5
0 N	[2]$B:=Q_{18-1}:=m$	Set B-register	30·0
1 N	$A:=A+m$	Add	23·5
2 N	$\{A:=m-A;\ Q_{18-1}:=m\}$	Negate and Add	26·5
3 N	$\{m_{18}:=0;\ m_{17-1}:=Q_{18-2}\}$	Store Q-register	25·0
4 N	$A:=m$	LOAD	23·5
5 N	$m:=A$	STORE	25·0
6 N	$A:=A$ and m	Collate	23·5
7 N	if A=0 then S:=M and Q:=undefined	Jump if zero	21·0[4]
8 N	S:=M	Jump	24·0
9 N	if A<0 then S:=M and Q:=undefined	Jump if negative	21·0[4]
10 N	m=m+1	Count in store	24·0
11 N	$m_{13-1}:=(S+1)_{13-1}$; $Q_{16-14}:=(S+1)_{16-14}$; $Q_{13-1}:=0$	Store S-register	31·6
12 N	$(A, Q):=A*m$	Multiply	76·5
13 N	$A:=\frac{(A, Q)}{m}\pm 2^{-17}$; $Q:=A\pm 2^{-17}$; $A_1:=1;\ Q_1:=0$	Divide	79·5
14 N	$(A, Q)+Q_1:=((A, Q)+Q_1)*2^Z$	Left shift	Up to 48 places
	$(A, Q)+Q_1:=((A, Q)+Q_1)*2^{Z-8192}$	Right shift	22+3Z
15 N	$2048\leqslant Z<6144$ (Z=7168)	Block Transfer[3]; Program terminate. i.e. return to lower level	20·5 (minimum)

Notes

1. In the first stage of B-modification the Q register is affected in an undefined manner, consequently the effect of a modified 3 instruction is undefined.
2. Interrupt cannot occur after this instruction.
3. Functions 14 and 15 are used with other addresses for input/output, and block transfer. Details are given against the relevant devices.
4. Add 6·5μs if jump occurs.
5. All times are quoted to a tolerance of ±10%.
   A is contents of Accumulator (18 bits).
   B is contents of B register for the current level (18 bits).
   S is content of the sequence control register (containing the address of the current instruction). The S register is incremented before the instruction is obeyed.
   Q is content of bits 18 to 2 of the Q register (extension Accumulator).
   $M = (S_{16-14}+N)_{16-1}$ or $(S_{16-14}+N+B)_{16-1}$
   m is the contents of location M
   Z = N or $(N+B)_{13-1}$

INSTRUCTIONS REFERRING TO THE BASIC PAPER TAPE STATION

Instruction	*Effect*	*Time*[1]	*Status bits*[4]
15 2048	Input one character from the paper tape reader to $A^{2,3}$	20·5μs	1
15 2052	Input one character from the on-line teleprinter to $A^{2,3}$	20·5μs	3
15 6144	Output one character (A_{1-8}) to the paper tape punch[2]	20·5μs	2
15 6148	Output one character (A_{1-8}) to the on-line teleprinter[2]	20·5μs	4

ADDITIONAL PAPER TAPE STATION FACILITIES

Instruction	*Effect*	*Time*	*Status bits*[4]
15 2050	Input one character from tape reader 2 to $A^{2,3}$	20·5μs[1]	1
15 2054	Input one character from teleprinter 2 to $A^{2,3}$	20·5μs[1]	3
15 6146	Output one character (A_{1-8}) to punch 2[2]	20·5μs[1]	2
15 6150	Output one character (A_{1-8}) to teleprinter 2[2]	20·5μs[1]	4
15 2049	Read status word[4]	20·5μs	
15 6145	Output control word[4]	20·5μs	
15 2051	Read status word of second paper tape station[4]	20·5μs	
15 6147	Output control word to second paper tape station[4]	20·5μs	

1. If the device is busy, the computer will be held up and the instruction will only be obeyed when the device becomes free.
2. The select input and select output switches may be used to override the choice of source and destination respectively.
3. The exact effect of these instructions, if C_{1-8} is the character input, is defined as:
   $A_{18-9} := A_{11-2}$ (shift 7 places left)
   $A_8 := A_1$ *or* C_8
   $A_{7-1} := C_{7-1}$
   If 7 track or 5 track tape is used C_8 or C_{8-6} respectively are undefined on input.

4. Status and control functions are only available if the appropriate on-line adaptor is fitted. If the indicated bit of the status word is one the appropriate device is available and the given instruction will be completed in the time shown.
If the device is not available the output or input instruction will be held up, unless the paper tape station is in on-line mode, set by output, control with A=1. Return to off-line mode by output control with A=0 or RESET.

INSTRUCTIONS REFERRING TO THE DIGITAL PLOTTER

Instruction	*Effect*	*Time*[1]
15 4864	Output one character to the plotter.	20·5μs
14 4864	Output Q characters ($1 \leqslant Q \leqslant 4095$) to the plotter. Characters are held one to a word. A contains the address of the first character.	sum of busy times for each character

Bit of word output	*Effect*	*Time*[2]
1	Move one step East.	3·3ms
3	Move one step West.	
3	Move one step North.	
4	Move one step South.	
5	Raise pen.	20ms
6	Lower pen.	

[1] If the device is busy, there will be a delay and the instruction will only be obeyed when the device becomes free.

[2] Time for which the plotter is busy and cannot accept another instruction.

9 KCS MAGNETIC TAPE FACILITIES

Programs are available for reading and writing blocks and records of data, and opening and closing magnetic tape files. The 903 9kcs Magnetic Tapes should normally be used via these routines. Details of the instruction code will be found in the Technical Manual. The status word read from the controller is usually printed when a non-recoverable error occurs. This may be interpreted by the table given below.

Bit	*Meaning when set to* 1 *in status word*
1	Handler busy (rewinding)
2	Handler in manual, or not available. All other bits undefined
3	Missed data transfer
4	Parity error
5	Short block
6	Long block
7	Write permit ring fitted
8	Tape on load point
9	End of tape detected
10	False end of block
11	An instruction has been rejected as 'Do Nothing'

Tape speed: 45 inches/sec.
Rewind speed: 180 inches/sec. (approx.)
Density: 200 characters/inch
(3 characters per word)
Inter block gap: 0·75 inches
Block length: Between 5 and 2047 words
Start/stop time: 18 millisec. (approx.)
Maximum tape length: 2400 feet

PRIORITY LEVEL PROGRAM ORGANISATION

Each priority level has its own sequence control register (SCR) and B register. These registers are locations in the store and can be referred to by program in the normal way.

A program may not address its own SCR location. It may address the SCR location of other priority levels.

Priority Level	*B. Reg. Location*	*S.C.R. Location*
1 highest level	1	0
2	3	2
3	5	4
4 base level	7	6

The accumulator and the auxiliary register are shared between all four levels so they must be safeguarded by program every time an interrupt occurs. All these conditions are fulfilled by the following control instructions. They are applicable to any program on levels 1, 2 or 3 which starts at location ENTRY.

```
L       0 Q1   (Reset Q18-2)
       14 1
        4 A1
       15 7168 (Terminate)
ENTRY  5 A1    (A1 and Q1 must be distinct)
        3 Q1   (for each interrupt program)
(INTERRUPT LEVEL PROGRAM)
          :
        8 L
          A1 +0
          Q1 +0
```

Note that this program does not preserve bit 1 of Q.

Programs entered by the JUMP button are obeyed from level 1. The Initial program should set up locations 2, 4 and 6 to appropriate addresses, then terminate to level 4, which is regarded as the base level.

INITIAL INSTRUCTIONS

Locations 8180 to 8191 inclusive are used for the initial instructions, used to input an initial program loader punched in 'binary' form. These instructions are entered on level 1 at location 8181. On 903 processors fitted with more than 8192 words of core store the locations 8180 to 8191 may be used as normal core store when the initial instructions are 'disabled'. The instructions are disabled whenever a 15 7168 is obeyed. They are enabled whenever the JUMP button is pressed. The contents of 8180 to 8191 will be preserved unless program is obeyed from those locations. The effect of reading these locations on a basic machine or while the instructions are enabled should be regarded as undefined.

The instructions are as follows:

```
8180   /15 8189 (–3)
ENTRY    0 8180
         4 8189
8183    15 2048
         9 8186
8185     8 8183
        15 2048
8187    /5 8180
        10 1
8189     4 1
         9 8182
8191     8 8177 (Trigger to program)
```

STORE ADDRESSING

The basic 8192 words of store can be addressed directly by the 13 bit instruction address (N). Larger stores are considered to be divided into modules of 8192 words (Module numbers from 0 to 7). Within each module, program and data are addressed relative to the first location (location 0) of that module. B-modification is used to access program and data outside the current module. The address actually accessed is $(S_{16-14}+N+B)_{16-1}$ therefore the address used for modification must be relative to the current module. Attempts to address store modules not fitted cause the processor to hold up.

SYMBOLIC INPUT ROUTINE

Symbolic Input Routine (SIR) enables programs to be written in modified machine code form. Essential details of SIR are given below. Full details of the routine can be obtained from the SIR Handbook.

S.I.R. Symbols used for the Major Facilities

(i) IDENTIFIER Group of up to six letters or numbers commencing with a letter.

(ii) GLOBAL IDENTIFIER LIST [ ] e.g. [CAT !!DOG RAT]
(CAT and RAT are Global, DOG is Sub-Global.)

(iii) CONSTANTS

(a) Integer or fraction	± e.g. −71032
(b) Octal groups	& e.g. & 770123
(c) Alphanumeric groups	£ e.g. £A23

(iv) ADDRESSES

(a) Absolute	an unsigned integer
(b) Block relative address	N;
(c) S.I.R. relative	;±N
(d) Identified address	Identifier ± Integer (if required)
(e) Literal address	Any constant as in (ii) above or = followed by a quasi instruction (absolute address only)

(v)	SKIP	>+N
(vi)	COMMENT	(THIS IS A COMMENT)
(vii)	PATCH	†A
(viii)	RESTORE	$
(ix)	OBEYED INSTRUCTION	' ' e.g. ' 8 A '
(x)	END OF TAPE	Ⓗ (HALT CODE)
(xi)	END OF PROGRAM	%

(Locate literals, cancel Sub-Global identifiers, list undeclared Global Identifiers and display a FIRST LAST NEXT message.)

Note

N is an unsigned integer.

A is any currently located address.

(xii) OPTION

*N where N is the sum of the following:

Value	Effect	Availability		
		Load GO	Non-Load GO	Check
1	Display labels.	0 or 1	0 or 1†	1
2	Load and go operation (otherwise punch relocatable binary tape).	1	0	0
4	Clear the store up to the assembler.	0 or 1	0	0
8	Punch a binary loader tape.	0	0 or 1	0
16	Continue assembly at location 32.	0 or 1	0	0
32	Set dictionary below the program.	0 or 1	0	0
64	Check without assembly.	0	0	1

† 903C only.

N is an unsigned integer.

The presumed option is * + 3

ERROR INDICATIONS

Assembling

E0 F>15 or quasi-instruction not absolute.
E1 Contextual error.
E2 Error in Octal or Alphanumeric Group.
E3 Label declared twice.
E4 Global identifier list error.
E5 Store full or patch error.
E6 Number overflow.
E7 Buffer overflow (over 120 characters in a line).
E8 Illegal character.
E9 Stop code not first on line.
EG Global label error.
EL Literal error.
EP Patch or Obeyed instruction error.
EU Unlocated identifier.

Loading

FA Misread or mispunched tape.
FC Label used twice.
FD Misread or mispunched tape.
FE Store full.
FF Check sum failure.
FP Patch or Obeyed instruction error.
FU Unlocated identifier.

903 ALGOL HARDWARE REPRESENTATION

Algol Symbol	903 hardware representation	Recommended mode of writing the symbol
a–z	A–Z	a–z
×	*	*
÷	"DIV"	div
≤	"LE"	le
≥	"GE"	ge
≠	"NE"	ne
∧	"AND"	and
∨	"OR"	or
⊃	"IMPL"	impl
≡	"EQUIV"	equiv
¬	"NOT"	not
begin end etc.	"BEGIN" "END" etc.	begin end etc.
	"CODE" "ALGOL"	code algol

SUMMARY OF STANDARD PROCEDURES

In the following representation:

X is a real expression	H is a handler number
Z is a real variable	N is a block number
I, J are integer expressions	P is number of words in a block
M is an integer variable	E is integer variable := last block or error
A is an integer array	STAT is integer variable := status
B is a real array	T is an integer indicating operation required
Q is a boolean expression	
S is a string	

Real
abs (X)
exp (X)
ln (X)
sqrt (X)
sin (X)
cos (X)
arctan (X)
checkr (X)

Integer
entier (X)
sign (X)
range (B, I)
lowbound (B, I)
checki (I)

Boolean
checkb (Q)

Output
print X, I, S, . .
digits (I)
scaled (I)
freepoint (I)
aligned (I, J)
sameline
prefix (S)
punch (I)
outstring (A, M)
checks (S)

Input
read Z, M, . . .
reader (I)
instring (A, M)

Control
wait
stop

Magnetic Tape
MTCHECK (H, E, STAT, L)
MTOPEN (H, A, T, S)
MTWRITE (H, A, P)
MTREAD (H, A, N)
MTCLOSE (H, T)

903 ALGOL TRANSLATOR ERROR MESSAGES

1 read misplaced.
2 print misplaced.
3 Constant or expression in read list.
4 Wrong delimiter in switch declaration.
5 Illegal actual parameter.
6 Too many parameters to a procedure.
7 Illegal number.
8 Integer constant too big.
9 Two statements in the same block are prefixed by the same label.
10 Identifier or constant not as expected.
11 Letter, digit, "." or "$_{10}$" misused.
12 true or false follows an identifier or constant.
13 comment does not follow ";" or begin
14 <reserved>
15 Unrecognised basic symbol.
16 No assignment to the procedure identifier occurred within the body of a type procedure.
17 An identifier in the value or specification part of a procedure is not a formal parameter.
18 Use of undeclared identifier.
19 Illegal symbol.
20 Non procedure identifier used as a statement.
21 ":=" omitted from for clause.
22 Illegal use of label name.
23 Inadmissible array declaration.
24 <switch name> not an actual parameter nor preceded by goto.
25 Non type procedure as function designator.
26 switch misplaced.
27 Declaration without identifier.
28 ":=" preceded by a constant or used inside an expression.
29 ":" in type or switch declaration or misused.
30 Adjacent delimiters inadmissible.
31 Constant before ":=" or "[", or constant or name of a string in a read list.

32 Item other than a non-type procedure used as a statement.

33 Identifier or constant follows a closing round or square bracket.

34 Relation on each side of a simple arithmetic expression.

35 Illegal statement, delimiter misused.

36 Declaration starts incorrectly.

37 Error between for and ":=".

38 Missing array or switch name or "[" misplaced.

39 ":" misused in array declaration.

40 end misused.

41 Local identifier used in array bound.

42 goto follows an identifier or a constant.

43 Wrong for clause preceding do

44 for misused.

45 Misused Boolean constant.

46 Assignment to procedure identifier outside procedure body.

47 real integer or boolean misplaced.

48 Identifier declared twice in same block-head.

49 Blank parameter.

50 No begin at start of program.

51 Wrong number of subscripts or parameters.

52 ":=" appears in an actual parameter list.

53 Statement ends incorrectly.

54 Declaration follows statement.

55 ':', goto or for used in expression.

56 Illegal parameter comment or) precedes identifier.

57 Wrong use of delimiter.

58 Relational or logical operator used as an arithmetic operator.

59 Illegal use of logical operator.

60 Omission or error precedes begin or begin follows ":=".

61 "(" misplaced or missing procedure name.

62 Function designator as designational expression.

63 Misplaced declarator.
64 Subscripted variable as statement.
65 Illegal specifier.
66 Misused comma or colon in an expression.
67 if misused.
68 if used in type declaration.
69 Corresponding if has been omitted, or conditional expression without an else.
70 Corresponding then missing.
71 Illegal character in inner string or missng close quote in previous string.
72 array misplaced.
73 Left square bracket not preceded by an identifier.
74 Unmatched closing square brackets.
75 Upper bound missing in array declaration.
76 Illegal type declaration.
77 Illegal array list.
78 Corresponding for missing.
79 A jump is made to a label declared, but not placed in the block that ends here.
80 step, until or while misused in for list element.
81 Misused ")" other than in expression.
82 ")" misplaced or unmatched.
83 Program too complex, i.e. some statement is too complicated.
84 Wrong delimiter after procedure statement.
85 Program too large, i.e. contains too many names, labels, constants or switches.
86 Error before procedure.
87 Repeated formal parameter.
88 Wrong formal parameter delimiter.
89 <reserved>
80 Wrong delimiter in value or specification part.
91 Input buffer overflow, i.e. more than 120 characters in a line.
92 Formal parameter has not appeared in the specification part.
93 Declaration terminated by end or containing begin.

94	A formal parameter which is a switch, string or procedure is called by value.
95	Switch designator has more than one subscript.
96	Wrong for clause.
97	then misused.
98	Illegal character or parity error. The character is replaced by ← in the displayed line, but ↑ is not printed beneath it.
99	Current use of identifier inconsistent with previous uses.
100	Conditional expression needs parentheses.
101	Wrong delimiter after procedure identifier in procedure declaration.
102	No ";" between formal parameter part and value or specification part.
103	Commas or colons wrong in array bounds.
104	div used with a real argument.
105	Illegal parameter deliminter after a string.
106	Integer labels not allowed.
107	Recursive function calls not allowed.
108	An actual parameter which is a procedure has one of its parameters called by value.
109	Constant should not be used in procedure heading.
110	Wrong specification part.
111	Different number of parameters from previous use of formal procedure or wrong number of subscripts.
112	Mixed types in multiple assignment.

Note: The following errors will cause an Algol program tape to shoot through and unload the reader, instead of stopping at the end:

(1) No halt code at the end of a tape which is not the last tape of the program.

(2) Insufficient end's to match all the begin's in the program.

(3) Missing ` at the end of a string.

The following errors will cause the end of an Algol program to be found prematurely:

(1) Missing begin.

(2) end or the comment following end not followed by end, else or a semicolon causing a begin to be treated as comment.
These errors lead to a breakdown of the block structure of Algol and will usually cause many error messages to be displayed.

Any one error, particularly in a declaration, may cause many spurious errors to follow.
WARNING messages are output if identifiers are not used or the comment following end contains a delimeter.

903 ALGOL LOADER ERRORS

FA, FD, FF	misread or mispunched tape.
FC	Two procedures (at least one of which is a library procedure) have the same name.
FE	Program too large to load.
FU	Missing library procedure.

903 ALGOL ENTRY POINTS

Translator

8	Normal translation.
9	Continue after halt.
10	Report mode.
11	Include check functions.
12	Library mode (for large programs).
13	Library mode plus check functions.
14	Normal plus reports (903C only).

Interpreter

8	Normal, load translated program.
9	Continue after halt.
10	Run program.
11	Load R.L.B. machine code.
12	Reload library.
13	Load large program overwriting library.

ALGOL RUN TIME ERRORS

Error Number	*Meaning*	*Value substituted or Effect on continuation at 9*
1	Parameter mismatch	continuation not possible
2	Space overflow, e.g. too much claim on store for an array	continuation not possible
3	Integer overflow	continuation not possible
4	Jump error, i.e. switch subscript outside its permitted range	continuation not possible
5	Subscript error, i.e. address outside the store area allotted to the array referred to	continuation not possible
6	Illegal symbol inside inner string quotes (only discovered when the string is output)	newline is substituted
7	Attempt to output an unstandardised floating point number. (Probably the result of an error in a code procedure.)	*** is substituted
8	Illegal character or ' found when attempting to read a number	The read routine is re-entered
9	Real overflow	P is substituted
10	Invalid argument for sin(E) or cos(E), i.e. exponent greater than or equal to 18	0·0 is substituted
11	Negative argument for sqrt(E)	sqrt(abs(E)) is substituted
12	Argument $>$ 40 for exp(E)	P is substituted
13	Argument $\leq$ 0 for log(E)	0·0 is substituted
14	Illegal character on data tape	space is substituted

Error Number	*Meaning*	*Value substituted or Effect on continuation at 9*
15	Parity error on data tape	space is substituted
16	Overflow of input buffer in 920 Algol	The entire line of text is ignored
17	Numeric character found before ' when attempting to read a string	The instring operation is ignored
18	illegal form of number e.g. 2 decimal points	Ignore existing number, re-enter read routine
19	A ↑ B with A and B real and A<0	0·0 is substituted
20	Program corrupted, probably due to an error in a code procedure	Continuation not possible
21	An attempt has been made to assign a value to a formal parameter which is a constant	Continuation not possible
22	Range of array subscript bounds is negative	Continuation not possible
23	Instring or outstring error, i.e. Array not a one-dimensional integer array, subscript value is outside range or instring overfills the array	Continuation not possible
24	Attempt to jump to a label in an inner block or into a for loop	Continuation not possible
25	Translator used is incompatible with this version of the interpreter	Continuation not possible

Notes:

1. $P = (1-2^{-27}) \times 2^{63}$
   except for overflow of a negative real number where
   $P = -1{\cdot}0 \times 2^{63}$
2. When continuation is not possible entry at 9 results in a dynamic stop.

903 FORTRAN

General

A 903 FORTRAN program is translated into SIR and then run as a SIR program in conjunction with the 903 FORTRAN run-time routines (tape 2). Input is free-format, and mixed type working is allowed, with automatic type conversion. When a real number is converted to an integer it is rounded towards zero.

Summary of FORTRAN Statements

Control	*Specification*	*Real functions*
IF (E) n_1, n_2, n_3	GLOBAL, P1, P2, ... PN]	ALOG (X)
GOTO n	DIMENSION A(10), K(20,20)	SIN (X)
GOTO (n_1, n_2, ... n_X)M	COMMON A, K, M, ..., R	COS (X)
DO n M=L_1, L_2, L_3	FUNCTION	ATAN (X)
DO n M=L_1, L_2	SUBROUTINE	ABS (X)
CONTINUE	FORMAT	SQRT (X)
		EXP (X)
CALL	*Input-output*	*Integer function*
RETURN	READ (L, n) M, R, A, K, ...	IABS (I)
PAUSE	READ (L) M, R, A, K, ...	*SIR code*
STOP	WRITE (L, n) Z_1, Z_2, A, K, ...	CODE
END	WRITE (L) Z_1, Z_2, A, K, ...	FORTRAN

X represents a real expression
R represents a real variable
E represents any expression
I represents an integer expression
M represents a simple integer variable
L represents an integer constant or variable
n represents a statement number
A is a real array
K is an integer array
Z represents any expression not containing a Function

Field descriptors (r=repeat count)

Floating point	rF w.d or F w.d	(d characters after point)
	rE w.d or E w.d	(Exponent printed)
Integer	rI w or I w	(w=total print positions)
Alphanumeric	nH$h_1h_2h_3$...h_n	(n characters)
Spaces	nX	(n spaces)

1	Unacceptable form to left of = sign in an arithmetic statement.
3	Two operators in succession in an arithmetic expression.
5	Parentheses do not match.
6	Subscripted variable has not been declared in a DIMENSION statement or Function not in GLOBAL.
7	Unacceptable subscript form.
8	An unsubscripted array name has appeared in an arithmetic statement.
	In statements of type DO n M = L_1, L_2, L_3 errors:
9	n is omitted, or is not an acceptable statement number.
10	i is not an unsubscripted integer variable, or is omitted.
11	There is an impermissible number of Li's.
12	One of the Li's is of impermissible form.
13	DO statement has no = sign.
14	DO statements have intersecting loops.
15	A DO loop terminates with a GOTO or IF statement.
16	There is at least one unterminated DO loop when END is reached.
18	A number found where a variable expected.
19	No variable where one expected.
20	A statement number has more than 5 digits.
21	An integer or real constant is out of range.
22	A CALL statement has an unacceptable format.
23	A FUNCTION or SUBROUTINE statement has an unacceptable format.
24	The word FORTRAN has not been preceded by a CODE statement.
25	An error in GOTO or an IF statement.
26	A mis-spelt or otherwise corrupt or unrecognisable statement.
27	Statement too long or too complex to compile.
28	Program too long or too complex to compile.
29	Error in FORMAT statement.

31 Error in DIMENSION statement.

32 Error in COMMON statement.

33 A variable has been declared twice in a COMMON list.

38 In a DIMENSION statement, a bound exceeds 8192.

39 A variable has been declared twice in a DIMENSION list.

40 A sub-program has more than 18 parameters.

41 A continuation line has been used other than after a GLOBAL or FORMAT statement.

42 Too many sub-programs declared in a GLOBAL statement.

43 Error in a READ or WRITE statement.

44 A FUNCTION or SUBROUTINE statement has appeared in a sub-program.

45 Error in a GLOBAL statement.

46 Too many variables have been used in a program or sub-program.

47 (i) A sub-program does not contain a RETURN statement.

(ii) RETURN has occurred outside a sub-program.

48 A number appears against a statement that should not be labelled.

49 The channel number in a READ or WRITE statement is not an integer.

50 Halt code not preceded by new line, therefore last statement not translated.

FORTRAN TRANSLATOR QUERIES

101 Variable name has more than 6 characters (first 6 are significant).

105 CONTINUE statement not numbered.

106 Statement following GOTO or IF is not numbered.

108 Un-numbered FORMAT statement.

109 EQUIVALENCE statement used (statement ignored).

110 An executable statement has occurred before a COMMON or DIMENSION statement. (Statement may be wrongly compiled).

113 More than one GLOBAL statement in a program unit.

FORTRAN LOADING ERRORS

E1		Paper tape mispunched.
E3	Qn	(n is an integer). Two statements numbered n.
E3	<identifier>	Two different entities with this name.
E5		Program too large to load by batch mode.
EU	Qn	statement number n referred to in GOTO, IF, READ, WRITE statements, but does not occur in program.
EU	<sub-program name>	This sub program has not been loaded.
EU	<identifier>	This identifier has been used, but has not been assigned any value (possibly mis-spelt).

903 FORTRAN ENTRY POINTS

Translator (tape 1) for basic system only.

8	Translate to SIR code.
9	Continue after halt.
10	Report mode.

Tape 2

8	Load SIR code for batch mode.
9	Continue after wait.
10	Indicate program complete.
11	Run the program.
12	Load for batch mode, display store map.
13	Load SIR code for relocatable mode.
14	Load relocatable program tape.
15	Load an extra relocatable binary (user's library) tape.

FORTRAN RUN TIME ERRORS

(Continuation is only possible where specified. A=address in object code.)

E1 A P1 P2
Number of parameters in sub-program not compatible with the call.
P1=Number in definition, P2=Number in call.

E2 A Q1 Q2
Types of parameters in sub-program not compatible with the call.
Q1 represents types in definition, Q2 those in call.

E3 A 1 S1 S2
Array subscript out of range. S1=actual value of subscript.
On continuation S1=1 or S2.

E4 A 1 S1 S2
Computed GOTO variable out of range. S1=actual value.
On continuation S1=1 or S2.

E6 A C
Parity error on input. C=Decimal value of character.
On continuation, character ignored.

E7 A C
Illegal character in number. C=Decimal value of character.
On continuation, character treated as terminator.

E10 A L3
Increment in DO statement, L3$\leqslant$0
On continuation L3 is taken as +1.

E 11 A N
Attempt to output non-standardised real number on channel N.
On continuation, number is ignored.

E12 O
Compiler overwritten.

E13 O
Program incorrectly compiled (N.B. entry at 10).

E14 O
Translated program and systems incompatible.

E15 X A
Real number overflow. (Ignore X).
On continuation number=largest possible signed value.

E16 X A
Illegal QF instruction (corrupt code).

E17 A
Overflow on real to integer conversion.

E18 A Z
Z=78 In ALOG (X), X$\leqslant$0. On continuation, result=0.
Z=81 In SQRT (X), X<0.
On continuation, result=SQRT (ABS(X)).
Z=88 In EXP (X), X>2^{16}
Note A**B is calculated EXP (B*ALOG(A)).
On continuation, result of EXP(X) is X.

PERIPHERALS AND INTERFACES

The following facilities can be added to the basic 903 at any time, enabling the system to be adapted to changing requirements:

Multiplexer: This unit enables more than one peripheral to be fitted to the system. It consists of 8 channels which may be extended to take an NPL interface.

Core Store: Units of 8,192 words may be used to increase the directly addressable core memory to a maximum of 65,536 words.

Digital Plotter: Available with 14-inch or 30-inch paper width, and operating at 300 increments per second, this unit enables the computer output to be presented in the form of permanently recorded graphs, charts and diagrams.

N.P.L. Interface: The computer can be fitted with the standard N.P.L. electrical interface which makes available the increasingly wide range of peripherals conforming to this standard.

Store Access Control Unit: Permits the transfer of data out of or into the extra store units, as the computer is carrying out other tasks.

Magnetic Tape Unit: A 9,000 character per second unit, consisting of electronic controller and mechanism compatible with the European Computer Manufacturers' Association standards.

4100 Interface: This allows standard 4100 peripherals such as card readers and punches to be attached to the 903.

Lineprinter: A 300-line per minute, 120 character per line lineprinter is available.

The purpose of these tables is to assist in the

1. Select the highest *multiple* of 64 less than (or equal
2. Set the first (left-hand) 7 buttons to the binary
3. Set the last (right-hand) 6 buttons to the binary

TABLE A

Multiple of 64	*Binary equivalent*	*Multiple of 64*	*Binary equivalent*	*Multiple of 64*	*Binary equivalent*
0	0000000	2048	0100000	4096	1000000
64	0000001	2112	0100001	4160	1000001
128	0000010	2176	0100010	4224	1000010
192	0000011	2240	0100011	4288	1000011
256	0000100	2304	0100100	4352	1000100
320	0000101	2368	0100101	4416	1000101
384	0000110	2432	0100110	4480	1000110
448	0000111	2496	0100111	4544	1000111
512	0001000	2560	0101000	4608	1001000
576	0001001	2624	0101001	4672	1001001
640	0001010	2688	0101010	4736	1001010
704	0001011	2752	0101011	4800	1001011
768	0001100	2816	0101100	4864	1001100
832	0001101	2880	0101101	4928	1001101
896	0001110	2944	0101110	4992	1001110
960	0001111	3008	0101111	5056	1001111
1024	0010000	3072	0110000	5120	1010000
1088	0010001	3136	0110001	5184	1010001
1152	0010010	3200	0110010	5248	1010010
1216	0010011	3264	0110011	5312	1010011
1280	0010100	3328	0110100	5376	1010100
1344	0010101	3392	0110101	5440	1010101
1408	0010110	3456	0110110	5504	1010110
1472	0010111	3520	0110111	5568	1010111
1536	0011000	3584	0111000	5632	1011000
1600	0011001	3648	0111001	5696	1011001
1664	0011010	3712	0111010	5760	1011010
1728	0011011	3776	0111011	5824	1011011
1792	0011100	3840	0111100	5888	1011100
1856	0011101	3904	0111101	5952	1011101
1920	0011110	3968	0111110	6016	1011110
1984	0011111	4032	0111111	6080	1011111

setting of binary addresses on the word generator.

to) the required address, and work out the difference (if any).
equivalent of the multiple, working from Table A.
equivalent of the difference, working from Table B.

TABLE B

Multiple of 64	*Binary equivalent*	*Difference*	*Binary equivalent*	*Difference*	*Binary equivalent*
6144	1100000	0	000000	32	100000
6208	1100001	1	000001	33	100001
6272	1100010	2	000010	34	100010
6336	1100011	3	000011	35	100011
6400	1100100	4	000100	36	100100
6464	1100101	5	000101	37	100101
6528	1100110	6	000110	38	100110
6592	1100111	7	000111	39	100111
6656	1101000	8	001000	40	101000
6720	1101001	9	001001	41	101001
6784	1101010	10	001010	42	101010
6848	1101011	11	001011	43	101011
6912	1101100	12	001100	44	101100
6976	1101101	13	001101	45	101101
7040	1101110	14	001110	46	101110
7104	1101111	15	001111	47	101111
7168	1110000	16	010000	48	110000
7232	1110001	17	010001	49	110001
7296	1110010	18	010010	50	110010
7360	1110011	19	010011	51	110011
7424	1110100	20	010100	52	110100
7488	1110101	21	010101	53	110101
7552	1110110	22	010110	54	110110
7616	1110111	23	010111	55	110111
7680	1111000	24	011000	56	111000
7744	1111001	25	011001	57	111001
7808	1111010	26	011010	58	111010
7872	1111011	27	011011	59	111011
7936	1111100	28	011100	60	111100
8000	1111101	29	011101	61	111101
8064	1111110	30	011110	62	111110
8128	1111111	31	011111	63	111111

POWERS OF 2 IN DECIMAL

2^n	n	2^{-n}
2	1	·5
4	2	·25
8	3	·125
16	4	·062 5
32	5	·031 25
64	6	·015 625
128	7	·007 812 5
256	8	·003 906 25
512	9	·001 953 125
1 024	10	·000 976 562 5
2 048	11	·000 488 281 25
4 096	12	·000 244 140 625
8 192	13	·000 122 070 312 5
16 384	14	·000 061 035 156 25
32 768	15	·000 030 517 578 125
65 536	16	·000 015 258 789 062 5
131 072	17	·000 007 629 394 531 25
262 144	18	·000 003 814 697 265 625
524 288	19	·000 001 907 348 632 812 5
1 048 576	20	·000 000 953 674 316 406 25
2 097 152	21	·000 000 476 837 158 203 125
4 194 304	22	·000 000 238 418 579 101 562 5
8 388 608	23	·000 000 119 209 289 550 781 25
16 777 216	24	·000 000 059 604 644 775 390 625
33 554 432	25	·000 000 029 802 322 387 695 313
67 108 864	26	·000 000 014 901 161 193 847 656
134 217 728	27	·000 000 007 450 580 596 923 828
268 435 456	28	·000 000 003 725 290 298 461 914
536 870 912	29	·000 000 001 862 645 149 230 957
1 073 741 824	30	·000 000 000 931 322 574 615 479
2 147 483 648	31	·000 000 000 465 661 287 307 739
4 294 967 296	32	·000 000 000 232 830 643 653 870
8 589 934 592	33	·000 000 000 116 415 321 826 935
17 179 869 184	34	·000 000 000 058 207 660 913 467
34 359 738 368	35	·000 000 000 029 103 830 456 734
68 719 476 736	36	·000 000 000 014 551 915 228 367
137 438 953 472	37	·000 000 000 007 275 957 614 183
274 877 906 944	38	·000 000 000 003 637 978 807 092
549 755 813 888	39	·000 000 000 001 818 989 403 546
1 099 511 627 776	40	·000 000 000 000 909 494 701 773

Useful Constants

π	= 3·141 592 653 590	$1/\pi$	= 0·318 309 886 184
$\log_{10}e$	= 0·434 294 481 903	$\log_e 10$	= 2·302 585 092 994
$\log_{10}2$	= 0·301 029 995 664	e	= 2·718 281 828 459
$\sqrt{2}$	= 1·414 213 562 373	$\sqrt{3}$	= 1·732 050 807 569
1 radian	= 57·295 779 513 082°	1°	= 0·017 453 292 520 radian

ELLIOTT COMPUTERS MARKETING
ENGLISH ELECTRIC COMPUTERS LTD
COMPUTER HOUSE, EUSTON CENTRE
LONDON N.W.1
Telephone: 01-387 7030

CATALOGUE No. 903 ISSUE 5 APRIL 1968