903/905/920
Useful Notes

Book No. 106

Copy No. 23

Amendment No. 2

AIRBORNE COMPUTING DIVISION
ELLIOTT FLIGHT AUTOMATION LIMITED

CONTENTS.

Tape Reader Modes.

On 903, 905, and 920 computers, the paper tape reader can be operated in 3 'Modes'. These modes are :-

Mode 1. The tape reader input instruction, 15 2048, causes the existing contents of the accumulator to be left-shifted 7 places, although the Q-register remains unchanged.

The 7 least significant bits of the accumulator are then set to the character on the tape reader, ignoring track 5.

Mode 2. The tape reader input instruction, 15 2048, causes the existing contents of the accumulator to be left-shifted 7 places, although the Q-register remains unchanged.

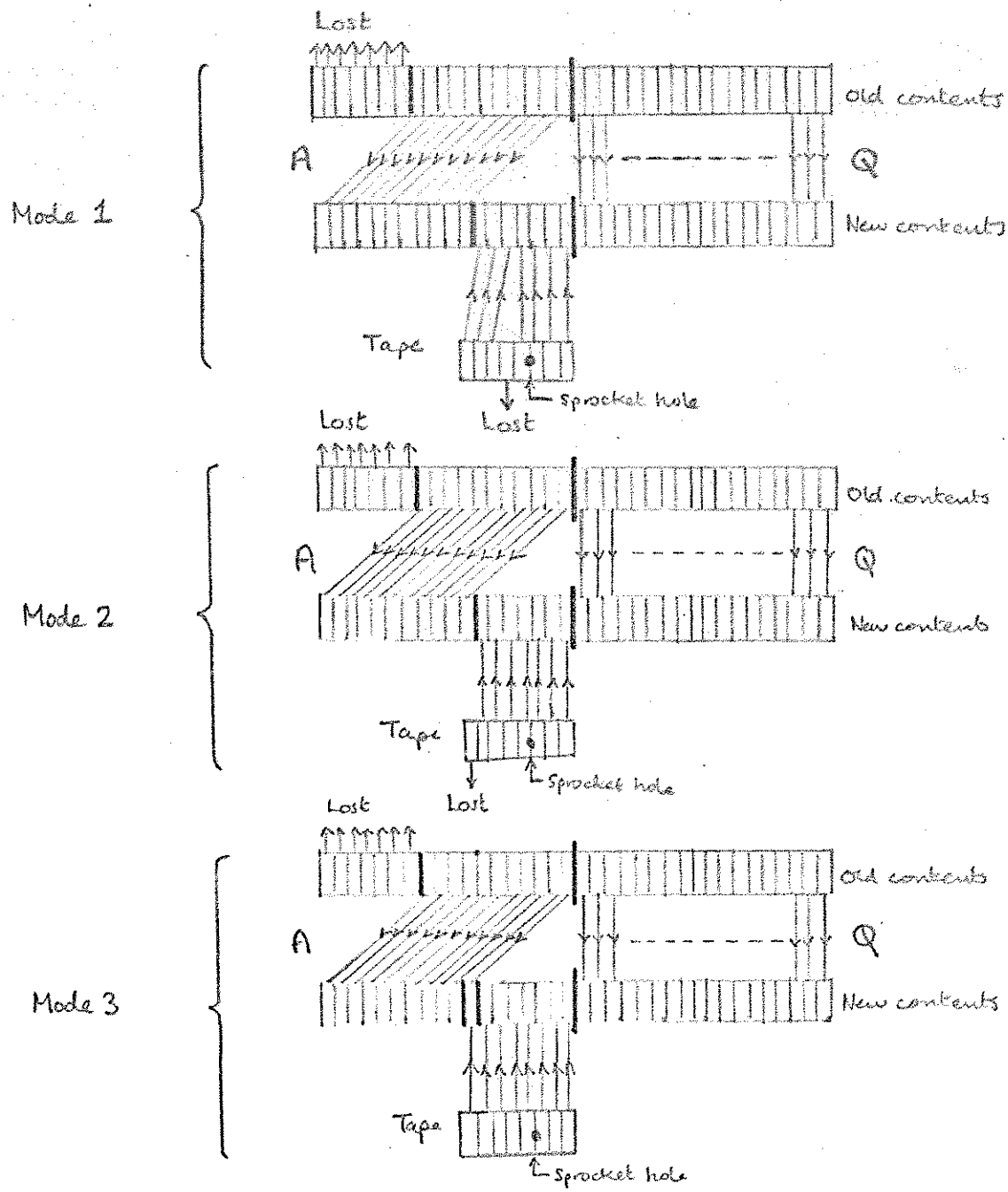
The 7 least significant bits of the accumulator are then set to the character on the tape reader, ignoring track 8.

Mode 3. The tape reader input instruction, 15 2048, causes the existing contents of the accumulator to be left-shifted 7 places although the Q-register remains unchanged.

The 8 least significant bits of the accumulator are then set to the character on the tape reader.

Thus bit 8 will become a '1' if bit 1 was '1' before obeying the instruction OR if track 8 of the tape has a hole in it.

Diagramatically :-



These 3 modes are not all available on all 903, 905, and 920 computers. There are 3 common configurations:-

A. Computers (other than 920A) having the MC*60 type paper tape station. (E.G. A 920B computer with an MCB60).

The paper tape station has a mode-switch labelled '1-2-3' and all 3 modes are available.

This type of paper tape station is not used with an on-line teleprinter.

B. Computers (other than 920A) having the MC*66F type paper tape station (E.G. A 920M computer with an MCM66F).

This paper tape station has switches labelled 'READER-AUTO-TELEPRINTER-AUTO-PUNCH', and may be used with an on-line teleprinter.

It only provides tape reader mode 3.

To use programmes written for mode 1 or 2 with this type of tape station, it is necessary to fit a mode-switch in the cable between the reader and tape station.

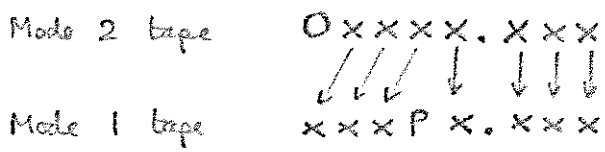
C. 920A computers.

These usually have 'Mode 1' input only.

Mode 2 input can be obtained by fitting a mode switch in the cable between the reader and tape station.

Mode 3 input can only be obtained by modifying the computer itself.

It should be noted that any tape intended to be read in mode 3 can in fact be read in mode 2 if it has no holes in track 8; and that any tape which can be read in mode 2 can be read in mode 1 by copying it (on a machine having mode 2 or 3) so as to rearrange tracks 5, 6, 7 & 8, thus:-



where P is punched, preferably, to give even parity.

Telecodes.

Three Telecodes are listed on the following pages.

The first of these is "920 Telecode". This code is used on the Elliott 503, and on some Elliott 900-series 18-bit machines (e.g. 920).

The second of these is "903 Telecode". This code is used on the Elliott 4100, and on most Elliott 900-series 18-bit machines (e.g. 903 and 920).

The third is "900 Series Telecode". This code is used on the newer computers of the 900-series, viz. 905, 920C, 902, 102C, and will eventually be the standard for the whole series.

Most of the early 920 Telecode software uses "Mode 1", and thus cannot be used on current 900 Series machines having "Mode 3" only; although it is gradually being re-written for mode 3.

Most new 900-Series software is being written in "900-Series" code. It is likely that only a limited number of these programs will be modified to operate in 920 Telecode.

It should be noted that the "903" and "900-Series" Telecodes are nearly identical to one another and to ISO and ASCII codes.

920 TELECODE

Character	Numerical Value	Elliott (920) Meaning
Zone 0		
00000.000	0	Blank
00010.001	1	
00010.010	2	New Line (CRFL)
00000.011	3	Throw
00010.100	4	Tab
00000.101	5	
00000.110	6	
00010.111	7	
00011.000	8	(
00001.001	9	)
00001.010	10	,
00011.011	11	£
00001.100	12	:
00011.101	13	&
00011.110	14	"
00001.111	15	/
Zone 1		
00110.000	16	0
00100.001	17	1
00100.010	18	2
00110.011	19	3
00100.100	20	4
00110.101	21	5
00110.110	22	6
00100.111	23	7
00101.000	24	8
00111.001	25	9
00111.010	26	10
00101.011	27	11
00111.100	28	=
00101.101	29	+
00101.110	30	−
00111.111	31	.

Character	Numerical Value	Elliott (920) Meaning
Zone 2		
01010.000	32	;
01000.001	33	A
01000.010	34	B
01010.011	35	C
01000.100	36	D
01010.101	37	E
01010.110	38	F
01000.111	39	G
01001.000	40	H
01011.001	41	I
01011.010	42	J
01001.011	43	K
01011.100	44	L
01001.101	45	M
01001.110	46	N
01011.111	47	O
Zone 3		
01100.000	48	P
01110.001	49	Q
01110.010	50	R
01100.011	51	S
01110.100	52	T
01100.101	53	U
01100.110	54	V
01110.111	55	W
01111.000	56	X
01101.001	57	Y
01101.010	58	Z
01111.011	59	
01101.100	60	
01111.101	61	
01111.110	62	Vert Bar
01101.111	63	

Character	Numerical Value	Elliott (920) Meaning
Zone 4		
10010.000	64	Space
10000.001	65	
10000.010	66	
10010.011	67	
10000.100	68	
10010.101	69	
10010.110	70	
10000.111	71	
10001.000	72	
10011.001	73	
10011.010	74	
10001.011	75	
10011.100	76	stop code
10001.101	77	
10001.110	78	
10011.111	79	
Zone 5		
10100.000	80	
10110.001	81	
10110.010	82	
10100.011	83	
10110.100	84	
10100.101	85	
10100.110	86	
10110.111	87	
10111.000	88	[
10101.001	89	]
10101.010	90	10 (suffix)
10111.011	91	<
10101.100	92	>
10111.101	93	↑
10111.110	94	~
10101.111	95	%

Character	Numerical Value	Elliott (920) Meaning
Zone 6		
11000.000	96	?
11010.001	97	a
11010.010	98	b
11000.011	99	c
11010.100	100	d
11000.101	101	e
11000.110	102	f
11010.111	103	g
11011.000	104	h
11001.001	105	i
11001.010	106	j
11011.011	107	k
11001.100	108	l
11011.101	109	m
11011.110	110	n
11001.111	111	o
Zone 7		
11110.000	112	p
11100.001	113	q
11100.010	114	r
11110.011	115	s
11100.100	116	t
11110.101	117	u
11110.110	118	v
11100.111	119	w
11101.000	120	x
11111.001	121	y
11111.010	122	z
11101.011	123	
11111.100	124	
11101.101	125	
11101.110	126	horiz. bar
11111.111	127	Erase

The numerical values above are those obtained with a mode 1 tape reader instruction, i.e. the tracks have values:-

X X X X X . X X X
64 32 16 0 8 4 2 1

Note that "Vert Bar" & "Horiz. Bar" are "none-escaping" symbols.

903 TELECODE

PAPER TAPE AND INTERNAL CODES

Code Value	Value with Parity	Telecode Character	Binary Pattern	SIR Internal code Octal	SIR Internal code Decimal	Code Value	Value with Parity	Telecode Character	Binary Pattern	SIR Internal code Octal	SIR Internal code 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[1],	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],	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],	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	01011·011	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],	01011·111	77	63
32	160	Space	10100·000	00	0	96	96	@[5]	01100·000		
33	33	![1/2]	00100·001			97	225	a	11100·001	41	33
34	34	"	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	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	10	00111·111	37	31	127	255	erase	11111·111		

[1]Ignored by Teleprinter
[2]New line on Flexowriter
[3]Ignored by Flexowriter
[5]Upper case on Teleprinter

The "Code value" above is that obtained by ignoring track 8 of the tape, whereas the "value with parity" includes this track.

"900 SERIES" TELECODE

1. Introduction

This note defines the character codes proposed to be used with future 900 series machines and, hopefully, their successors. The 8 bit code to be used is in line with recommendations which have been agreed between international standards bodies in the fields of data-processing (I.S.O.) and communications (C.C.I.T.T.). A British Standard based on the international recommendations is being prepared, the proposed code accords with the latest draft of this standard.

The proposed code involves minor changes from the Elliott 4100 code used currently; the changes are such that appears possible in general to work with mixtures of hardware and software using these two codes at least within the former Elliott organisation. There are major differences between the proposed code and the Elliott 503 code which is still in use in some areas of activity.

It is not anticipated that any further code changes will be necessary in view of the degree of agreement now reached internationally on this subject. The proviso must be made however that the final British Standard on the subject might be different from the current draft and a change may then be desirable to accord with the standard.

2. 8 bit character code

This code is compatible with I.S.C., C.C.I.T.T. and E.C.M.A. recommendations.

Binary 87654321	Num. Value	Assign -ment	Binary 87654321	Num. Value	Assign -ment
00000000	0	NULL	10100000	32	Space
10000001	1		00100001	33	!
10000010	2		00100010	34	"
00000011	3		10100011	35	£
10000100	4		00100100	36	$
00000101	5		10100101	37	%
00000110	6		10100110	38	&
10000111	7	BELL	00100111	39	' or ¹
10001000	8	BACK-SPACE	00101000	40	(
00001001	9	TAB	10101001	41	)
00001010	10	LF	10101010	42	*
10001011	11	VT	00101011	43	+
00001100	12	FF	10101100	44	,
10001101	13	CR	00101101	45	-
10001110	14	SHIFT OUT	00101110	46	.
00001111	15	SHIFT IN	10101111	47	/
10010000	16		00110000	48	0
00010001	17		10110001	49	1
00010010	18		10110010	50	2
10010011	19		00110011	51	3
00010100	20	HALT	10110100	52	4
10010101	21		00110101	53	5
10010110	22		00110110	54	6
00010111	23		10110111	55	7
00011000	24		10111000	56	8
10011001	25		00111001	57	9
10011010	26		00111010	58	:
00011011	27		10111011	59	;
10011100	28		00111100	60	<
00011101	29		10111101	61	=
00011110	30		10111110	62	>
10011111	31		00111111	63	?

Binary 87654321	Num. Value	Assign -ment	Binary 87654321	Num. Value	Assign -ment
11000000	64	@	01100000	96	`
01000001	65	A	11100001	97	a
01000010	66	B	11100010	98	b
11000011	67	C	01100011	99	c
01000100	68	D	11100100	100	d
11000101	69	E	01100101	101	e
11000110	70	F	01100110	102	f
01000111	71	G	11100111	103	g
01001000	72	H	11101000	104	h
11001001	73	I	01101001	105	i
11001010	74	J	01101010	106	j
01001011	75	K	11101011	107	k
11001100	76	L	01101100	108	l
01001101	77	M	11101101	109	m
01001110	78	N	11101110	110	n
11001111	79	O	01101111	111	o
01010000	80	P	11110000	112	p
11010001	81	Q	01110001	113	q
11010010	82	R	01110010	114	r
01010011	83	S	11110011	115	s
11010100	84	T	01110100	116	t
01010101	85	U	11110101	117	u
01010110	86	V	11110110	118	v
11010111	87	W	01110111	119	w
11011000	88	X	01111000	120	x
01011001	89	Y	11111001	121	y
01011010	90	Z	11111010	122	z
11011011	91	[	01111011	123	{
01011100	92	\	11111100	124	|
11011101	93	]	01111101	125	}
11011110	94	↑ or ^	01111110	126	~
01011111	95	←	11111111	127	E

TAB = Horizontal tabulate
LF = Line Feed
VT = Vertical tabulate
FF = form feed
CR = carriage return
E = Delete

3. Notes on character set

Where alternatives are listed (i.e. ! or ′ and ↑ or ^) these should be regarded as typographical variations of the same basic character. ′ and ↑ are the preferred versions but expediency may require the use of the alternatives in some hardware.

The preferred printed shapes for letter O and zero are for letter O to be squared (O) and zero to be narrowed (0). (This is also I.C.L. 1900, and 4100 practice.)

4. Comparison with 4100/903 code

Compared with the 4100/903 code the following alterations have taken place:-

(i) £ has been moved from value 92 to value 35 replacing ½ in the latter position, \ replaces £ as value 92.

(ii) $_{10}$ has been replaced by ? at value 63.

(iii) @ (formerly value 96) and ` (formerly value 64) have been interchanged.

(iv) { | } — have been added in previously unassigned positions.

(v) ←has been replaced by ___ at value 95.

5. Comparison with draft British Standard

The code is believed to agree with the current draft British Standard. The draft B.S. however allows $_{10}$ or ½, as alternatives to \ at value 92, it is not proposed to adopt either of these, it is believed that no other manufacturer is yet doing this.

6. Comparison with U.S. standard

The graphic representations of certain codes differ as follows -

Value	Proposed	U.S.A.S.I.I.
35	£	#
124	|	!
126	—	~

These would thus appear to be an infinitesimal problem in using equipment to the U.S. standard with 900 series.

16

A.C.D. Internal Code, 1/12/69.

The purpose of defining an "internal code" is to provide a standard software interface for the handling of symbols between a program and peripherals, or between sections of a program.

Many programs contain "built-in" routines for the input and output of symbols in various telecodes to various peripherals, these routines include such facilities as ignoring blanks and erases, converting from one telecode to another, reading tape one line at a time into a buffer, etc.

By dividing a program into a "main part" and "character input & output subroutine" which communicate a standard internal code, two advantages are obtained :-

① Writers of new programs can lift a character subroutine "off the shelf," instead of writing a "built-in" routine.

② Users of existing programs can easily change them to operate in a different Telecode or via a different peripheral, by writing just one new character subroutine.

The internal code defined on the next page is now standard; character input and output subroutines operating in this code for use with Telecode tapes punched in the 3 codes described earlier in this book are described elsewhere.

The symbols listed in the internal code are all those symbols used by current software and all those occupying the same position in "903" and "900 series" code. The internal code is thus very simply related to :-

903 Telecode
"900-Series" Telecode
ISO code
ASCII code
SIR internal code

It is NOT simply related to 920 Telecode.

Note that "£" and "↑" both have 2 positions in the code. Input programs should tolerate both values, and output programs should use values 35 and 96.

(In new programs the following values can be avoided so as to enable 903 and 900 series Teletype machines to be used interchangeably:-

35, 92; 63; 64, 96; 95.)

INTERNAL CODE, 1/12/69

0		32	Ⓢ	64	`	96	`
1		33	!	65	A	97	a
2		34	"	66	B	98	b
3		35	£	67	C	99	c
4		36	$	68	D	100	d
5		37	%	69	E	101	e
6		38	&	70	F	102	f
7	Ⓑ	39	'	71	G	103	g
8		40	(	72	H	104	h
9	Ⓣ	41	)	73	I	105	i
10	Ⓝ	42	*	74	J	106	j
11	Ⓥ	43	+	75	K	107	k
12		44	,	76	L	108	l
13		45	-	77	M	109	m
14		46	.	78	N	110	n
15		47	/	79	O	111	o
16		48	0	80	P	112	p
17		49	1	81	Q	113	q
18		50	2	82	R	114	r
19		51	3	83	S	115	s
20	Ⓗ	52	4	84	T	116	t
21		53	5	85	U	117	u
22		54	6	86	V	118	v
23		55	7	87	W	119	w
24		56	8	88	X	120	x
25		57	9	89	Y	121	y
26		58	:	90	Z	122	z
27		59	;	91	[	123	
28		60	<	92	£	124	
29		61	=	93	]	125	
30		62	>	94	↑	126	
31		63	⑩	95	←	127	

Ⓑ Bell
Ⓣ Horizontal Tab
Ⓝ Newline, or C/R+L/F
Ⓥ Vertical Tab (Throw)
Ⓗ Halt or Stopcode
Ⓢ Space

6-BIT CODE.

For efficient use of store the user may require to store characters in a 6-bit code. The following method is recommended :-

Conversion from 7-bit to 6-bit code

Values 32 to 95	:	replace by 0 to 63.	(subtract 32)
Values 96 to 127	:	replace by 32 to 63.	(subtract 64)
Value 10, (N)	:	replace by 1	
Value 20, (H)	:	replace by 63	
Other values	:	replace by 0, (S).	

Conversion from 6-bit to 7-bit code.

Value 1, (K)	:	replace by 10.
Value 63, (H)	:	replace by 20.
other values	:	Add 32.

Thus the 6-bit code contains the same symbols as the 7-bit code except that :-

Lower case letters are replaced by upper case
"Tab" becomes "Space" (Also "Bell" & "Vert. Tab")
"!" and "←" are effectively lost.

Useful Numbers

TABLES OF BINARY EQUIVALENTS

Multiple of 64	*Binary Equivalent*	*Multiple of 64*	*Binary Equivalent*	*Multiple of 64*	*Binary Equivalent*	*Multiple of 64*	*Binary Equivalent*
0	0000000	2048	0100000	4096	1000000	6144	1100000
64	0000001	2112	0100001	4160	1000001	6208	1100001
128	0000010	2176	0100010	4224	1000010	6272	1100010
192	0000011	2240	0100011	4288	1000011	6336	1100011
256	0000100	2304	0100100	4352	1000100	6400	1100100
320	0000101	2368	0100101	4416	1000101	6464	1100101
384	0000110	2432	0100110	4480	1000110	6528	1100110
448	0000111	2496	0100111	4544	1000111	6592	1100111
512	0001000	2560	0101000	4608	1001000	6656	1101000
576	0001001	2624	0101001	4672	1001001	6720	1101001
640	0001010	2688	0101010	4736	1001010	6784	1101010
704	0001011	2752	0101011	4800	1001011	6848	1101011
768	0001100	2816	0101100	4864	1001100	6912	1101100
832	0001101	2880	0101101	4928	1001101	6976	1101101
896	0001110	2944	0101110	4992	1001110	7040	1101110
960	0001111	3008	0101111	5056	1001111	7104	1101111
1024	0010000	3072	0110000	5120	1010000	7168	1110000
1088	0010001	3136	0110001	5184	1010001	7232	1110001
1152	0010010	3200	0110010	5248	1010010	7296	1110010
1216	0010011	3264	0110011	5312	1010011	7360	1110011
1280	0010100	3328	0110100	5376	1010100	7424	1110100
1344	0010101	3392	0110101	5440	1010101	7488	1110101
1408	0010110	3456	0110110	5504	1010110	7552	1110110
1472	0010111	3520	0110111	5568	1010111	7616	1110111
1536	0011000	3584	0111000	5632	1011000	7680	1111000
1600	0011001	3648	0111001	5696	1011001	7744	1111001
1664	0011010	3712	0111010	5760	1011010	7808	1111010
1728	0011011	3776	0111011	5824	1011011	7872	1111011
1792	0011100	3840	0111100	5888	1011100	7936	1111100
1856	0011101	3904	0111101	5952	1011101	8000	1111101
1920	0011110	3968	0111110	6016	1011110	8064	1111110
1984	0011111	4032	0111111	6080	1011111	8128	1111111

Difference	*Binary Equivalent*	*Difference*	*Binary Equivalent*
0	000000	32	100000
1	000001	33	100001
2	000010	34	100010
3	000011	35	100011
4	000100	36	100100
5	000101	37	100101
6	000110	38	100110
7	000111	39	100111
8	001000	40	101000
9	001001	41	101001
10	001010	42	101010
11	001011	43	101011
12	001100	44	101100
13	001101	45	101101
14	001110	46	101110
15	001111	47	101111
16	010000	48	110000
17	010001	49	110001
18	010010	50	110010
19	010011	51	110011
20	010100	52	110100
21	010101	53	110101
22	010110	54	110110
23	010111	55	110111
24	011000	56	111000
25	011001	57	111001
26	011010	58	111010
27	011011	59	111011
28	011100	60	111100
29	011101	61	111101
30	011110	62	111110
31	011111	63	111111

SOME USEFUL CONSTANTS

$\pi = 3.141\ 592\ 653\ 590$
$\log_{10} e = 0.434\ 294\ 481\ 903$
$\log_{10} 2 = 0.301\ 029\ 995\ 664$
$\sqrt{2} = 1.414\ 213\ 562\ 373$
1 radian $= 57.295\ 779\ 513\ 082°$

$1/\pi = 0.318\ 309\ 886\ 184$
$\log_e 10 = 2.302\ 585\ 092\ 994$
$e = 2.718\ 281\ 828\ 459$
$\sqrt{3} = 1.732\ 050\ 807\ 569$
$1° = 0.017\ 453\ 292\ 520$ radian

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

A.C.D. 900-Series 18-bit Binary Tape format, 1/4/70.

Whilst all 900-series 18-bit Binary tapes can be, by definition, read into a 900-Series 18-bit computer using initial instructions, there are many formats which these tapes can take. The purpose of defining a standard format is to enable those programs which read binary tapes as data to be simplified. (The 'VERIFY' program used to compare a binary tape with the store is one such program).

Binary tapes punched in the 1/4/70 format are suitable for loading directly by initial instructions using reader mode 2 or 3. Only tracks 1-7 of the tape are used; track 8 must be blank. There is no parity track.

The tapes have 3 sections, viz a loader, a body, and a tail. The loader is read, by initial instructions, into the locations adjacent to initial instructions. The loader then reads the body of the tape into store, and simultaneously forms a store sum-check. Initial instructions are then re-entered to read in the tail (which partially over-writes the loader), the tail compares the sum-check just formed by the loader with that punched in the tail. In the case of error an indication is given by continuous output on the punch.

The loader defined below is capable of loading program into any locations of the first store module except 0 - 1 and 8167 - 8191, and into any locations in extended store modules, in any order. After the sum-check has been performed the program can be automatically entered at any location in the first module, if required.

In the body of the tape, a binary word is represented by 3 tape characters. If the bits of the word are called X18, X17, X16, ... X3, X2, X1, (where X18 is the sign bit) then the corresponding tape characters are:-

0	$\overline{X18}$	X18	X18	X18	.	X17	X16	X15
0	X14	X13	X12	X11	.	X10	X9	X8
0	X7	X6	X5	X4	.	X3	X2	X1

where the full-stops represent the sprocket track. As already mentioned, the 8th track (on the left in the diagram) is left blank. As the first character contains both X18 and its inverse $\overline{X18}$, it is impossible for the first character of the word to be a blank, even if the word itself is zero. (Note, however, that the other two characters can be blank). The significance of this is explained later.

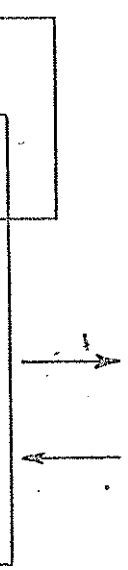
A word can be punched in the form described above by the instructions:-

```
14   8178
 1   &100
15   6144
 4   +0
14   7
15   6144
 4   +0
14   7
15   6144
```

and can be read back in using 3 consecutive '15 2048' instructions.

The loader uses 13 locations, and, using the notation of SIR, is:-

```
(8167)    SC1      > 1
(8168)             4        INSTR
(8169)    INSTR    5        OBEY
(8170)             4        +0
(8171)            15        2048
(8172)             7        ;-4
(8173)            15        2048
(8174)            15        2048
(8175)    OBEY     > 1
(8176)             1        SC1
(8177)             5        SC1
(8178)            10        OBEY
(8179)             8        ;-9
```



The loader is entered, from initial instructions, at 8177 with both the Accumulator and B-register clear. Thus the first action of the loader is to clear the sum-check, SC1. What happens thereafter depends upon the body of the tape. If a blank character is read, location OBEY is set to '5 OBEY'. If a non-blank character is read, the remaining 2 characters of a word are read in, then the instruction in location OBEY is obeyed, the word is added to SC1, and OBEY is incremented.

To load words into store locations N, N+1, and N+2 the first word punched on the tape, after a blank, is '/5 N-1' followed by the 3 words to be loaded. Since the B-register is clear the modification has no effect, but its use will be explained later.

For example, if the body of the tape consisted of:-

A blank
3 characters forming the word /5 7
3 characters forming the word 4 36
3 characters forming the word 5 37
3 characters forming the word 8 10
A blank
3 characters forming the word /5 35
3 characters forming the word +300

then the instructions 4 36, 5 37, and 8 ;+0 would be stored in locations 8, 9 and 10; and the constant +300 would be stored in location 36.

The B-register is used to load extended store modules, and is set by punching a blank followed by a '5 0' instruction and the required value. If the above example had been preceded by:

A blank
3 characters forming the word 5 0
3 characters forming the word +8192

then locations 8200, 8201, 8202 and 8228 would have been loaded. Remember that the contents of OBEY are incremented between reading each word, so that '5 0' is obeyed as '5 1'.

The body of the tape ends with a blank followed by an '8 8180' instruction which, when placed in OBEY and incremented, causes a jump to initial instructions at 8181, to read the tail in. Note that, between reading in the '8 8180' and obeying it, one more word will be read in, which will not be added to the sum-check. The value of this word is of no consequence, and is accordingly defined to be +0.

The tail uses 6 of the locations previously used by the loader, and, using the notation of SIR, is:-

```
(8174)    SC2    Sumcheck formed by punching program
(8175)           7   ;+0 <-----  or 7 START
(8176)           15  6144    |
(8177)           4   SC2   <--|
(8178)           2   SC1     |
(8179)           8   ;-4 -----|
```

The tail is entered, from initial instructions, at 8177. If SC1 and SC2 are the same, either a dynamic stop occurs or the program just loaded is entered by the obeyed '7' instruction. Otherwise continuous punch output occurs. Note that since the sum-check SC2 is punched in the tail of the tape, rather than the loader, the program punching the tape need not know the value of the sumcheck in advance, but can form it whilst punching the tape.

A tape in 1/4/70 format is preceded by a clear-store if one is needed, but not otherwise. Thus, a store-dump of all non-zero locations needs a clear store, but a store-dump of ALL locations does not. A correctly-written program assembled by 2-PASS SIR usually does NOT require a clear-store.

For a single store module, the clear-store program is, in SIR notation:-

```
(8175)              8   ;+2 <-----
(8176)              8   8181    |
(8177)    OBEY      5   2   <---|
(8178)              10  OBEY
(8179)              8   ;-4 -----|
```

This is entered, from initial instructions, at 8177 with the Accumulator clear. It clears locations 2 to 8175, then, when 8175 is cleared, instruction 8176 is obeyed causing a jump back to initial instructions.

To clear extended stores the above clear-store is preceded by:-

```
(8171)              10   1
(8172)              /5   8191
(8173)              4    1
(8174)              1    COUNT
(8175)              7    8181
(8176)              4    +0
(8177)              8    ;-6
(8178)              +0        (Literal)
(8179)   COUNT      +8192 - Size of Store
```

This is entered, from initial instructions, at 8177 with the Accumulator and B-register clear; and clears location 8192 upwards, then returns to initial instructions.

Since the words of the loader, tail, and 2 clear-stores are read by initial instructions (as distinct from the words of the body, which are read by the loader) they are punched as 4 tape characters each, 3 of the characters representing the bits of the word as defined earlier, and the remaining character, which precedes the word, acting as a marker. Track 4 of this marker must be a 1 but the other bits have no significance (other than to indicate the nature of the program used to punch the tape).

The loader, tail, and 2 clear-stores are each preceded by the 3 words:-

```
(8177)              0    8179
(8178)              8    8182
(8179)              -N
```

which instruct initial instructions to load the N following words.

The whole tape starts and ends with 180 blanks. The clear stores, if present, are separated from each other and from the loader by 4 blanks.

Note that location 8167, being a workspace, is not punched as part of the loader, but that 8175, being part way through the loader, is punched for simplicity as +0. The literal +0 in the loader is obtained from location 1, which, although not always zero, always has zeros in its bottom 13 bits, which is sufficient.

Thus, the items which occur on the binary tape can be represented diagramatically as:-